

Project 1 FYS-STK4155

Taha Mohamed Ali, Simen Stjernen og Diriba T.Dumesso

07. Oktober 2019

Table of contents

1	Introduction	1
2	Assignment A	2
3	Assignment B	4
4	Assignment C	6
5	Assignment D	9
6	Assignment E	13
7	Assignment F	15

References

List of Figures

1	Values of MSE and R^2 score for OLS without resampling	3
2	Train and Test data plot	4
3	Mean square error for OLS with bootstrap resampling (train og test)	5
4	Bias-Variance plot of OLS with bootstrap resampling	9
5	Plot of the coefficients β for different λ	10
6	Plot of the R^2 score for different λ (Ridge)	11
7	Plot of the MSE train and test for different λ (Ridge)	12
8	Plot of the β coefficients for different λ	13
9	Plot of the R^2 score for different λ (Lasso)	14
10	Plot of the SRTM data for 20%	15
11	Plot of the SRTM data for degree 1 and 4	15
12	Plot of the SRTM data for degree 8 and 10	16
13	Plot of the SRTM data for degree 20	16

1 Introduction

In this project we will experiment with several different ways of approximating the Frankfunction. Firstly we will use three different regression methods (OLS, Ridge, Lasso). We will discuss resampling using the Bootstrap method on training and test data, and check how good approximations our models give using the Mean Square Error and R^2 functions. Finally we will use all three on real data, and decide if we prefer to use OLS, Ridge or Lasso to approximate our data.

2 Assignment A

Equation for beta

$$\beta = (X^T X)^{-1} X^T y \quad (1.0)$$

Formel for OLS Formula for OLS

$$y_i = \beta_p x_{ip} + \varepsilon_i, \quad (1.1)$$

On vector form

$$y_i = x_i^T \beta + \varepsilon_i, \quad (1.2)$$

The variance

$$\text{Var}[\hat{\beta} | X] = \sigma^2 (X^T X)^{-1} = \sigma^2 \quad (1.3)$$

OLS is the most basic form of linear regression algorithms we will be working with in this project. We will start by looking at what were after in the equations. In linear regression we want to approximate a set of data to a line with a know function. Ideally we will get a approximation so good that we could use directly as our data. We try to produce this approximation by assembling a design matrix, and from that find coefficients we hope are good.

We do this in our code by defining to arrays x and y. We use these to calculate the Frankefunksjon $z(x,y)$. Then we use them to create our design-matrix X. The beta coefficients are then calculated as stated in formula (1.0). Scikit-learn contains functions for calculating mean square error and R^2 score, we choose to also calculate these our self following the formulas below.

Formula for MSE and R^2

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

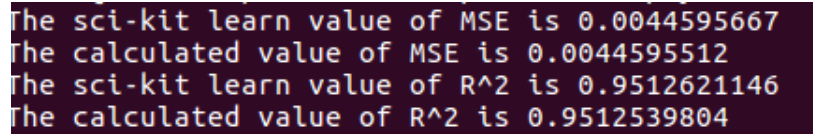
$$R^2(\hat{y}, \tilde{y}) = 1 - \frac{\sum_{i=0}^{n-1} (y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n-1} (y_i - \bar{y})^2},$$

where the mean value of \bar{y} is defined as

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

We would like to have mean square error as close to zero as possible, and the R^2 score as close to 1 as possible. This will give us information about how good our model is at approximating the Frankefunction. Mean square error tells us the size of the mean error between the real value of Frankefunction z and our approximation \tilde{z} . The R^2 score is the percentage of the response variable variation that is explained by our linear model. [4]

Here is an image of the output of the OLS program.



```
The sci-kit learn value of MSE is 0.0044595667
The calculated value of MSE is 0.0044595512
The sci-kit learn value of R^2 is 0.9512621146
The calculated value of R^2 is 0.9512539804
```

Figure 1: Values of MSE and R^2 score for OLS without resampling

3 Assignment B

Resampling is a useful tool in computing and fitting of data. It can be useful to resample data if we for example have a dataset with less data points than we would like. In that instance we can resample the data to bulk out or set of points, and hopefully get a better approximation than we would otherwise. We used the train-test-split functionality from sci-kit learn to split our dataset. We chose to split it in to 80 percent training data and 20 percent test data. We do this to make us able to comment on how good the resampling method is.

