Project 1,FYS-STK4155, Linear Regression

Knut Hauge Engvik

October 6, 2019

Abstract

The aim of this project is to evaluate three common methods of regression analysis. Ordinary least squares, Ridge regression and LASSO regression.

The methods were first evaluated on a data set generated by a known function to determine validity of the algorithms and code. Then, the methods were applied to a set of geographical data to see if the algorithms could make sensible predictions on this type of data.

Introduction

The goal of regression analysis is to infer a relationship between a set of predictors and the data one is to analyze. Ordinary least squares (OLS), Ridge and LASSO regression all assume a linear relationship between the predictors and the response variables. That is, it assumes that for predictors $\boldsymbol{a_i}$ and data $\boldsymbol{y}$ we are looking for a solution to the equation

$$b_0 + b_1\boldsymbol{a_1} + b_2\boldsymbol{a_2}\cdots + b_n\boldsymbol{a_n} = y \tag{1}$$

With more than one data point his becomes a system of linear eqations

$$A\boldsymbol{b} = \boldsymbol{y} \tag{2}$$

The equation 2Introductionequation.0.2 usually does not have a solution when the number of data points is large, i.e. the system is overdetermined. It is however, by definition, solvable if one replaces $\boldsymbol{y}$ with $\tilde{\boldsymbol{y}}$, its projection onto the column space of A. One could then argue that $\tilde{\boldsymbol{y}}$ is the linear combination of the predictors that most closely resembles $\boldsymbol{y}$ and thus the best approximation. This is the basis for OLS, it minimizes the distance, in the L_2 norm, between $\boldsymbol{y}$ and $\tilde{\boldsymbol{y}}$.

One problem that might arise from using OLS is that it tends to overfit the training data leading to complex models that do not generalize well. To remedy this one can apply a kind of numerical Occam's razor by penalizing model complexity.

Both Ridge and Lasso regression penalize coefficient size using the L_2 and L_1 norms respectively. These two methods are similar, but not identical. As the L_1 norm is a linear penalty, the LASSO will tend to favor solutions along the predictor axis, i.e set some weights to zero. This does not happen using Ridge as the penalty rapidly approaches zero when the weights get small. Computationaly Lasso is more demanding as solutions depend on gradient decent methods.

Methods

Unless otherwise specified, data sets were generated using Franke's function, given by equation 3Methodsequation.0.3, for $x, y \in [0, 1]$. Then noise drawn from a normal distribution with mean 0 and standard deviation $0.1, 0.3$ or 0.6 was added to each data point. The three standard deviations will be referred to as lo, medium and high noise respectively.

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

Throughout this project the predictors used were polynomial terms in the coordinates x,y for some data point. Thus, for polynomial order 2, the design matrix would take the form

$$A = \begin{bmatrix} 1 & x & y & x^2 & xy & y^2 \\ 1 & x & y & x^2 & xy & y^2 \\ \vdots & & \ddots & & \vdots & \\ 1 & x & y & x^2 & xy & y^2 \end{bmatrix} \tag{4}$$

Results

1 OLS on Franke's function

For the project we were asked to write our own code and perform OLS regression on data from Franke's function and then calculate the mean squared error(MSE) and the coefficient of determination R2 for models employing polynomial orders up to 5. From figure 1This figure shows MSE and R2 values using OLS and polynomials of order 1 to 5 on data generated with Franke's functionfigure.1 one can see that both metrics improve with increasing model complexity, i.e. with the order of the polynomial used to approximate the data set. This trend

continued for higher order polynomials and peaked around polynomial order 10. At this point the MSE was approximately equal to the irreducible error introduced by the noise added to the data generating function. That is $MSE \approx 0.1^2 = var(noise)$.

A table of the variance in the coefficients is included in figure 295% confidence interval bounds for polynomial coefficients. Estimated using OLS and polynomials of order 1 to 5 on data generated with Franke's function. For instance the confidence interval for the first coefficient for polynomial order 1, the intercept, would be $\beta_0 \pm 0.002198$figure.2. As one can see the confidence interval widens with increasing complexity. This was even more evident when fitting polynomials of order higher than 5. At polynomial order 10 and up the covariance matrix of the coefficients became unstable returning run time errors and NaN entries in the matrix.

