FYS-STK4155: Project 1

Simen Nyhus Bastnes

10. October 2020

Abstract

In this project, we will study various methods of linear regression, namely the Ordinary Least Squares method, Ridge, and Lasso regression, as well as investigate resampling the data via Bootstrapping and k-fold Cross-Validation. The data we will be looking at is the so-called Franke function, as well as terrain elevation data from a region south-east in Norway. The Franke function gives us a way of testing our implementation before moving on to the more complex terrain data. For the Franke function, we found that Ridge regression performed the best for both Bootstrap and Cross-validation, when it comes to both accuracy and stability. Cross-validation gave the lowest estimate for the error. For the terrain data, we looked at a single patch from the map, and found that OLS performed a decent bit better than Ridge and Lasso, but since we did not reach the overfitting regime, the results might change if we had used a higher $p_{\max}$.

1 Introduction

With the emergence of more powerful computers, the field of machine learning is steadily becoming an integral part of both business and many fields of science. While many of the concepts and algorithms used in machine learning today has been known for a long time, some of them have simply been too computationally expensive to do efficiently. Linear regression is one of the simplest and most-studied forms of machine learning, and provides a good introduction to concepts commonly used in machine learning.

In this project, we will look at three different methods of regression analysis and compare how they fare against each other. The methods we will be using is the Ordinary Least Squares method, Ridge regression, and Lasso regression. We will also see how resampling the data affects the results from the regression methods, by implementing the Bootstrap algorithm and the k-fold Cross-Validation.

There are two different data sets that will be studied in this project. The first, is the Franke function from [1], as well as terrain data for a region in Norway taken from [2]. First, in Chapter 2 we will introduce the theory behind linear regression, as well as the the regression methods and resampling methods employed later in the project. In Chapter 3 we go through the implementation of the methods, explaining how the code is structured and used. Then, in Chapter 4 we go through the results of both the Franke function and the terrain data, while discussing them. Lastly, we conclude our findings in Chapter 5.

2 Theory

2.1 Linear regression

Linear regression is a method of fitting a set of p *predictors* $\boldsymbol{X}$ to a data set $\boldsymbol{y}$, while minimizing the error between the *response* $\tilde{\boldsymbol{y}}$ and the actual data $\boldsymbol{y}$. For each of the n samples y_i in the data set the relationship between the response and the predictors $\boldsymbol{X}_i$ is modeled in a linear fashion, giving us the following matrix equation

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

where $\beta = (\beta_0, \beta_1, ..., \beta_{p-1})^\mathsf{T}$ are the regression parameters we are trying to estimate, one for each predictor, and $\boldsymbol{\epsilon}$ is the error in our approximation. The matrix $\mathbf{X}$ is often called the design matrix, and the equation can be written a bit more explicitly as

$$y_i = \beta_0 + X_{i,1}\beta_1 + ... + X_{i,p-1}\beta_{p-1} + \epsilon_i$$

Exactly what each predictor is can vary a lot from case to case, and how the design matrix is set up is important for the accuracy of the fit. In our case, we will focus on a form of linear regression where the predictors is on the form of a polynomial in the input parameters. In the case where we have a data set $\boldsymbol{y}(\boldsymbol{x})$, the design matrix can for example be written on the form of

$$\mathbf{X} = (\boldsymbol{x}^0, \boldsymbol{x}^1, ..., \boldsymbol{x}^{p-1})$$

With that said, we still need some way to find the β's that fit the data best, and we will now look at three ways to try to do this.

2.1.1 Ordinary least squares

Following Chapter 2.3 of Hastie et al. [3], in order to find the optimal regression parameters β, the OLS method minimizes the residual sum of squares

$$\text{RSS}(\beta) = \sum_{i=1}^{N}(y_i - x_i^T\beta)^2$$

With $\boldsymbol{y}$ as the vector containing all N y_i, and X an $N \times p$ matrix as shown in section 2.1, this can be written as

$$\text{RSS}(\lambda) = (\boldsymbol{y} - \mathbf{X}\beta)^\mathsf{T}(\boldsymbol{y} - \mathbf{X}\beta)$$

Differentiating with respect to β we get

$$\frac{\partial \text{RSS}}{\partial \beta} = X^\mathsf{T}(\boldsymbol{y} - \mathbf{X}\beta)$$

In order to find an optimal β, this has to be zero

$$\begin{aligned}\mathbf{X}^\mathsf{T}(\boldsymbol{y} - \mathbf{X}\beta) &= 0\\ \mathbf{X}^\mathsf{T}\mathbf{X}\beta &= \mathbf{X}^T\boldsymbol{y}\end{aligned}$$

Finally giving us the expression for the optimal regression parameters

$$\beta = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\boldsymbol{y}$$

assuming that $\mathbf{X}^\mathsf{T}\mathbf{X}$ is invertible.

2.1.2 Ridge regression

Ridge regression is an example of a so-called shrinkage method, which shrinks the regression coefficients by adding a small penalty proportional to their size.

$$\beta^{\text{Ridge}} = \min_{\beta \in \mathbb{R}^p} \frac{1}{n} ||\boldsymbol{X}\beta - \boldsymbol{y}||_2^2 + \lambda ||\beta||_2^2$$

where λ is a regularization parameter that controls the amount of shrinkage, and we $||\beta||_2^2 \leq t$ where t is a finite number larger than zero. The higher λ, the more shrinkage occurs. This can be shown to give the Ridge solution

$$\beta^{\text{Ridge}} = (\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda I)^{-1}\mathbf{X}^\mathsf{T}\boldsymbol{y}$$

The aim with Ridge regression is to limit the potential problems with singularities when computing the inverse of $\mathbf{X}^\mathsf{T}\mathbf{X}$, which can be a problem when there are many correlated variables.

2.1.3 Lasso regression

Lasso regression is another shrinkage method, with a slightly different optimization equation compared to Ridge regression

$$\beta^{\text{Lasso}} = \min_{\beta \in \mathbb{R}^p} \frac{1}{n} ||\boldsymbol{X}\beta - \boldsymbol{y}||_2^2 + \lambda ||\beta||_1$$

where $||\beta||_1$ is the L_1 norm.

2.2 Bias-Variance decomposition

Starting from the so-called cost function, we want to derive an expression for the variance and bias for the predicted values $\tilde{\boldsymbol{y}}$

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

assuming that the true data is generated from a noisy model $\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\epsilon}$ where ϵ is normally distributed with zero mean and standard deviation σ^2, we get

$$\mathbb{E}[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2] = \mathbb{E}\Big[(f + \boldsymbol{\epsilon} - \tilde{\boldsymbol{y}})^2\Big]$$

We add and subtract $\mathbb{E}[\tilde{\boldsymbol{y}}]$ to the expression

$$\begin{aligned} &= \mathbb{E}\Big[(f + \boldsymbol{\epsilon} - \tilde{\boldsymbol{y}} + \mathbb{E}[\tilde{\boldsymbol{y}}] - \mathbb{E}[\tilde{\boldsymbol{y}}])^2\Big] \\ &= \mathbb{E}\Big[(f - \mathbb{E}[\tilde{\boldsymbol{y}}]) - (\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{\boldsymbol{y}}]) + \boldsymbol{\epsilon})^2\Big] \end{aligned}$$

Expanding the expression gives us

$$\begin{aligned} &= \mathbb{E}\Big[(f - \mathbb{E}[\tilde{\boldsymbol{y}}])^2 + (\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{\boldsymbol{y}}])^2 + \boldsymbol{\epsilon}^2 - 2(f - \mathbb{E}[\tilde{\boldsymbol{y}}])(\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{\boldsymbol{y}}]) \\ &+ 2(f - \mathbb{E}[\tilde{\boldsymbol{y}}])\boldsymbol{\epsilon} - 2(\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{\boldsymbol{y}}])\boldsymbol{\epsilon}\Big] \end{aligned}$$