We use training data to as the name would suggest to train our model. The test data is used to predict the real values. If training and test data give the same values, or said a different way if calculated value = predicted value we have a perfect model.

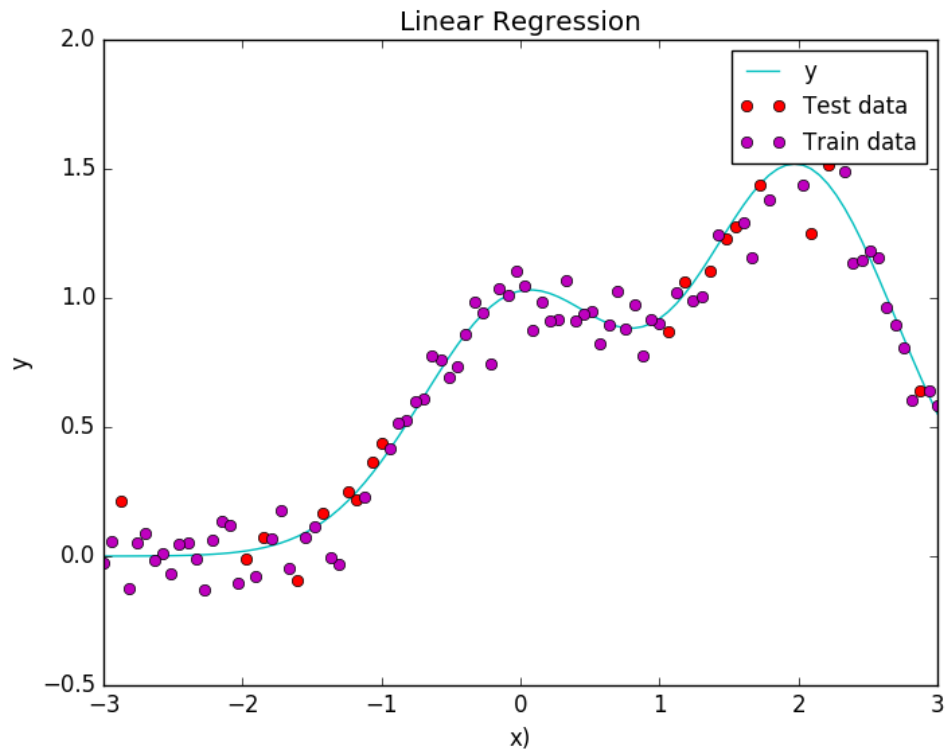


Figure 2: Train and Test data plot

Bootstrapping is a widely used resampling method. It is based on random sampling with replacement. [3] In our case we use it to make our prediction

better. Every time we run a bootstrap our predicted value \hat{z} is given a new value based on a resampling of our training data. We can then find better values for the MSE, by taking the mean of all our bootstraps.