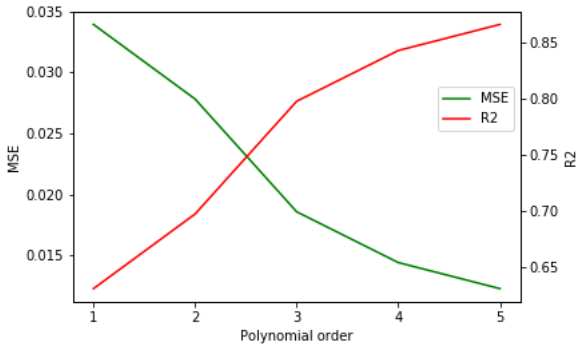

Figure 1: This figure shows MSE and R2 values using OLS and polynomials of order 1 to 5 on data generated with Franke's function

Beta	Order 1	Order 2	Order 3	Order 4	Order 5
0	0.0021983	0.0033529	0.0035005	0.004023	0.0045025
1	0.0028677	0.0099642	0.0181731	0.0322516	0.0517841
2	0.0028677	0.0099642	0.0181731	0.0322516	0.0517841
3	0	0.0088433	0.035747	0.1063568	0.2541353
4	0	0.0079023	0.0284457	0.082964	0.1974448
5	0	0.0088433	0.035747	0.1063568	0.2541353
6	0	0	0.0225672	0.1467227	0.57554
7	0	0	0.0197713	0.111892	0.4236825
8	0	0	0.0197713	0.111892	0.4236825
9	0	0	0.0225672	0.1467227	0.57554
10	0	0	0	0.0711307	0.6030137
11	0	0	0	0.0617961	0.4573117
12	0	0	0	0.0605874	0.4244523
13	0	0	0	0.0617961	0.4573117
14	0	0	0	0.0711307	0.6030137
15	0	0	0	0	0.2364864
16	0	0	0	0	0.2042844
17	0	0	0	0	0.1986103
18	0	0	0	0	0.1986103
19	0	0	0	0	0.2042844
20	0	0	0	0	0.2364864

Figure 2: 95% confidence interval bounds for polynomial coefficients. Estimated using OLS and polynomials of order 1 to 5 on data generated with Franke's function. For instance the confidence interval for the first coefficient for polynomial order 1, the intercept, would be $\beta_0 \pm 0.002198$.

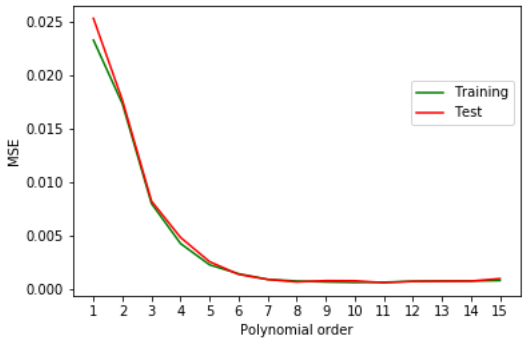

Figure 3: This figure shows the MSE using OLS and polynomials of order 1 to 15 on data generated with Franke's function. The data was split into training and testing sets using scikitlearns traintestsplit algorithm

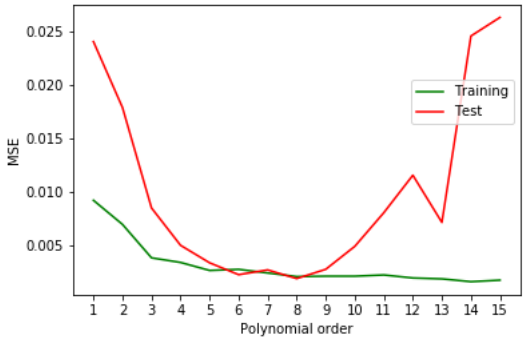

Figure 4: This figure shows average MSE for 5fold cross validation using OLS and polynomials of order 1 to 15 on data generated with Franke's function