We can use that the mean of $\boldsymbol{\epsilon}$ is zero, thus making the last two terms vanish $\mathbb{E}[\boldsymbol{\epsilon}] = 0$

$$= \mathbb{E}\Big[(f - \mathbb{E}[\tilde{\boldsymbol{y}}])^2 + (\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{\boldsymbol{y}}])^2 + \boldsymbol{\epsilon}^2 - 2(f - \mathbb{E}[\tilde{\boldsymbol{y}}])(\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{\boldsymbol{y}}])\Big]$$

For this to give the results we want, the last term should also be zero, but I don't see how, and often the derivation is just skipped over, or with confusing notation. Assuming it is zero, we get

$$\mathbb{E}[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2] = \mathbb{E}[(f - \mathbb{E}[\tilde{\boldsymbol{y}}])^2] + \mathbb{E}[(\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{\boldsymbol{y}}])^2] + \mathbb{E}[\boldsymbol{\epsilon}^2]$$

Using that $\mathbb{E}[\boldsymbol{\epsilon}^2] = \sigma^2$, we get the final expression for the bias-variance decomposition

$$\mathbb{E}[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2] = \mathbb{E}[(f - \mathbb{E}[\tilde{\boldsymbol{y}}])^2] + \mathbb{E}[(\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{\boldsymbol{y}}])^2] + \sigma^2$$

Where the left-hand side is the so-called mean squared error (henceforth referred to as MSE), the first term is the bias term, which measures how much the our model $\tilde{\boldsymbol{y}}$ differs from the true model $f(\mathbf{x})$. The second term is the variance of our model, while the third term is the noise. High bias indicates underfitting, and the model is not complex enough to capture the relevant features, while high variance can indicate that the model is overfitting, and adjusting to the smallest variations in the data set.

Thus, our goal in this project is to try to find the middle ground between high bias and high variance, and that will be the point where the MSE is the lowest, and thus it makes a lot of sense to use it as a measure for how good our model is.

The MSE can be written more explicitly as

$$\text{MSE}(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2$$

Another way of assessing the results would be the R^2 score

$$R^2(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{\boldsymbol{y}})^2}$$

where

$$\bar{\boldsymbol{y}} = \frac{1}{n}\sum_{i=0}^{n-1} y_i$$

2.3 Confidence intervals

For the OLS and Ridge regression cases, it is possible to derive the variance of β (a proper derivation is given in [4]), and thus the confidence intervals as well. For the OLS method, the variance is given by

$$\text{Var}(\beta) = \sigma^2(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}$$

where σ^2 is the estimated variance of y given by

$$\sigma^2 = \frac{1}{N - p - 1}\sigma_{i=1}^{N}(y_i - \tilde{y}_i)^2$$

Taking the square root of the diagonal of $(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}$ gives us an estimate of the variance of the j-th regression coefficient

$$\sigma^2(\beta_j) = \sigma^2\sqrt{[\mathbf{X}^\mathsf{T}\mathbf{X}]^{-1}_{jj}}$$

Letting us construct the 95% confidence intervals by

$$\mathrm{CI}(\beta_j) = \left[\beta_j - 2\sqrt{\sigma_2(\beta_j)}\,,\ \ \beta_j + 2\sqrt{\sigma_2(\beta_j)}\right]$$

Similarly, the variance for β for Ridge regression can be found to be

$$\mathrm{Var}[\beta^{\mathrm{Ridge}}] = \sigma^2[\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I}]^{-1}\mathbf{X}^\mathsf{T}\mathbf{X}[(\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I})^{-1}]^\mathsf{T}$$

and confidence interval can be constructed following the same steps as done above for OLS.

2.4 Resampling methods

In order to assess our models properly, we will be using some form of resampling. Specifically, we will be looking at the Bootstrap, and the standard k-fold Cross-validation resampling methods.

2.4.1 Bootstrap

The basic concept of the Bootstrap resampling method is to create new data sets by drawing samples (with replacement) from the training data set. Algorithm 1 shows the Bootstrap method given a data set $\mathcal{L}$ consisting of the data $\mathbf{X}_\mathcal{L} = \{(y_j, \boldsymbol{x}_j), j = 0 \ldots n-1\}$

Algorithm 1 Bootstrap

1: Split the data set $\mathbf{X}_\mathcal{L}$ into training $\mathbf{X}_{\mathcal{L},\mathrm{train}}$ and test data sets $\mathbf{X}_{\mathcal{L},\mathrm{test}}$
2: **for** $i = 0$, $N_\mathrm{bs} - 1$ **do**
3: Create a new data set $\mathbf{X}_{\mathcal{L},i}$ by drawing samples with replacement from $\mathbf{X}_{\mathcal{L},\mathrm{train}}$.
4: Fit the model using $\mathbf{X}_{\mathcal{L},i}$.
5: Evaluate the model on the test set $\mathbf{X}_{\mathcal{L},\mathrm{test}}$ and store the results.
6: **end for**
7: Assess the model by looking at the distribution of computed quantities, for example looking at the mean of the MSE.

2.4.2 Cross-validation

Cross-validation is a resampling method where (in the case of k-fold CV) the data set is split into k-folds of training and test data sets. Algorithm 2 shows the standard k-fold cross-validation.

Algorithm 2 k-fold Cross-Validation

1: Shuffle the data set $\mathbf{X}_\mathcal{L}$ randomly.
2: Split the data set $\mathbf{X}_\mathcal{L}$ into k folds/subsets $\{\mathbf{X}_{\mathcal{L},i}, i = 0 \ldots k-1\}$.
3: **for** $i = 0$, $k - 1$ **do**
4: Set $\mathbf{X}_{\mathcal{L},i}$ as the test data set and the rest of the folds as the training set.
5: Fit the model using the training data set as defined above.
6: Evaluate the model on the i-th test set $\mathbf{X}_{\mathcal{L},i}$ and store the results.
7: **end for**
8: Assess the model by looking at the distribution of computed quantities.

3 Implementation

3.1 Code

All scripts used to generate the results in this project can be found in the Github repository [5]. The code can be found in the `src` folder, while figures can be found in the `figures` folder. The data files for the terrain map as well as calculated results can be found in `datafiles`. The `benchmark` folder contains a data file for each section of the code with information about the parameters used, in order to easier verify that the program works.

The main program `main.py` is split into several different run modes, where the specific details of each one is detailed in table 1. When starting the program, it prompts you to input one of the following arguments (a,b,c,d,e,f,g), where (a-e) relates to the Franke function, and (f,g) to the terrain data.

Table 1: Explanation of each of the different run modes for the main program `main.py`.

Run mode	**Function**
a	Performs an OLS regression on the Franke function with a constant degree p. Prints out the MSE and R^2 score, and plots the variance of the regression coefficients β
b	Performs OLS with Bootstrap and CV on the Franke function. Saves to file the error, bias, and variance for each p, and plots all relevant plots.
c	Performs a Cross-validation and compares it to the one in Scikit-learn.
d	Runs Ridge regression with Bootstrap and CV for a set of λ values over a set of p values. Saves the results to file, and plots the results.
e	Same as c, expect done with Lasso regression
f	Creates plots of the patch we are looking at for the terrain map
g	Depending on what is set to the `reg_str` variable, performs one of the three regression methods with Bootstrap and CV on the terrain data. Results are saved to file and plotted.

The program first defines common variables needed for all the different parts of the program, and then goes into specific sections of the code relating to either the Franke function or the terrain data in order to initialize the data set that is to be used.

Then, there is a fairly long function that performs the actual regression over the polynomial degree p and hyperparameter λ for all parts excluding `a` and `f`. This function performs Bootstrap and CV with the regression method specified by the variable `reg_str`, returning a huge amount of arrays with all the results. This function is a result of restructuring in order to minimize the amount of duplicate code (where the regression method used is the major difference), in order to easier find and fix some bugs that were present in the code.

After that, each run mode has its own section of code to call the regression function, and plots/saves the results.