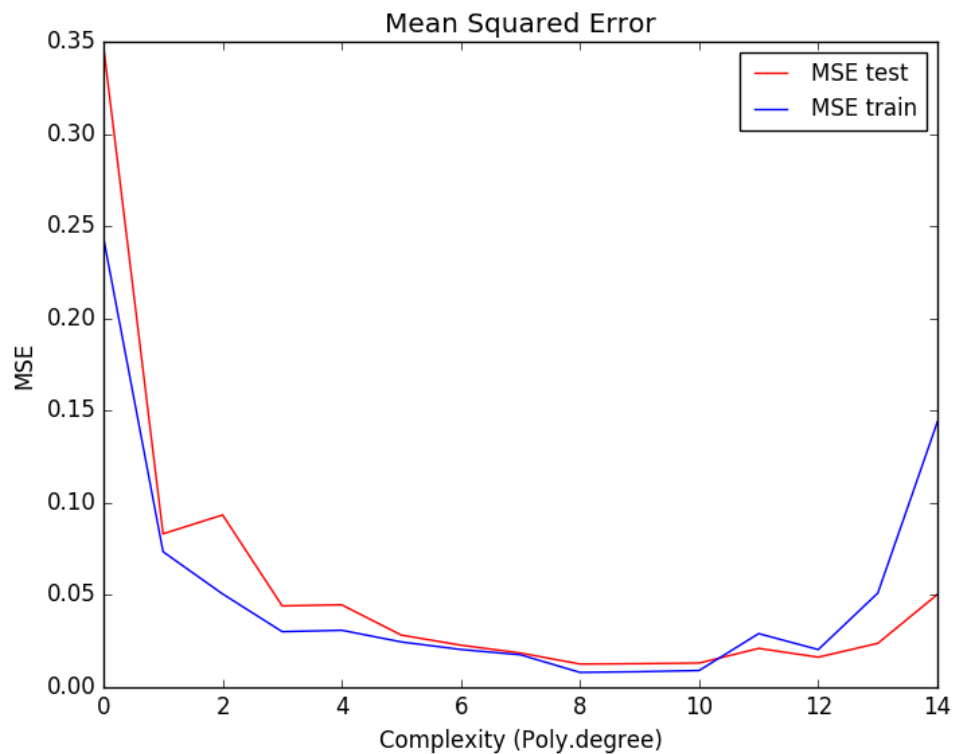


Figure 3: Mean square error for OLS with bootstrap resampling (train og test)

The figure above is a plot of training and test data on a random function with bootstrap resampling. Unfortunately none of our PCs were able to run the bootstrap algorithm on the Frankefunction, so we had to settle for this.

4 Assignment C

In statistics the Bias of a function estimates the difference between the expected value and the real value of the function.[1]

Bias is defined by this formula

$$Bias_{\theta}[\hat{\theta}] = E_{x|\theta}[\hat{\theta}] - \theta = E_{x|\theta}[\hat{\theta} - \theta],$$

Variance is a measurement of the variation in a set of data. It can be split into two types. Theoretical variance which always contains an underlying variation in the probability distribution and empirical variance which is the variation of one selected point from a statistical distribution. [2]

The formula for theoretical variance.

$$\sigma^2 = Var[X] = E[(X - E[X])^2]$$

The formula for empirical variance.

$$s^2 = \hat{\sigma}^2 = \widehat{Var}[X] = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x}_n)^2$$

The proof of

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

It starts by finding a function $\tilde{y}(x)$ which approximates the true function $y(x)$ as close as possible by using some learning algorithm. By calculating the mean square error between y and $\tilde{y}(x)$ we want

$$(y - \hat{y}(x))^2$$

to be minimal for every x

$$x_1, \dots, x_n$$

and for points outside our selection as well. We can of course not hope to do this perfectly, since y_i contains some noise function ε . This means that we need to be prepared to accept some error in any function we construct.

$$E[(y - \hat{y}(x))^2] = \left(\text{Bias}[\hat{y}(x)] \right)^2 + \text{Var}[\hat{y}(x)] + \sigma^2$$

Where the Bias and variance is defined by

$$\text{Bias}[\hat{f}(x)] = E[\hat{f}(x)] - E[f(x)]$$

$$\text{Var}[X] = E[X^2] - \left(E[X] \right)^2.$$

Rewriting the operation

$$E[X^2] = \text{Var}[X] + \left(E[X] \right)^2.$$

Proof of the function

$$E[(y - \hat{y})^2] = E[(f + \varepsilon - \hat{y})^2]$$

$$= E[(f + \varepsilon - \hat{y} + E[\hat{y}] - E[\hat{y}])^2]$$

$$= E[(f - E[\hat{y}])^2] + E[\varepsilon^2] + E[(E[\hat{y}] - \hat{y})^2] + 2E[(f - E[\hat{y}])\varepsilon] + 2E[\varepsilon(E[\hat{y}] - \hat{y})] + 2E[(E[\hat{y}] - \hat{y})(f - E[\hat{y}])]$$