2 Bias variance tradeoff

The mean squared error can be expressed as the sum of the variance,the square of the bias and an irreducible error term. Consider the equation

$$\begin{aligned} &\boldsymbol{E}\left[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2\right] = \boldsymbol{E}\left[(\boldsymbol{y}-\boldsymbol{E}[\tilde{\boldsymbol{y}}]+\boldsymbol{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})^2\right] = \\ &\boldsymbol{E}\left[(\boldsymbol{y}-\boldsymbol{E}[\tilde{\boldsymbol{y}}])^2+(\boldsymbol{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})^2+2(\boldsymbol{y}-\boldsymbol{E}[\tilde{\boldsymbol{y}}])(\boldsymbol{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})\right] \end{aligned} \tag{5}$$

Under the assumption that $\boldsymbol{y} = f + \epsilon$, we observe that when distributing expectations the last term becomes

$$2\left(\boldsymbol{E}\left[(f-\boldsymbol{E}[\tilde{\boldsymbol{y}}])(\boldsymbol{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})\right]+\boldsymbol{E}\left[\epsilon(\boldsymbol{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})\right]\right) \tag{6}$$

The left term is 0 as $(f - \boldsymbol{E}[\tilde{\boldsymbol{y}}])$ is a constant and $(\boldsymbol{E}[\tilde{\boldsymbol{y}}] - \tilde{\boldsymbol{y}})$ has expectation 0. The right term is 0 if ϵ is independent of $\tilde{\boldsymbol{y}}$ and we are left with

$$\boldsymbol{E}\left[(\boldsymbol{y}-\boldsymbol{E}[\tilde{\boldsymbol{y}}])^2+(\boldsymbol{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})^2\right] \tag{7}$$

Again, using that $\boldsymbol{y} = f + \epsilon$, this becomes

$$\begin{aligned} &\boldsymbol{E}\left[(f+\epsilon-\boldsymbol{E}[\tilde{\boldsymbol{y}}])^2+(\boldsymbol{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})^2\right] = \\ &\boldsymbol{E}\big[(f^2+\epsilon^2+(\boldsymbol{E}[\tilde{\boldsymbol{y}}])^2+2f\epsilon-2f\boldsymbol{E}[\tilde{\boldsymbol{y}}]-2\epsilon\boldsymbol{E}[\tilde{\boldsymbol{y}}])+(\boldsymbol{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})^2\big] \end{aligned} \tag{8}$$

Distributing the expectation operator yields

$$\boldsymbol{E}\big[(f-\boldsymbol{E}[\tilde{\boldsymbol{y}}])^2\big]+\boldsymbol{E}\big[\epsilon^2+2f\epsilon-2f\boldsymbol{E}[\tilde{\boldsymbol{y}}]-2\epsilon\boldsymbol{E}[\tilde{\boldsymbol{y}}]\big]+\boldsymbol{E}\big[(\boldsymbol{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})^2\big] \tag{9}$$

The middle term in equation 9Bias variance tradeoffequation.2.9 yields all zeros except for $\boldsymbol{E}[\epsilon^2]$ which is the variance of the irreducible error term. We are left with

$$\boldsymbol{E}\big[(f-\boldsymbol{E}[\tilde{\boldsymbol{y}}])^2\big]+\boldsymbol{E}\big[(\boldsymbol{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})^2\big]+\sigma^2 \tag{10}$$

In component form this can be rewritten as.

$$\frac{1}{n}\sum_i(f_i-\boldsymbol{E}[\tilde{\boldsymbol{y}}])^2+\frac{1}{n}\sum_i(\boldsymbol{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}}_i)^2+\sigma^2 \tag{11}$$

The first term is the square of the bias. That is, it is the square of the difference between the expected value of the estimator and the actual value of f. The second term is the variance of the estimator and the third term is the variance of the noise.

Looking back at the MSE observed using OLS, the trend was that the MSE of the training set decreased as a function of polynomial order while the MSE on the testing could suddenly blow up. When not using resampling, as seen in figure 3This figure shows the MSE using OLS and polynomials of order 1 to 15 on data generated with Franke's function. The data was split into training and testing sets using scikitlearns traintestsplit algorithmfigure.3, the trend was not obvious and the results were largely dependent on the random seeds used

for splitting the data. When resampling was employed, as seen in figure 4This figure shows average MSE for 5fold cross validation using OLS and polynomials of order 1 to 15 on data generated with Franke's functionfigure.4, this trend became more apparent.