3.2 Data sets

In this project, we will be using two different data sets. The first, is the Franke function shown in appendix A.1. This function is a two-dimensional function that has been widely used for testing implementations for regression and interpolation.

The second data set is terrain elevation maps taken from [2], and the maps are stored in the GeoTIFF file format, and are from the SRTM Arc-Second Global data set. In the `datafiles` folder there are three maps, each of one region of Norway, though only one of them, `SRTM_data_Norway_3.tif` is studied in this project. This map is over the south-eastern part of Norway. A plot of the entire map, as well as a more detailed description of the location is given in figure 12 in appendix A.2.

4 Results and discussion

In this section, we will be showing some select results from running the code described in Chapter 3. More figures and data files, as well as some simpler benchmark runs can be found within their respective folders in the Github repository [5]. The results will be split into two sections, the first one pertaining to the Franke function, while the second is the analysis of the terrain data.

4.1 Franke function

First of all, we plot the Franke function with randomly drawn x, y samples, without noise, which can be seen in figure 1.

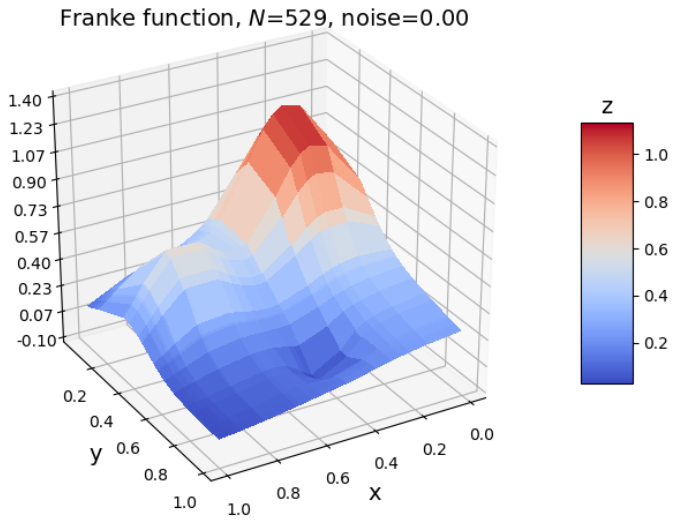


Figure 1: Example of the Franke function with no noise, with 529 randomly drawn points where $x, y \in [0, 1]$.

We start the regression analysis by performing the OLS method on the data set. Table 2 shows the MSE and R2 score for the train and test set where $p = 5$, $N = 529$. For this polynomial

degree, both the training and test set shows good results, indicating that there is no significant overfitting.

Table 2: Mean squared error and R2 score for the Franke function training and test set when $p = 5$, $N = 529$, and noise $\sigma = 0.05$

Data set	MSE	R2
train	0.003334	0.949904
test	0.004674	0.929855

The 95% confidence intervals for β_{OLS} is shown in figure 2, with the same parameters. The confidence intervals are quite close to the coefficients, which makes sense given the MSE and R2 scores we found earlier.

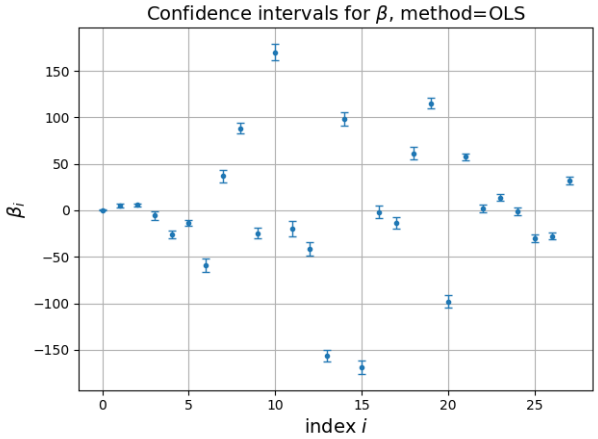
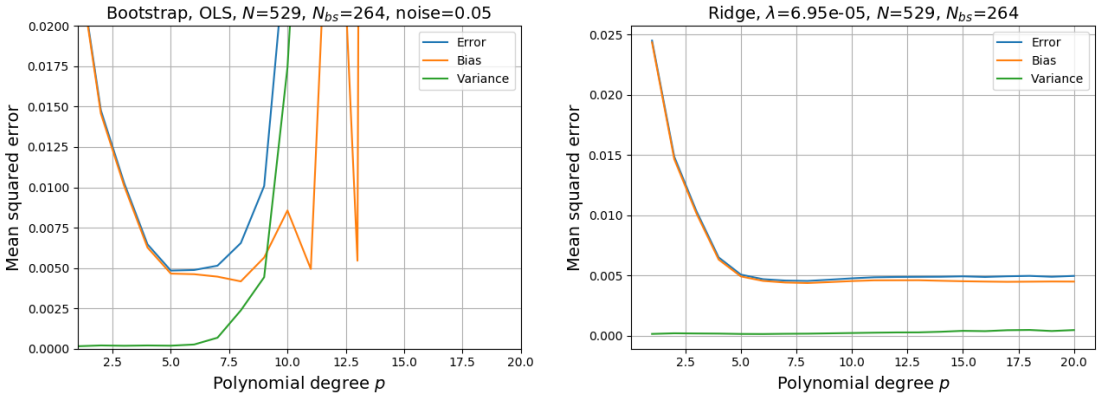


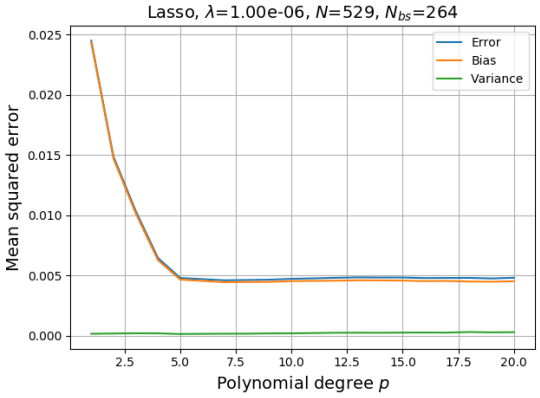
Figure 2: 95% confidence intervals for the regression coefficients β_{OLS} for $p = 6$, $N = 529$, and noise $\sigma = 0.05$.

To study how the model is affected by the degree p, we study the bias-variance trade-off by employing the Bootstrap resampling method. Now, we will be studying all three regression methods, OLS, Ridge and Lasso to make comparing easier. Figure 3 shows the bias-variance trade-off for all three methods, with the Ridge and Lasso ones corresponding to the λ that gave the lowest MSE. We see that for the OLS, once we go beyond $p = 7$, the error and variance skyrockets. For Ridge and Lasso however, everything is fairly stable for $p > 5$. This shows the effect of the regularization parameter λ, keeping the variance from exploding, and avoiding overfitting.



(a) OLS

(b) Ridge

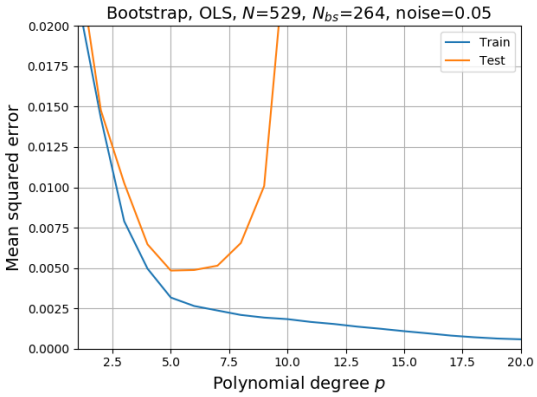


(c) Lasso

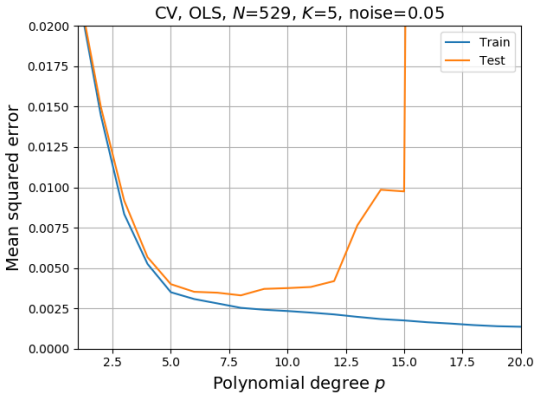
Figure 3: Bias-variance plots for OLS, Ridge, and Lasso regression, where $N = 529$, $\sigma = 0.05$ and $N_{\text{bs}} = N/2$. The best-fit hyperparameter for Ridge was found to be $\lambda = 2.98 \cdot 10^{-5}$, and $\lambda = 3.16 \cdot 10^{-5}$ for Lasso.