$$= (f - E[\hat{y}])^2 + E[\varepsilon^2] + E[(E[\hat{y}] - \hat{y})^2] + 2(f - E[\hat{y}])E[\varepsilon] + 2E[\varepsilon]E[E[\hat{y}] - \hat{y}] + 2E[E[\hat{y}] - \hat{y}](f - E[\hat{y}])$$

$$= (f - E[\hat{y}])^2 + E[\varepsilon^2] + E[(E[\hat{y}] - \hat{y})^2]$$

$$= (f - E[\hat{y}])^2 + Var[f] + Var[\hat{y}]$$

$$= Bias[\hat{y}]^2 + Var[f] + Var[\hat{y}]$$

$$= Bias[\hat{y}]^2 + \sigma^2 + Var[\hat{y}]$$

As we can see in the plot below, the variance increases at higher complexity and the bias decreases. It looks as we would have expected based on the proof. The expected MSE appears to be the sum of Bias and Variance in every point (with some noise).

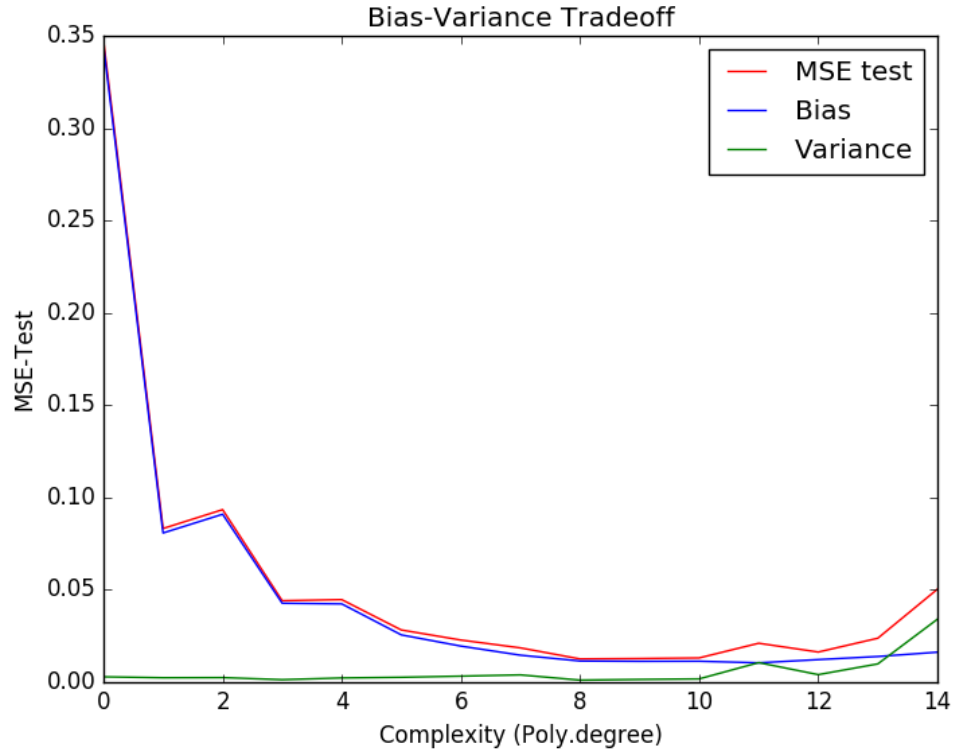


Figure 4: Bias-Variance plot of OLS with bootstrap resampling

5 Assignment D

Ridge Regression

Ridge regression is a type of regularization techniques, which is used to deal with overfitting in the presence of large datasets by adding penalties to the regression function while minimizing the error between predicted and actual observations. The cost function to be minimized is the RSS plus the sum of square of the magnitude of weights. The mathematical representation is:

$$\begin{aligned}
 C(W) &= RSS(\beta) + \lambda * (sum_of_square_of_weights) \\
 &= \sum_{i=1}^n y_i - \sum_{j=0}^m \beta_j x_{ij}^2 + \lambda \sum_{j=0}^m \beta_j^2
 \end{aligned}$$

$$\sum_{j=1}^m (y_i - \hat{y}_i)^2 = \sum \left(y_i - \sum_{j=0}^m \beta_j * x_{ij} \right)^2 + \lambda \sum_{j=0}^m \beta_j^2$$