From equation 11Bias variance tradeoffequation.2.11 this increase in MSE should be a function of the bias, variance and noise variance. As the noise was kept constant and the OLS is an unbiased estimator, the increasing MSE should be entirely due to increasing variance.

3 Ridge regression

Ridge regression with λ ranging from 10^{-5} to 1 was used to analyze Franke's function. It seemed that, overall, a smaller λ yielded more accurate predictions. This was however not universally true. When both noise levels and polynomial orders were high, the smallest λ values displayed over fitting, with MSE scores for the testing data much higher than the MSE on the training set. In this case the intermediate λ values, around 10^{-2} to 10^{-3}, performed best. Sometimes even extreme values like $\lambda = 1$ would win out. Typical results can be seen in figure 5This figure shows MSE values using Ridge regression on data created with Franke's function and low levels of noisefigure.5 and 6This figure shows MSE values using Ridge regression on data created with Franke's function and high levels of noisefigure.6 for low and high noise respectively.

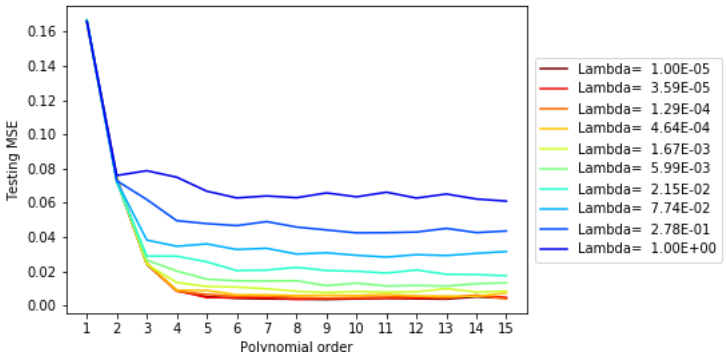

Figure 5: This figure shows MSE values using Ridge regression on data created with Franke's function and low levels of noise.

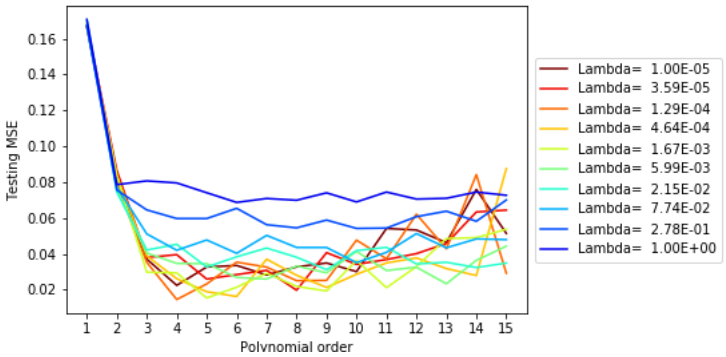

Figure 6: This figure shows MSE values using Ridge regression on data created with Franke's function and high levels of noise.

4 LASSO regression

For the LASSO α parmeters used were in the range 10^{-5} to 10^{-2}. Applying larger penalties than this resulted in the removal of more or less all predictors. As for the Ridge regression the lower penalties seemed to do best with medium to low penalties (10^{-5} to 10^{-4}) performing similarly at high noise, high model complexity. In comparison to the Ridge regression, LASSO seemed less affected by the model complexity, i.e the polynomial order used. This is probably due to the LASSO method's tendency to zero out predictors thus, in a sense, reducing model complexity. Typical results can be seen in figure 7This figure shows MSE values using LASSO regression on data created with Franke's function and low levels of noisefigure.7 and 8This figure shows MSE values using LASSO regression on data created with Franke's function and high levels of noisefigure.8 for low and high noise respectively.