Figure 13 in appendix B shows how the OLS results depend on the number of bootstraps. The figure also is run with double the amount of data points, showing that the general shape is very similar, though the overfitting starts at slightly higher degree p. This is expected, as a with more data points, as there is more complexity in the data set. Also in the appendix, figure 14 shows the 95% confidence intervals of β for Ridge regression, which compared to the OLS confidence intervals shown in figure 2 appear to be slightly larger.

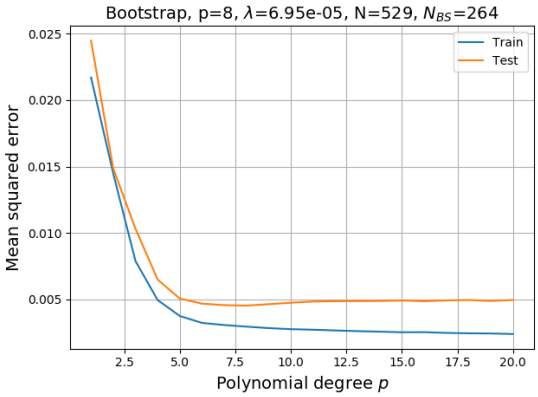
Before looking closer at the hyperparameters λ, we look at the results from the k-fold Cross-validation. Figure 4 shows the difference between the training and test MSE for all regression methods and both Bootstrap and Cross-validation. We see that CV gives very similar results to the Bootstrap, with some discrepancies. For OLS, the test MSE starts shooting up a lot later than for the Bootstrap results. For Ridge, the test MSE lies a lot closer to the training set up until $p = 15$, where it starts going slightly up. Finally, Lasso in general is closer to the training MSE.



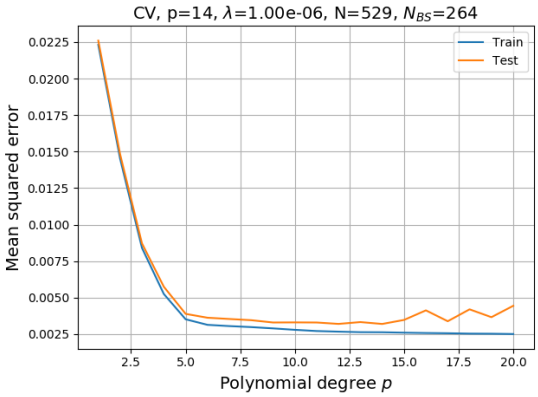
(a) OLS - Bootstrap



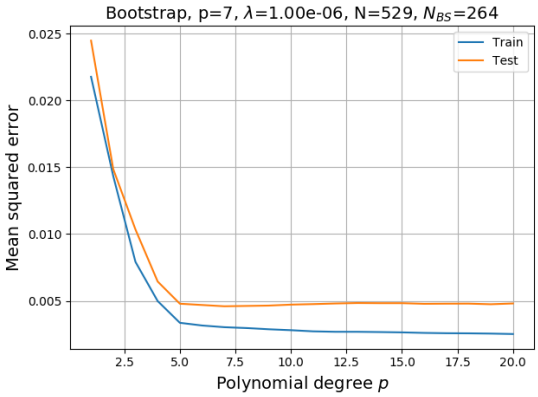
(b) OLS - Cross-validation



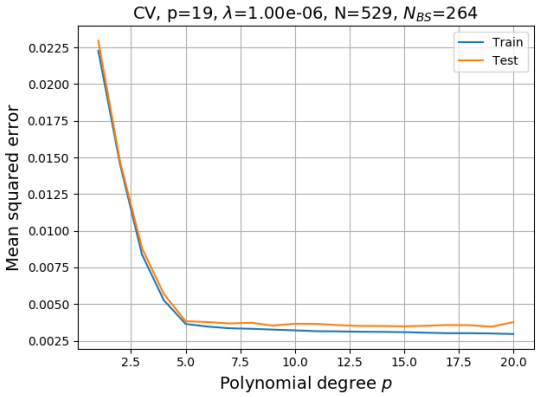
(c) Ridge - Bootstrap



(d) Ridge - Cross-validation



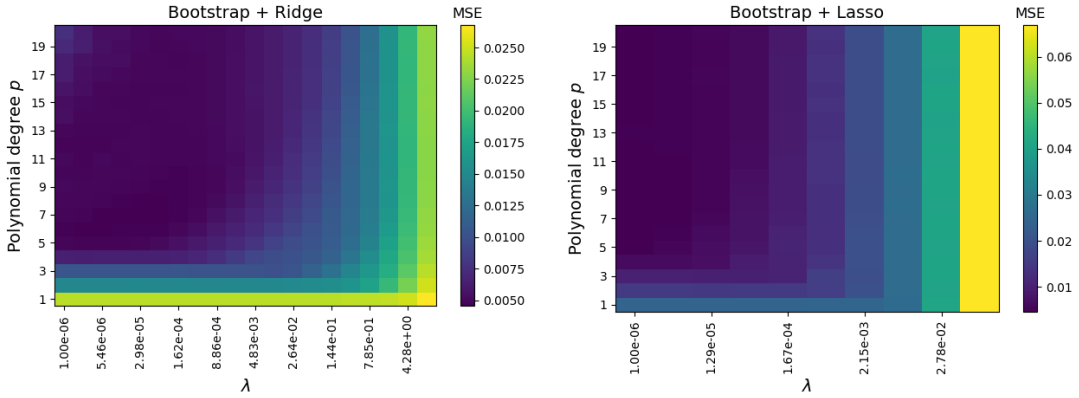
(e) Lasso - Bootstrap



(f) Lasso - Cross-validation

Figure 4: Bias-variance plots for OLS, Ridge, and Lasso regression, where $N = 529$, $\sigma = 0.05$ and $N_{\text{bs}} = N/2$. The best-fit hyperparameter for Ridge was found to be $\lambda = 2.98 \cdot 10^{-5}$, and $\lambda = 3.16 \cdot 10^{-5}$ for Lasso. The p-value on the Ridge + CV panel should have been the p-value corresponding to the minimum MSE value, but for some reason it is wrong.

One way of looking at the hyperparameters would be to look at the heat map from all the values of p and λ that we used. Figure 5 shows the MSE heat map for Ridge and Lasso using Bootstrap. Heat maps for the bias, variance, and MSE from CV can be found in the Github repository [5]. It is clear that the model prefers lower λ, though for high model complexity/polynomial degree a slightly higher λ is preferred. Ridge has a slightly larger space of parameters that give good results, while the Lasso results start getting worse faster as λ is increased.