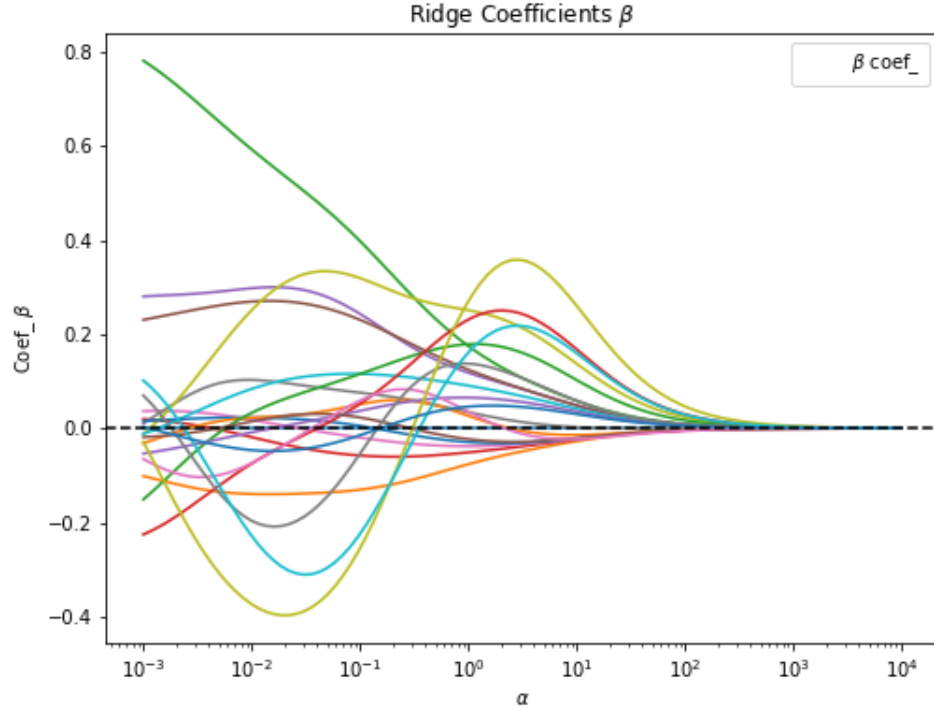


Figure 5: Plot of the coefficients β for different λ

From the polynomial regression, we have seen that as the model complexity increases, the models tends to fit even at smaller deviations in the training data set. Though this leads to overfitting. Thus, the implementation of Ridge regression overcomes this limitation by penalizing the coefficients (β) by the penalty term (λ). It regularizes the coefficients such that if the coefficients take large values the optimization function is penalized.

The degree of freedom λ were computed by using the SVD method.