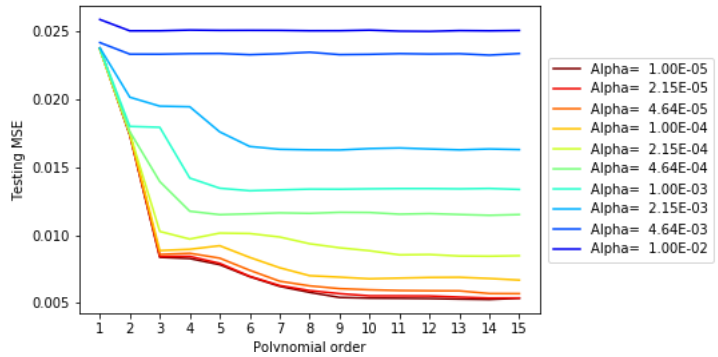

Figure 7: This figure shows MSE values using LASSO regression on data created with Franke's function and low levels of noise.

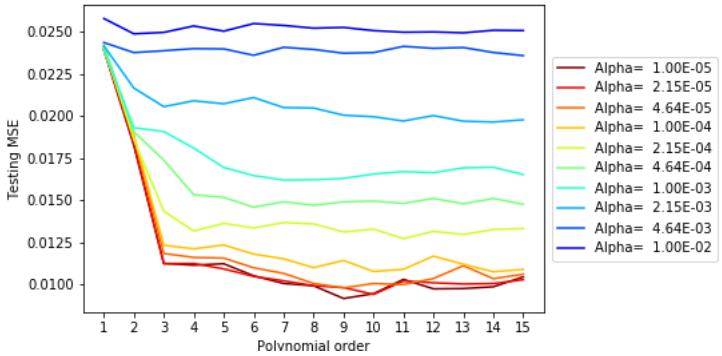

Figure 8: This figure shows MSE values using LASSO regression on data created with Franke's function and high levels of noise.

5 Analyzing terrain data

The data used in this part was not generated by Franke's function, but rather drawn from a gray scale image of Norwegian terrain (figure 9This figure shows a gray scale areal image of Norwegian terrainfigure.9). The task was to evaluate the predictive abilities of the OLS, Ridge and LASSO regression on this data set.

The number of data points in this image is $3601 * 1801$. Calculating the design matrix for higher order polynomials is thus rather time consuming and with multiple iterations for various polynomial orders and regression parameters it became prohibitive. As such, initial analysis using a wide range of parameters was performed on random, small $(100 * 100)$ squares of the image to narrow in on optimal parameters.

Comparing with the analysis on Franke's function there did not seem to be any negative effect for increasing model compexity. Even for OLS the MSE continued to decline and did not show any signs of increasing, even at polynomial orders higher than twenty.

One interesting thing to note is that the training and testing MSE were all but identical,see figures10This figure shows MSE values using OLS regression on terrain testing datafigure.10,11This figure shows MSE values using OLS regression on terrain training datafigure.11,12This figure shows MSE values using Ridge regression on terrain testing datafigure.12,13This figure shows MSE values using Ridge regression on terrain training datafigure.13,14This figure shows MSE values using LASSO regression on terrain testing datafigure.14 and 15This figure shows MSE values using LASSO regression on terrain training datafigure.15, implying that all three models had low variance. One cannot rule out that this stems from some overlooked error in the coding, but efforts were made to locate potential causes and none could be found. And, though the training and testing data looks identical, they did differ slightly.

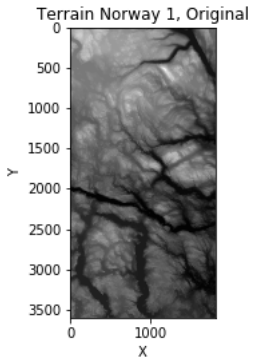

Figure 9: This figure shows a gray scale areal image of Norwegian terrain.