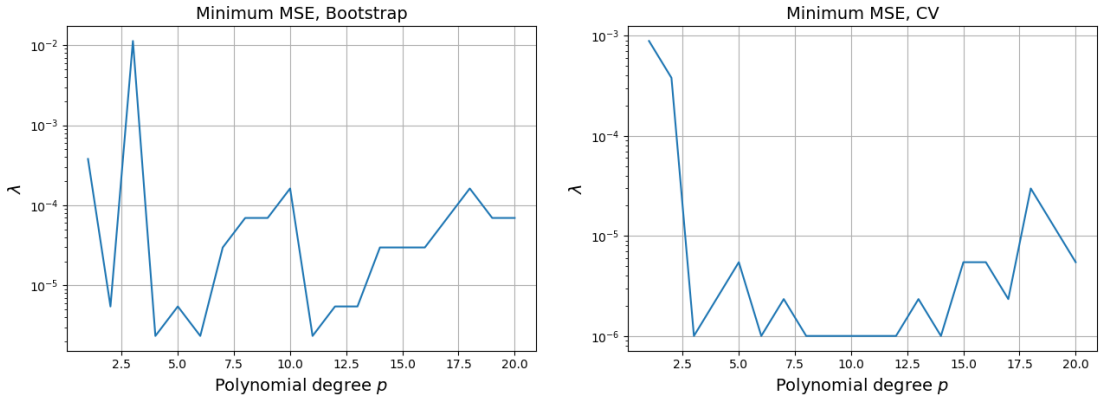
(a) Ridge

(b) Lasso

Figure 5: Heat maps showing the MSE plotted against polynomial degree p and the hyperparameter λ for both Ridge and Lasso, using Bootstrap resampling. For Ridge, we calculated the results for 20 log-spaced λs between $1 \cdot 10^{-6}$ and $1 \cdot 10^{1}$, while for Lasso we calculated 10 values of λ between $1 \cdot 10^{-6}$ and $1 \cdot 10^{-1}$.

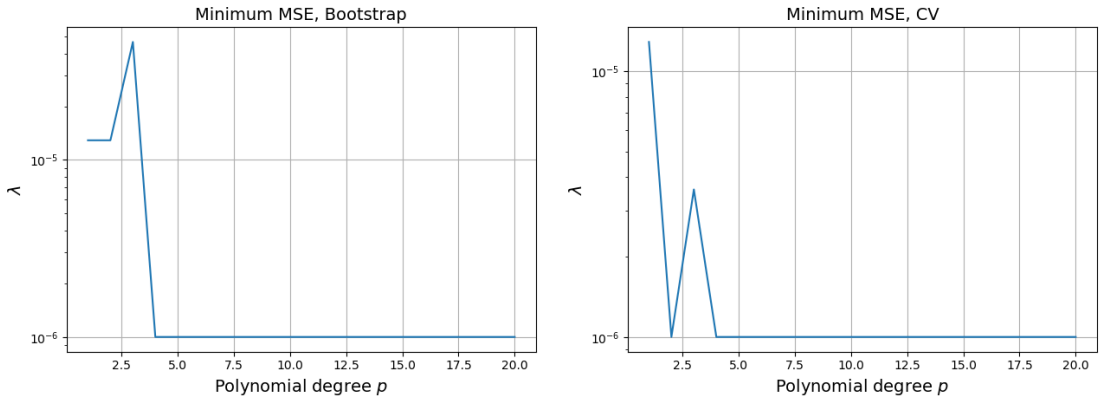
Figure 6 shows λ_{min} plotted against p for both Ridge and Lasso where λ_{min} is the λ corresponding to the lowest MSE for that polynomial degree. This is done both for Bootstrap and Cross-validation. For Ridge and Bootstrap, the optimal λ varies quite a bit for each value of p, while for CV, it stays at the lowest value for $p \in [8, 12]$, where there is a peak in the Bootstrap. The axes are slightly different, so plotting them on top of each other would have been interesting.

For Lasso, the λ value drops down to the lowest value at $p \sim 3 - 4$ for both Bootstrap and CV, staying there until the end. This shows that we probably should have tried running with even lower values. Running with the starting point $\lambda = 1 \cdot 10^{-6}$ took extremely long time to converge (or just reach the maximum iteration count), so trying with any lower is not very tempting.



(a) Ridge - Bootstrap

(b) Ridge - Cross-validation



(c) Lasso - Bootstrap

(d) Lasso - Cross-validation

Figure 6: Hyperparameter λ plotted against polynomial degree p for Ridge/Lasso regression with Bootstrap/Cross-validation. The λ value corresponds to the one that gives the lowest MSE value at the given p.

Finally, table 3 shows the best-fit parameters p and λ for each of the three regression methods, and for each of the resampling methods. For both Bootstrap and Cross-validation, Ridge regression seems to be the best option, with both OLS and Lasso only slightly behind. For the Bootstrap, Lasso performs better than OLS, while for CV, OLS performs better. Looking at the R2 scores could have been useful to see how significant those differences are, but based on the train/test MSE plots in figure 4, the differences are fairly small. Worth noting out is that while Ridge and Lasso are very similar results wise, Ridge is quite a bit faster, as Lasso can be extremely slow since there is no analytical solution. OLS on the other hand is even faster than Ridge, but has a significant disadvantage when it comes to stability. As shown with both the bias-variance trade-off and train-test plots, Ridge and Lasso has a much wider range of parameters that give good results, as the hyperparameter suppresses the overfitting problem with OLS.

Table 3: The best-fit parameters for OLS, Ridge and Lasso regression given the lowest MSE value. First three columns is the Bootstrap, last three is Cross-validation

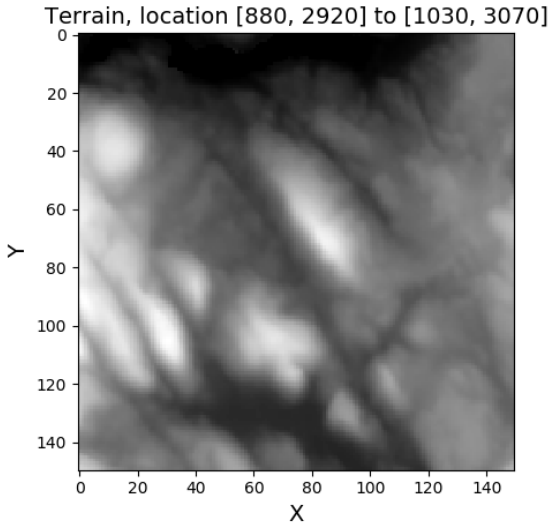
Reg. method	p (BS)	λ (BS)	MSE (BS)	p (CV)	λ (CV)	MSE (CV)
OLS	5	-	$4.84 \cdot 10^{-3}$	8	-	$3.30 \cdot 10^{-3}$
Ridge	8	$6.95 \cdot 10^{-5}$	$4.55 \cdot 10^{-3}$	15	$5.46 \cdot 10^{-6}$	$3.28 \cdot 10^{-3}$
Lasso	7	$1.00 \cdot 10^{-6}$	$4.59 \cdot 10^{-3}$	19	$1.00 \cdot 10^{-6}$	$3.46 \cdot 10^{-3}$

Before moving on to the terrain data, to point out that for some reason the p and λ values present in the title of figure 4 for the CV+Ridge panel specifically for some reason is incorrect. Quite a bit of time was spent looking at the code and output files without understanding why that one specifically is wrong.

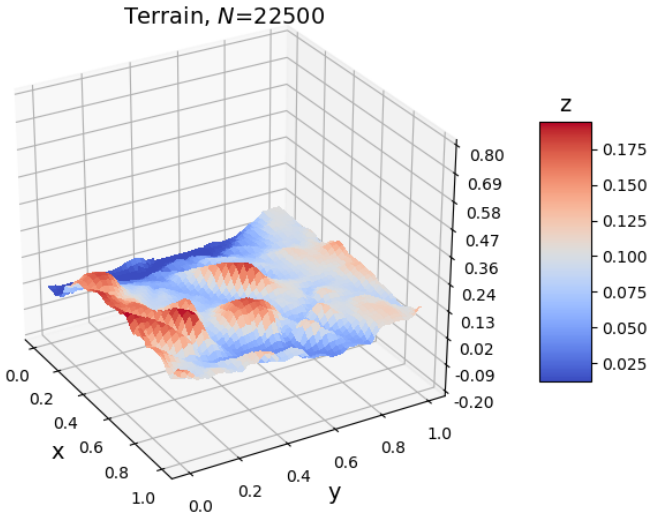
4.2 Terrain data

We look at the terrain map described in chapter 3.2. The entire map is 3601×1801 data points, so performing the analysis on the entire map is outside of the scope of this project, so instead we will be focusing on a smaller region of the map. Ideally, we would have performed this analysis on many patches of the map, but sadly due to time constraints we will only have the results for this one patch. Altering which area and the size of it is relatively easy in the code, should one desire to test it out.

Figure 7 shows the zoomed in map and a 3D surface plot of the patch we will be studying in this section. First, we normalize the map by the maximum height in the dataset, and choose a 150×150 grid for our region, giving us a data set of size 22500 samples, quite a bit more than we had for the Franke function. The region we chose is located at (x, y) $(880, 2920)$ to $(1030, 3070)$, and the center of the map corresponds roughly to where the author grew up. Beyond that, the area consists of several fairly sharp mountains/hills and decent variation in the features.



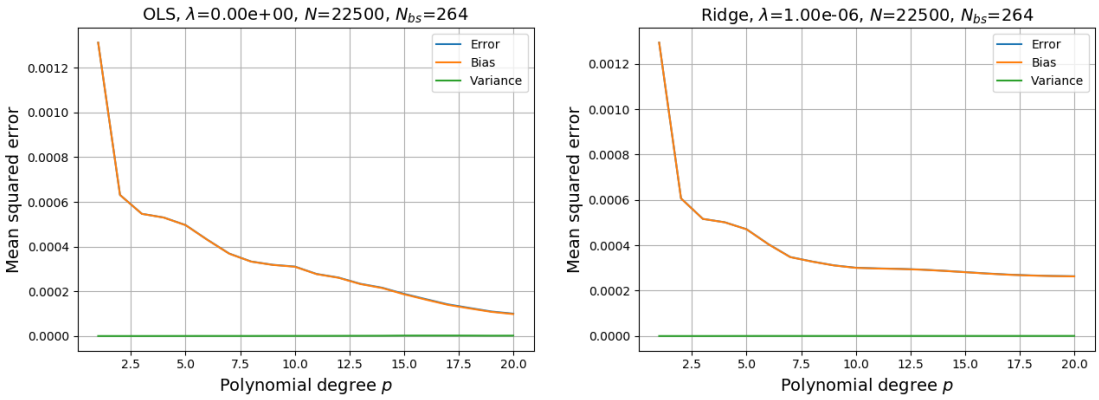
(a) Terrain



(b) 3D surface plot

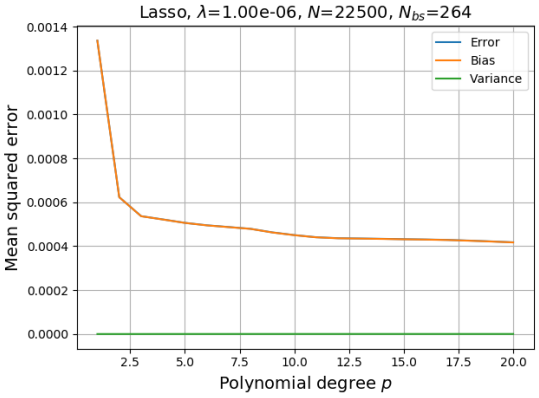
Figure 7: Top: Zoomed in view of the map. Bottom: 3D surface plot of the same region. The region we are studying is located at (x, y) $(880, 2920)$ to $(1030, 3070)$, with $N = 22500$ samples.

Figure 8 shows the bias-variance trade-off for all regression methods using Bootstrap. The OLS keeps decreasing as the degree increases, and seems to have a lower error than both Ridge and Lasso for roughly $p > 11$. Since we have so many data points, it might be that the OLS does not yet start overfitting, and possibly the noise level of the data is low enough to push this threshold further up. Testing with higher values of p would probably be the way forward, but doing so takes significantly more time, as the number of terms in the design matrix keeps increasing by $p+2$ for every new degree up you go, with 231 terms for $p = 20$, 496 for $p = 30$, 861 for $p = 40$.



(a) OLS

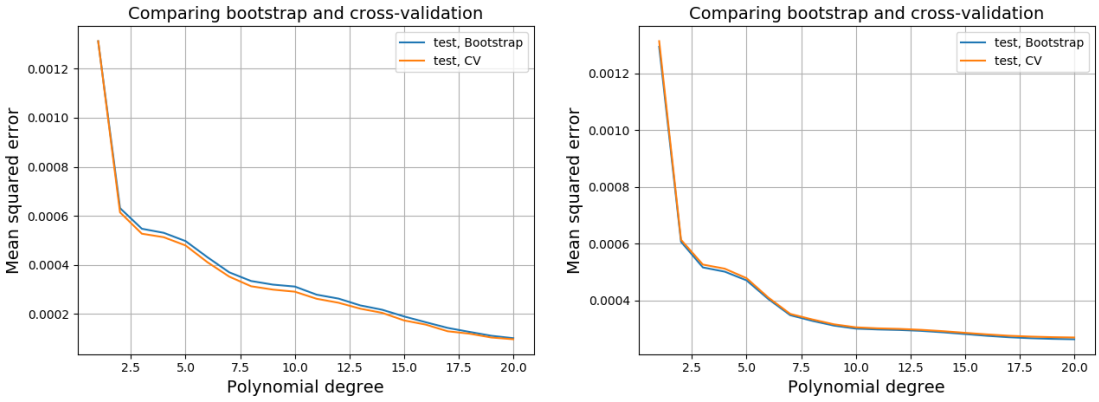
(b) Ridge



(c) Lasso

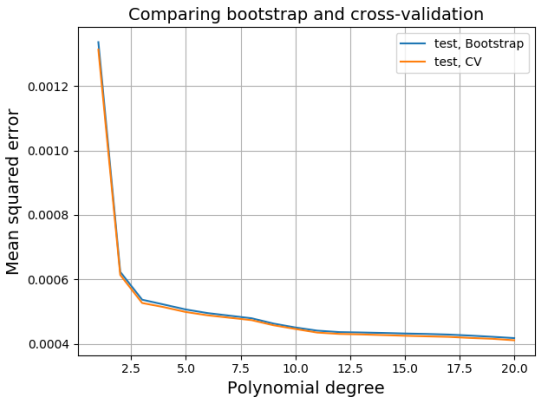
Figure 8: Bias-variance plots for OLS, Ridge, and Lasso regression, for the terrain patch, $N = 22500$, $N_{\text{bs}} = 264$.

Figure 9 shows the direct comparison between the test MSE from Bootstrap and Cross-validation, computed for each of the regression methods. We see that the results are quite close, with Bootstrap giving a slightly lower error for higher p-values for Ridge, which contrasts the results for the Franke function where Cross-validation gave a lower test error than Bootstrap, though the difference for the terrain data is much smaller than it was for the Franke function. For OLS and Lasso, Bootstrap has a slightly higher error than CV.



(a) OLS

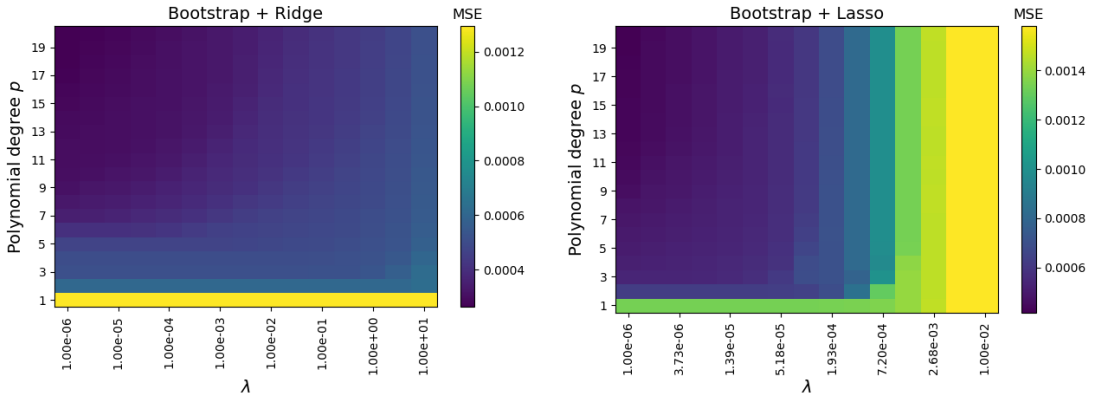
(b) Ridge



(c) Lasso

Figure 9: Comparison between the test MSE from Bootstrap and Cross-validation, for each regression method. Data set is the terrain data described earlier, $N = 22500$, $N_{\text{bs}} = 264$.

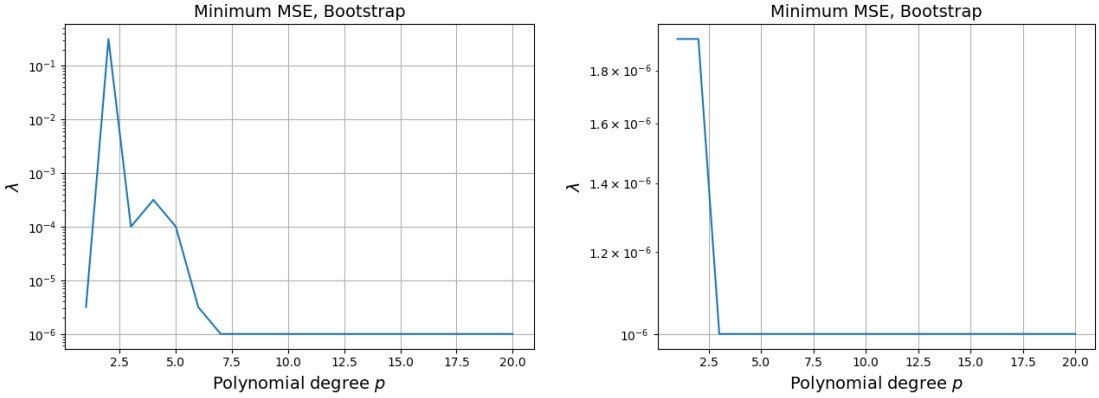
To study the hyperparameter, we again start by plotting the heat maps for Ridge and Lasso. Figure 10 shows the MSE heat maps when using Bootstrap. The Cross-validation results can be found in appendix C, figure 15. We see that the Ridge results shows a smaller dependence on the hyperparameter compared to the Franke function, as even the highest λs give decent results. For Lasso, the results are more in line with the Franke function, with a much smaller range of viable λ values. Note the difference in the x-axis range.



(a) Ridge (b) Lasso

Figure 10: Heat maps showing the MSE plotted against polynomial degree p and the hyperparameter λ for both Ridge and Lasso, using Bootstrap resampling. For Ridge, we calculated the results for 15 log-spaced λs between $1 \cdot 10^{-6}$ and $1 \cdot 10^{1}$, while for Lasso we calculated 15 values of λ between $1 \cdot 10^{-6}$ and $1 \cdot 10^{-2}$. $N_{\text{bs}} = 264$.

Figure 11 shows λ_{min} plotted against p for both Ridge and Lasso where λ_{min} is the λ corresponding to the lowest MSE for that polynomial degree. This is done both with both Bootstrap and CV, but only Bootstrap is shown here since they show the same trends. We see that both Ridge and Lasso goes towards the minimum λ value and stays there as p is increased. This happens faster for Lasso regression at $p = 3$, and $p = 7$ for Ridge.



(a) Ridge - Bootstrap (b) Lasso - Bootstrap

Figure 11: Hyperparameter λ plotted against polynomial degree p for Ridge/Lasso regression with Bootstrap. The λ value corresponds to the one that gives the lowest MSE value at the given p.

Finally, we summarize our results for the terrain data in a table. Table 4 shows the optimal (p, λ) values and the corresponding MSE value for all three regression methods and both resampling methods. Unsurprisingly, all the p values end up being $p = 20$, the highest model complexity,

which fits what we observed in figure 8 and figure 9. Beyond that, we see that OLS gives the best result for the terrain data by a decent margin. Lasso performs the worst of the bunch, with a similar difference to Ridge as was between OLS and Ridge. We also see that Bootstrap and Cross-validation give very similar results, being much closer to each other compared to earlier with the Franke function.

It is clear that in order to properly evaluate the regression methods, a higher p value is needed, which sadly is difficult to do time wise. Also, in order to do so, it would probably be a good idea to try to optimize the code in order to speed up some of the loops, and potentially parallelizing some of them. It would also be ideal to try to test out more regions of the map and not just this one, to see if perhaps a region that is more uneven yields different results.

Table 4: The best-fit parameters for OLS, Ridge and Lasso regression for the terrain patch, given the lowest MSE value. First three columns is the Bootstrap, last three is Cross-validation.

Reg. method	p (BS)	λ (BS)	MSE (BS)	p (CV)	λ (CV)	MSE (CV)
OLS	20	-	$1.00 \cdot 10^{-4}$	20	-	$9.58 \cdot 10^{-5}$
Ridge	20	$1.00 \cdot 10^{-6}$	$2.64 \cdot 10^{-4}$	20	$1.00 \cdot 10^{-6}$	$2.70 \cdot 10^{-4}$
Lasso	20	$1.00 \cdot 10^{-6}$	$4.18 \cdot 10^{-4}$	20	$1.00 \cdot 10^{-6}$	$4.11 \cdot 10^{-4}$

5 Conclusion

In this project, we set out to investigate how different methods of linear regression performs on the Franke function, as well as terrain elevation data. To assess our models, we employed Bootstrap and Cross-validation as resampling methods.

For the Franke function, we found that all three methods, OLS, Ridge and Lasso performed very similarly, giving errors on the scale 10^{-3} for the best-fit model. Ridge was however the best model, both in accuracy and stability. Lasso yielded almost as good results as Ridge, but was significantly slower as it has no analytical solution. Cross-validation gave the lowest errors, with roughly 30% improvement over Bootstrapping for the best-fit models.

For the terrain data, we found that the OLS performed the best, with Ridge being a decent bit behind, and Lasso roughly as much behind Ridge. The best fit for OLS was at $p = 20$, which gave an MSE of $9.58 \cdot 10^{-5}$. In this case, Cross-validation was only marginally better for OLS and Lasso regression, and slightly worse for Ridge, but the difference is minor. As all the best fits was the highest p-value we tested, testing higher polynomial degrees would be necessary.

For future work, looking closer at the terrain data would probably be a good idea. Ideally, more parts of the maps should be checked instead of just one small patch, in order to see how more complex and noisy regions affect the results, as well as increasing the degrees that are tested (and the hyperparameter λ). This however would take significantly more time, especially for Lasso, so looking into parallelizing the analysis would be relevant. Since all p and λ computations are independent, doing so should not be too difficult.

References

[1] Richard Franke. A critical comparison of some methods for interpolation of scattered data, 1979.

[2] Earthexplorer. `https://earthexplorer.usgs.gov/`.

[3] Hastie et al. *The Elements of Statistical Learning - Data Mining, Inference, and Prediction.* Springer, second edition edition, 2017.

[4] Wessel N. van Wieringen. Lecture notes on ridge regression, 2020.

[5] Github repository, project 1. `https://github.com/simennb/fysstk4155-project1`.

Appendix

A Data sets

A.1 Franke function

The Franke function as given in [1]

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

where $x, y \in [0,1]$.

A.2 Terrain data

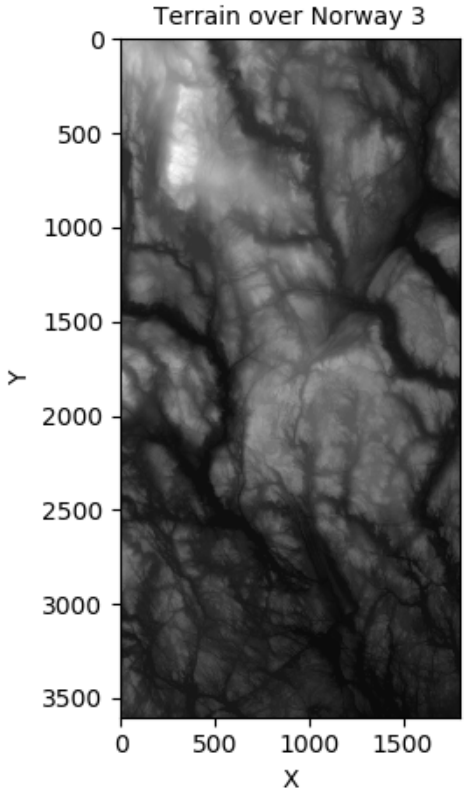
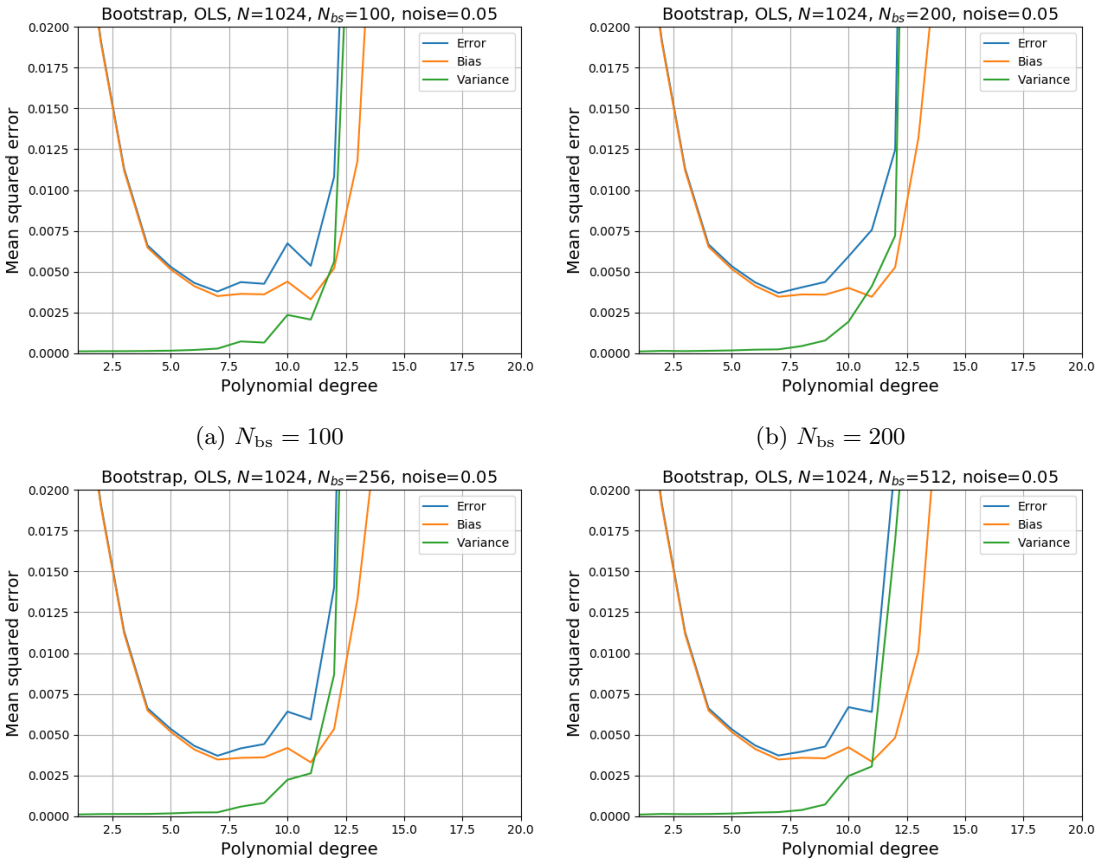


Figure 12: Plot of the entire terrain map in `SRTM_data_Norway_3.tif`. Map is over a region south-east in Norway (Vestfold og Telemark, Viken). Coordinates $(x, y) = (1100, 3000)$ correspond to the location of the city Skien.

B Franke function plots



(a) $N_{\text{bs}} = 100$

(b) $N_{\text{bs}} = 200$

(c) $N_{\text{bs}} = 256$

(d) $N_{\text{bs}} = 512$

Figure 13: Bias-variance plots for OLS where the number of bootstraps N_{bs} is varied. $N = 1024$, $\sigma = 0.05$. In general, all four produce very similar results.

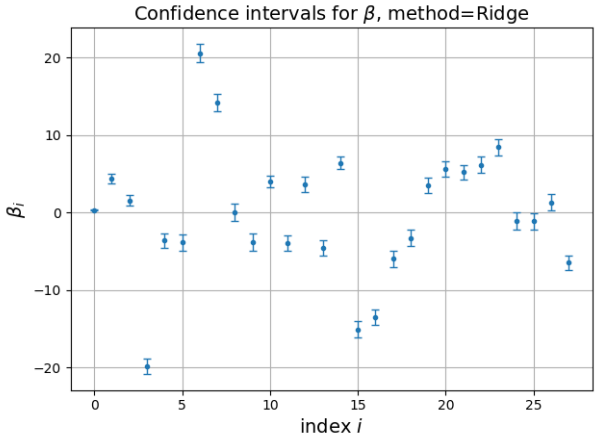
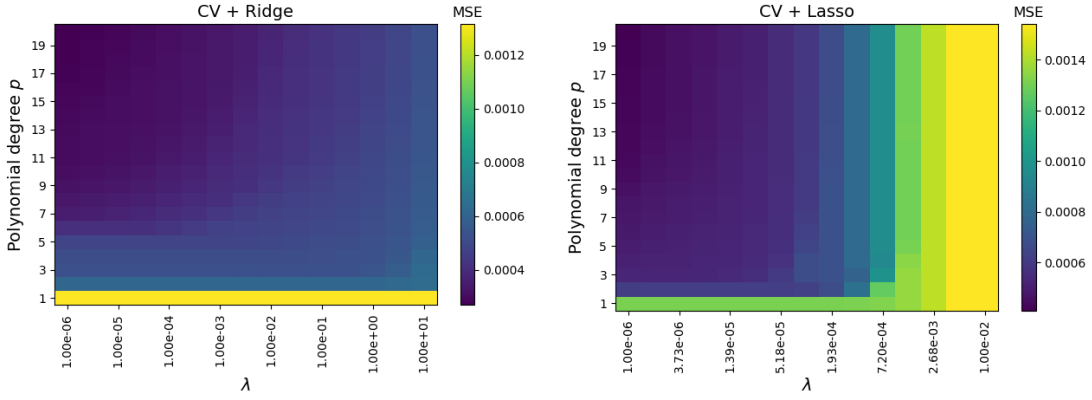


Figure 14: 95% confidence intervals for the regression coefficients β_{Ridge} for $p = 6$, $N = 529$, and noise $\sigma = 0.05$.

C Terrain data plots



(a) Ridge

(b) Lasso

Figure 15: Heat maps showing the MSE plotted against polynomial degree p and the hyperparameter λ for both Ridge and Lasso, using Cross-validation. For Ridge, we calculated the results for 15 log-spaced λs between $1 \cdot 10^{-6}$ and $1 \cdot 10^{1}$, while for Lasso we calculated 15 values of λ between $1 \cdot 10^{-6}$ and $1 \cdot 10^{-2}$. Plots are almost identical to the Bootstrap version found in figure 10.