$$df(\lambda) = \left(\frac{s^2}{s^2 + \lambda} \right)$$

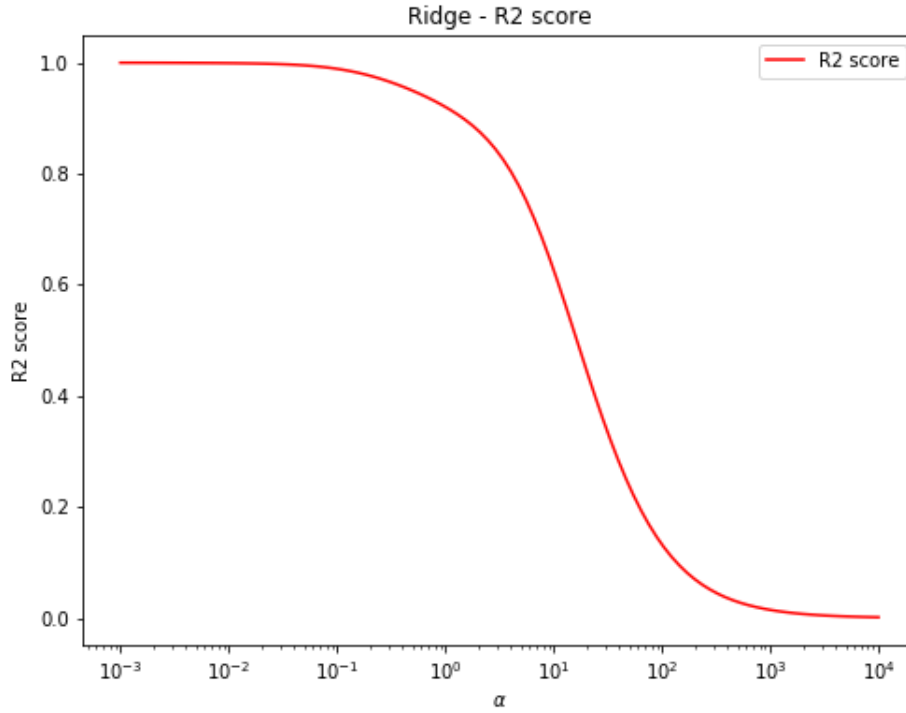


Figure 6: Plot of the R^2 score for different λ (Ridge)

On the other hand, as the value of λ increases, the model complexity reduces. Though higher values of λ reduce overfitting, it can also significantly cause under fitting as well. Thus λ should be chosen wisely by widely accepted technique called cross-validation, i.e. the value of λ is iterated over a range of values.

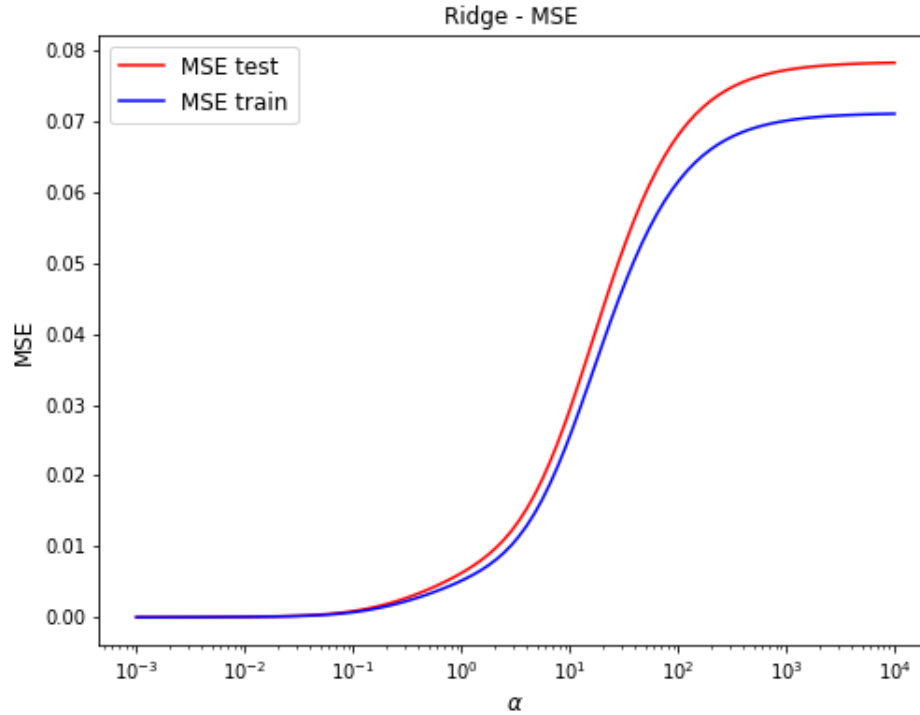


Figure 7: Plot of the MSE train and test for different λ (Ridge)

Observations:

1. The RSS increases with increase in λ , thus reduces the model 1em-complexity
2. Even a very small values of λ gives us significant reduction in magnitude of coefficients.
3. High λ values can lead to significant under fitting.
4. Though the coefficients are very small, they are NOT zero.