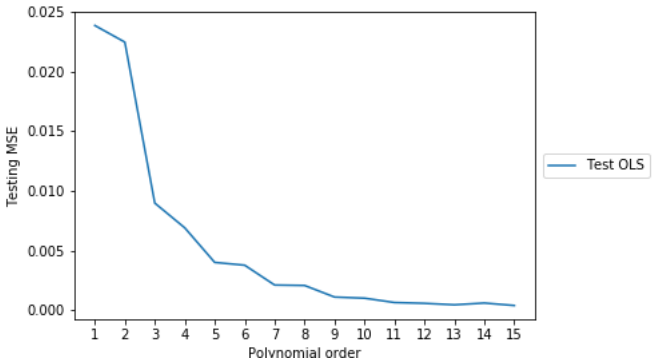

Figure 10: This figure shows MSE values using OLS regression on terrain testing data.

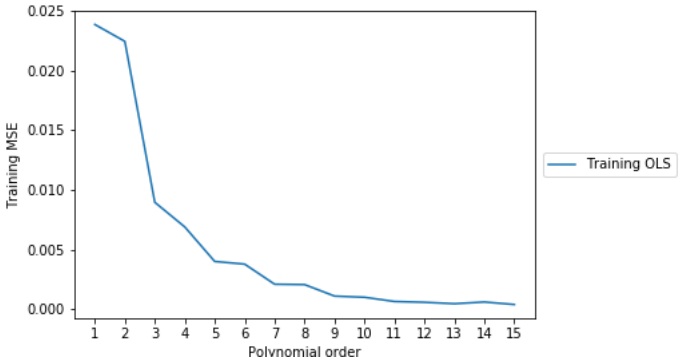

Figure 11: This figure shows MSE values using OLS regression on terrain training data.

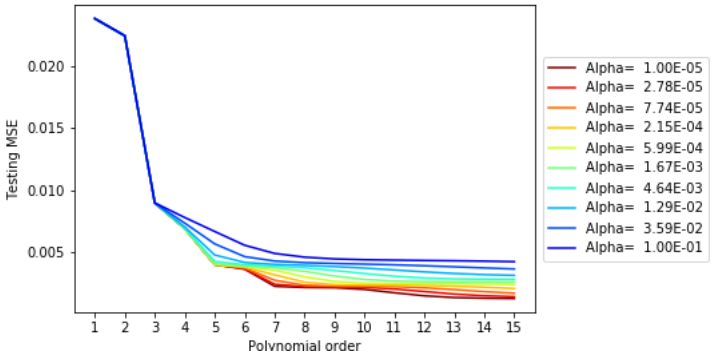

Figure 12: This figure shows MSE values using Ridge regression on terrain testing data.

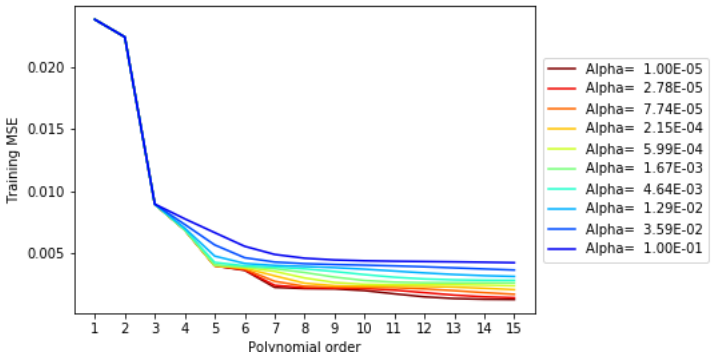

Figure 13: This figure shows MSE values using Ridge regression on terrain training data.

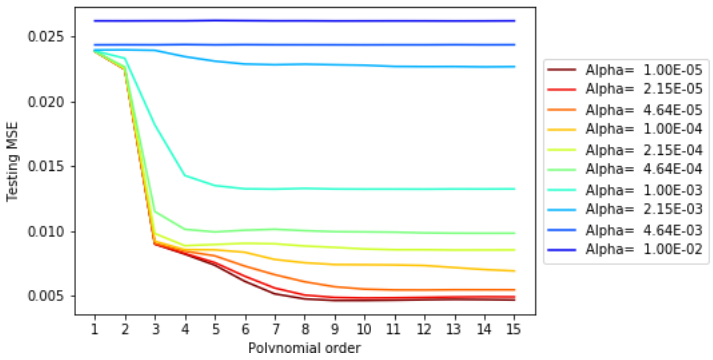

Figure 14: This figure shows MSE values using LASSO regression on terrain testing data.

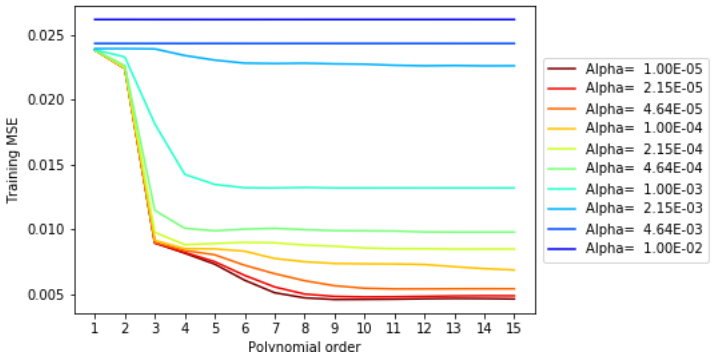

Figure 15: This figure shows MSE values using LASSO regression on terrain training data.

6 Rebuilding images

From the tests done on the sample squares, it seems like the best model should be of high complexity and low bias. Using polynomial orders 15 to 22, with both Ridge and Lasso parameters = 10^{-4}. The data points used to train the models were picked out as follows: For each 5th point along each axis, a datapoint was used for training. That is, the data points were all the elements with index on the form $(5i, 5j)$ in the data matrix. This amounts to roughly 4% of the data used for training.

The model was used to predict gray scale values of the remaining 96% of the data and the prediction was denormalized in an attempt to rebuild the original image. In figure 16This is he terrain image from Norway estimated from one in twenty five evenly spaced data points. The terrain was modeled using a polynomial of order 15. Left: Prediction using OLS. Middle: Prediction using Ridge. Right: Prediction using LASSOfigure.16 is shown the result using a model of polynomial order 15. While the images are rather smeared, one can pick out certain features of the original image. Particularly in the lower left corner, with thick, dark lines the OLS and Ridge regression methods seemed to pick up on some details. In figure 17This is he terrain image from Norway estimated from one in twenty five evenly spaced data points. The terrain was modeled using a polynomial of order 18. Left: Prediction using OLS. Middle: Prediction using Ridge. Right: Prediction using LASSOfigure.17 this is even clearer, using a polynomial of order 18. The LASSO did rather poorly in all cases.

One could ask if using these regression methods is a good strategy for analyzing terrain data of this form. The answer would probably depend on the goal of the analysis. There does not seem to be an a priori reason to assume that the x and y coordinates have any real relation to the data at a given coordinate. As such, it is expected that any predictive value such a model has would be nil outside the range of the training coordinates.

Consider the following two situations. First, if for training data one uses points spread out over the entire picture, as was done here to rebuild the picture in figure 9This figure shows a gray scale areal image of Norwegian terrainfigure.9. Then, as seen in figure 17This is he terrain image from Norway estimated from one in twenty five evenly spaced data points. The terrain was modeled using a polynomial of order 18. Left: Prediction using OLS. Middle: Prediction using Ridge. Right: Prediction using LASSOfigure.17, reasonable predictive capabilities can be demonstrated and this type of usage may have some applicability.

If however the goal is to generalize to other terrain images, this approach seems futile. If one considers one of the neighbouring terrain squares, say to the right, it is not obvious what coordinates should be used. Using the same coordinates would of course yield an identical terrain square. A natural extension would in this case lead to an analysis on coordinates with identical y values and x values shifted by the width, in data points, of the image. Predictions made would probably be erratic and bear little resemblance to reality, except maybe at the very margins of the original square.

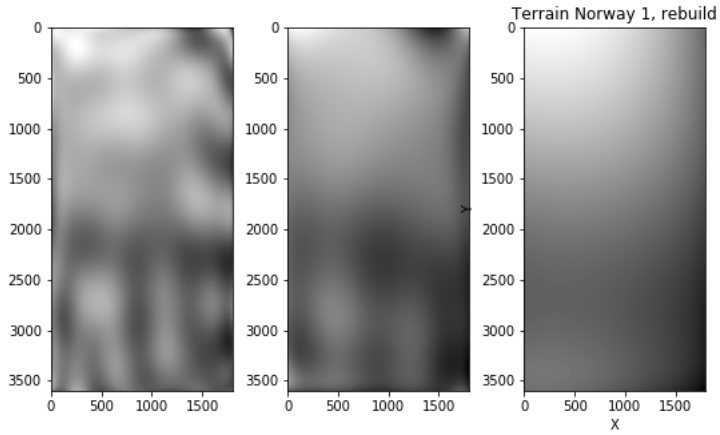

Figure 16: This is he terrain image from Norway estimated from one in twenty five evenly spaced data points. The terrain was modeled using a polynomial of order 15. Left: Prediction using OLS. Middle: Prediction using Ridge. Right: Prediction using LASSO

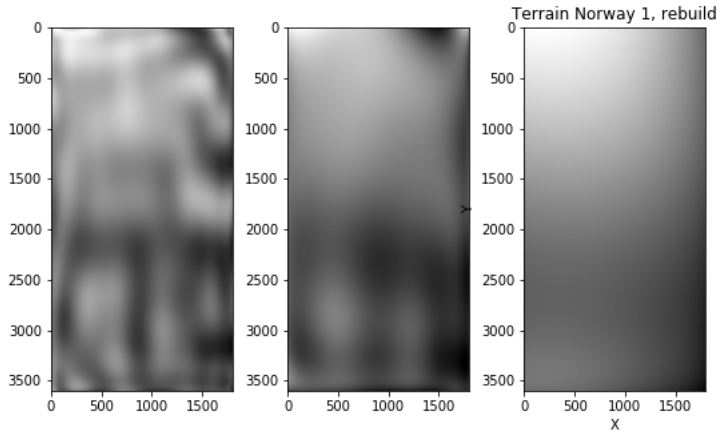

Figure 17: This is he terrain image from Norway estimated from one in twenty five evenly spaced data points. The terrain was modeled using a polynomial of order 18. Left: Prediction using OLS. Middle: Prediction using Ridge. Right: Prediction using LASSO

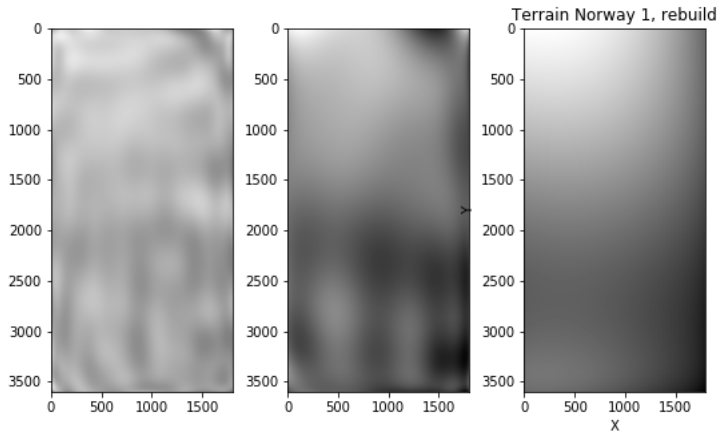

Figure 18: This is he terrain image from Norway estimated from one in twenty five evenly spaced data points. The terrain was modeled using a polynomial of order 22. Left: Prediction using OLS. Middle: Prediction using Ridge. Right: Prediction using LASSO

7 Conclusion

Observations made during this project may suggest a general approach to selecting an appropriate model for a given data set. First start with a low complexity, unbiased model. Increase complexity until the model displays acceptable results on training data. If the model shows similar results on testing data, all is good. If not, introduce bias to see if the variance can be reduced. In a bias variance tradeoff like this, the aim is to trade a large amount of variance for a small amount of bias.

All data and code can be found online at `https://github.com/KnutFys/projectone`.

References

[1] Wieringen,W.N.van, *Lecture Notes on Ridge Regression*, Version 0.30,2019, https://arxiv.org/pdf/1509.09169.pdf

[2] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. (2009) *The Elements of Statistical Learning (2.ed)*. New York: Springer-Verlag