One of the main obstacles in using ridge regression is in choosing an appropriate value of λ . A widely accept technique to choose λ is the cross-validation, i.e. the value of λ is iterated over a range of values.

6 Assignment E

Lasso Regression

LASSO, which stands for Least Absolute Shrinkage and Selection Operator, is another type of regression with regularization techniques, penalizing the magnitude of the coefficients of the features by a factor equivalent to sum of absolute value of magnitude coefficients. L1 regularization

$$C(X) = \text{RSS} + \lambda * (\text{sum of absolute value of coefficients})$$

$$C(X, \beta; \lambda) = (X\beta - y)^T (X\beta - y) + \lambda \sqrt{\beta^T \beta}$$

Here, λ works similar to that of ridge and provides a trade-off between balancing RSS and magnitude of coefficients. Like that of ridge, λ can take various values. By iterating, one can observe that:

$\lambda = 0$: Same coefficients as OLS

$\lambda = \infty$: All coefficients shrink ('same' as Ridge)

$0 < \lambda < \infty$: coefficients between 0 and that of OLS

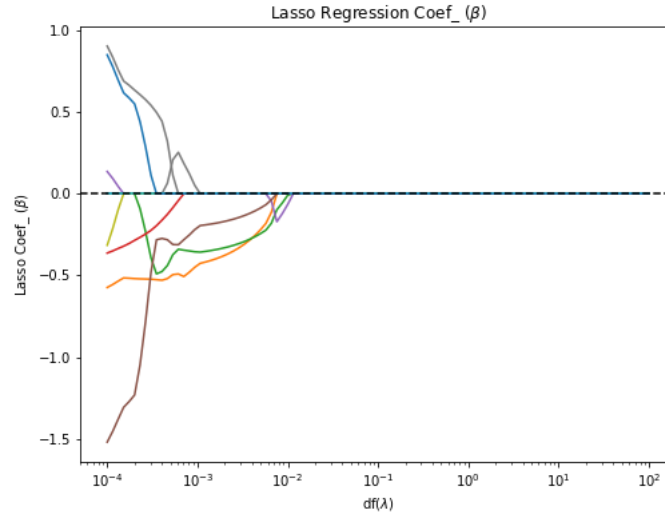


Figure 8: Plot of the β coefficients for different λ

Unless penalized, it is clearly evident that the size of coefficients increase exponentially with increase in model complexity

A large coefficient signify that we're putting a lot of emphasis on that feature, i.e. the particular feature is a good predictor for the outcome. When it becomes too large, the algorithm starts modelling intricate relations to estimate the output and ends up overfitting to the particular training data.

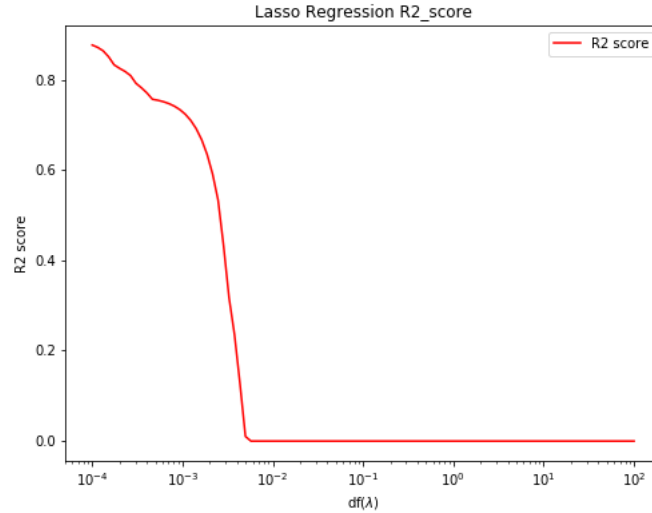


Figure 9: Plot of the R^2 score for different λ (Lasso)

The ridge coefficients are a reduced factor of the simple linear regression coefficients and thus never attain zero values but very small values. The lasso coefficients become zero in a certain range and are reduced by a constant factor, which explains their low magnitude in comparison to ridge.

7 Assignment F

We unfortunately did not get the terrain data to work. As we could not get our program to display the data in any meaningful way.

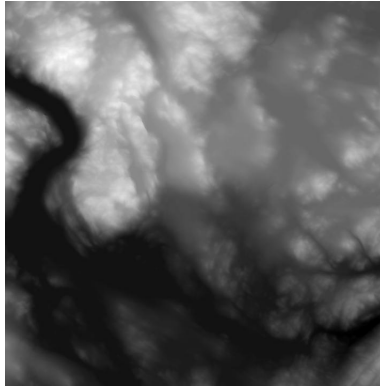


Figure 10: Plot of the SRTM data for 20%

Applying OLS on the sample test data for different degrees, we get the following values:

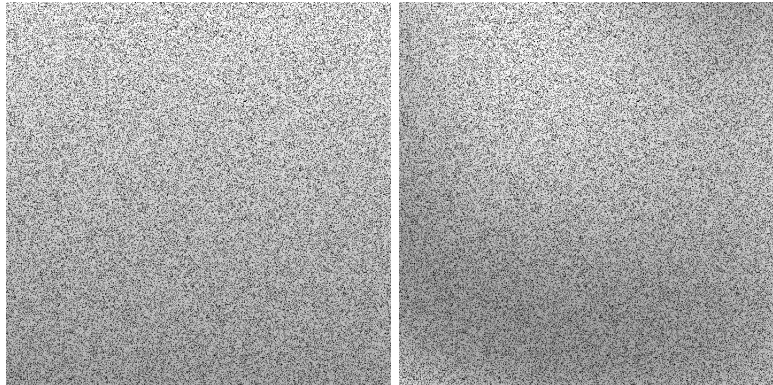


Figure 11: Plot of the SRTM data for degree 1 and 4

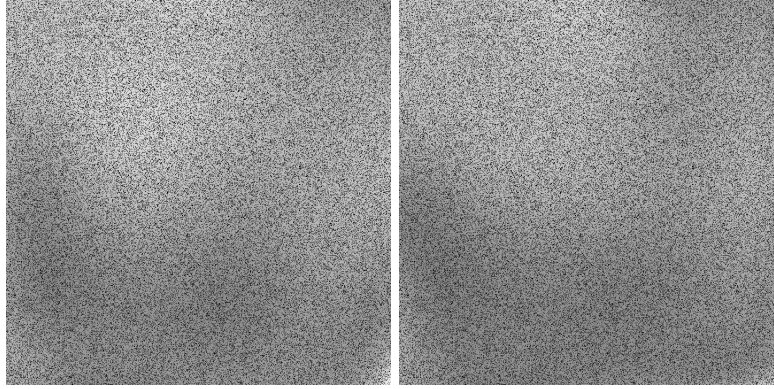


Figure 12: Plot of the SRTM data for degree 8 and 10

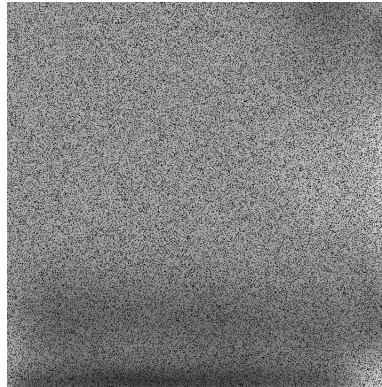


Figure 13: Plot of the SRTM data for degree 20

Due to the problems of extracting the data in to a designmatrix and perform all the three regression methods on the Terrain data.

References

- [1] Bias: Bias of an estimator
https://en.wikipedia.org/wiki/Bias_of_an_estimator
- [2] Varians: formel og definisjon
<https://en.wikipedia.org/wiki/Variance>
- [3] Bootstraps: Bootstrapping(statistics)
[https://en.wikipedia.org/wiki/Bootstrapping_\(statistics\)](https://en.wikipedia.org/wiki/Bootstrapping_(statistics))
- [4] R^2 score: Regression Analysis: How Do I Interpret R-squared and Assess the Goodness-of-Fit?
<https://blog.minitab.com/blog/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit>