FYS-STK4155 – Applied data analysis and machine learning
Project 1 - Regression analysis and resampling methods

Jon A Ottesen

(Dated: 9. oktober 2019)

The main topic of this projects is to study different regression methods including Ordinary Least Squares, Ridge regression and Lasso regression. These methods in conjunction with resampling techniques such as k-fold cross-validation are used in polynomial fitting on the 2-dimensional Franke function surface with added Gaussian noise and terrain data. The different methods give varied result in reproducing the original surface on a test section with Ridge giving the smallest $MSE = 0.0075$ with respect to the Franke function when predicting data, whereas lasso gave the worst estimate with $MSE = 0.0087$. Finally real-life data is the analyzed using the same techniques as the Franke function with OLS giving the best error estimates with $R^2 = 0.8657$ for 15^{th} degree polynomial over ridge, with lasso preforming worst.

I. INTRODUCTION

The first paper on regression methods was published by Legendre in 1805 and Gauss in 1809 about the least square method[5]. Albeit being a relatively simple theory developed before the computational era of statistics its importance cannot be overstated, and is still prominent and even outperforming newer more complex methods in some cases. The simple nature of the least square theory makes it an excellent starting point for further studies in regression theory while still giving decent results in real-life data-analysis. Especially when applied together with resampling methods which has a vital role in modern statistical data analysis.

Regression methods are often used in conjunction with measured data to determine the underlying patterns. Unlike in most research where linear regression is used as a tool for data-analysis, it will here be the main object of study itself. To stimulate a more real-life usage of linear regression actual data[1] is used together with generated data from Franke's function. The central themes are thus resampling methods, error analysis such as the mean squared error, the bias-variance tradeoff and most importantly the main linear regression methods themselves: OLS, rigde and lasso regression.

[1] https://github.com/CompPhysics/MachineLearning/tree/master/doc/Projects/2019/Project1/DataFiles

II. THEORY

As a basis for most of the theory below we will assume the data follows this form:

$$\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\epsilon} \tag{II.1}$$

where $\boldsymbol{y}$ is the data-sample, f is the function describing the data and $\boldsymbol{\epsilon}$ is the noise of the data following a Gaussian distribution with a mean of 0.

A. Moments and error analysis

1. *Moments in statistics*

To describe, analyse and understand data, statistical knowledge is essential. Various moments in statistics are therefore used when describing key elements of data, models or functions. In our case we want to describe and understand the model created with linear regression, and for this the 1^{st} and 2^{and} order moments are a necessity.

For a real valued continuous function $g(x)$ for real-valued x, the n-th moment is given by

$$\mu_n(c) = \int_{-\infty}^{\infty} (x-c)^n g(x) dx \tag{II.2}$$

with c as a real variable. By restricting g as a normalized non-negative function the 1st moment around $c = 0$ is the mean value given by

$$\mu = \int_{-\infty}^{\infty} x g(x) dx. \tag{II.3}$$

Higher order moments are often defined around the mean:

$$\mu_n(\mu) = \int_{-\infty}^{\infty} (x-\mu) g(x) dx \tag{II.4}$$

to provide more qualitative information about the distribution. The 2nd moment is the variance

$$\mu_2(\mu) = \sigma^2 = \int_{-\infty}^{\infty} (x-\mu)^2 g(x) dx \tag{II.5}$$

where σ is the standard-deviation.

So far I have only looked at the continuous case where the probability distribution is know. In the non continuous case with a sample of finite size the true moments of the distribution can realistically only be approximated. There is however theoretically possible to get the correct values by making use of the central limit theorem.

For a finite sample of size n taken from the function g the 1st moment i.e the mean value is for the sample is given by:

$$\overline{g^s} = \frac{1}{n} \sum_{i=1}^{n} g_i^s \tag{II.6}$$

The 2an moment i.e variance is for the sample

$$\mathrm{Var}(g^s) = \frac{1}{n} \sum_{i=1}^{n} (g_i^s - \overline{g^s})^2 \tag{II.7}$$

where in both cases g^s is a sample of data from the distribution $g(x)$. Note however that $\overline{g^s}$ and $\mathrm{Var}(g^s)$ are only approximations to the real mean μ and variance μ_2 for the distribution g.

Let's assume we are finite sample of size n $\boldsymbol{y}$ which obeys function II.1, where the function $\hat{\boldsymbol{f}}$ is the approximation of $f(\boldsymbol{x})$ by regression methods. The variance of the noise $\boldsymbol{\epsilon}$ can be approximated by

$$\mathrm{Var}(\epsilon) \approx \frac{1}{n-m} \sum_{i=1}^{n} (y_i - \hat{g}_i)^2 \tag{II.8}$$

where $m = p-1$ and p is the number of columns in the design matrix $\boldsymbol{X}$([1] page 47).

2. *Error analysis*

There are a multitude of different cost functions for evaluating the error in a model. Among them are the mean square error(MSE) and the R^2 score function which are respectivly given by

$$MSE(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2 \tag{II.9}$$

$$R^2(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = 1 - \frac{\sum_{i=1}^{n} (y_i - \tilde{y}_i)^2}{\sum_{i=1}^{n} (y_i - \overline{y})^2}. \tag{II.10}$$

Notice how for $m = 0$ in equation II.7 the we are left with the equation for the MSE.

B. Linear regression methods

The very basis of linear regression is based around the assumption that the form of the data can be written on the form of equation II.1. This form can again be written as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{II.11}$$

where $\boldsymbol{y}, \boldsymbol{\epsilon} \in \mathbb{R}^{n \times 1}$ whereas $\boldsymbol{\beta} \in \mathbb{R}^{p \times 1}$ and $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ is the design matrix.

The goal of linear regression is thus to approximate $\boldsymbol{y}$ with

$$\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta} \tag{II.12}$$

by optimizing $\boldsymbol{\beta}$.

1. Ordinary least squares

The ordinary least square method optimizes $\boldsymbol{\beta}$ by minimizing the MSE cost function

$$C(\boldsymbol{X},\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i) \tag{II.13}$$

$$= \frac{1}{n}\left[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^T(\boldsymbol{y}-\tilde{\boldsymbol{y}})\right] \tag{II.14}$$

$$= \frac{1}{n}\left[(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})\right]. \tag{II.15}$$

The steps is to take following derivative

$$\frac{\partial C(\boldsymbol{X},\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0 \tag{II.16}$$

which result in the OLS formula

$$\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{II.17}$$

A more rigorous version of the same derivation is found in [2].

Another way of deriving the OLS formula is by considering

$$\tilde{\boldsymbol{y}} = \text{proj}_{\text{Col X}}\boldsymbol{y}. \tag{II.18}$$

Following this there exist a $\boldsymbol{\beta}$ such that

$$\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}. \tag{II.19}$$

A consequence of having $\tilde{\boldsymbol{y}} = \text{proj}_{\text{Col X}}\boldsymbol{y}$ is that $\boldsymbol{y}-\tilde{\boldsymbol{y}}$ is orthogonal to the vector space spanned by $\boldsymbol{X}$ as stated by the Orthogonal Decomposition Theorem. Therefore any column in $\boldsymbol{X}$ must be orthogonal to $\boldsymbol{y}-\tilde{\boldsymbol{y}}$ such that

$$\boldsymbol{X}_i \cdot (\boldsymbol{y}-\tilde{\boldsymbol{y}}) = 0. \tag{II.20}$$

This implies

$$\boldsymbol{X}^T(\boldsymbol{y}-\tilde{\boldsymbol{y}}) = 0 \tag{II.21}$$

and finally

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{X}^T\tilde{\boldsymbol{y}} \tag{II.22}$$

which can be rearranged to equation II.17. The derivation is taken from [3] in chapter 6.5.

Both derivations are equal in the sense that they yield the OLS formula, but their interpretations are different.

2. Ridge and Lasso regression

Ridge and lasso regression are two shrinkage methods in linear regression. They work by imposing a penalty on the regression coefficients based on their size. This is done by minimizing their respective cost functions, and for ridge regression this cost function is given by

$$C(\boldsymbol{X},\boldsymbol{\beta})_{\text{Ridge}} = \frac{1}{n}\left[(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}\right] \tag{II.23}$$

which is variant of the MSE with $\lambda > 0$. The coefficients for $\boldsymbol{\beta}$ in ridge regression can be written in the following closed form

$$\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{II.24}$$

where $\boldsymbol{I} \in \mathbb{R}^{p\times p}$. Further analysis of equation II.24 would reveal how the λ parameter shrinks terms i beta, but that is not necessary for this project.

Lasso regression much like OLS and ridge regression is a minimization problem. However unlike OLS and ridge regression there exist no analytical solution to the cost function minimized in lasso regression:

$$C(\boldsymbol{X},\boldsymbol{\beta}) = \frac{1}{n}||\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1. \tag{II.25}$$

To solve lasso regression the implementation of gradient decent types of algorithms are a necessity. Such algorithms are outside the scope of this project and will not be covered in any depth, instead preexisting tools will be utilized.

3. Confidence intervals

The confidence intervals for the β_i terms from equation II.17 at 68% confidence intervals for OLS is given by:

$$\sigma(\beta_i)^{\text{OLS}} = \sigma\sqrt{(\boldsymbol{X}^T\boldsymbol{X})_{ii}}. \tag{II.26}$$

Thus the variance of $\boldsymbol{\beta}^{\text{OLS}}$ is given by the diagonal of the square matrix $\boldsymbol{X}^T\boldsymbol{X}$ multiplied with

the variance of the error in function II.1. For ridge regression the confidence intervals of the β_i terms from equation II.24 at 68% is given by

$$\sigma(\beta_i)^{\text{Ridge}} = \sigma\sqrt{\left[\left(\boldsymbol{X^T X}+\lambda \boldsymbol{I}\right)^{-1}\boldsymbol{X^T X}\left(\left(\boldsymbol{X^T X}+\lambda \boldsymbol{I}\right)^{-1}\right)^T\right]_{ii}}.$$

4. SVD

Numerical matrix inversion has a tendency to become unstable for larger matrices $\boldsymbol{X}$. To avoid this problem, SVD will be utilized to provide a more stable algorithm. Without going into to much technicality the goal of this subsection is to formulate mainly equation II.17 and II.24 but also their confidence intervals in terms of $\boldsymbol{U}, \boldsymbol{V}$ and $\boldsymbol{\Sigma}$.

Any $n \times p$ matrix $\boldsymbol{X}$ with rank r can be written as

$$\boldsymbol{X} = \boldsymbol{U\Sigma V}^T \tag{II.27}$$

with $\boldsymbol{U}$ being an orthogonal $n \times n$ matrix and $\boldsymbol{V}$ an orthogonal $p \times p$ matrix. Whereas $\boldsymbol{\Sigma}$ is a $n \times p$ matrix where the first r diagonal elements are the singular values of $\boldsymbol{X}$[2]. A small note is that $\boldsymbol{V}^{-1} = \boldsymbol{V}^T$ and $\boldsymbol{U}^{-1} = \boldsymbol{U}^T$ since they are orthogonal matrices.

The formulas for both OLS and ridge regression can be rewritten in terms of $\boldsymbol{\Sigma}, \boldsymbol{U}, \boldsymbol{V}$ as

$$\boldsymbol{\beta}_{OLS} = \boldsymbol{V}(\boldsymbol{\Sigma})^{-1}\boldsymbol{U}^T\boldsymbol{y} \tag{II.28}$$

$$\boldsymbol{\beta}_{Ridge} = \boldsymbol{V}(\boldsymbol{\Sigma}^2 + \lambda \boldsymbol{I})^{-1}\boldsymbol{\Sigma}^T\boldsymbol{U}^T\boldsymbol{y}. \tag{II.29}$$

Both derivations can be found at [4]. Further the confidence intervals for the β_i from equation II.26 can be rewritten as

$$\sigma(\beta_i)^{\text{OLS}} = \sqrt{\left(\boldsymbol{V}\left(\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}\right)^T\boldsymbol{V}^T\right)_{ii}}\sigma \tag{II.30}$$

while the confidence interval for Ridge regression takes the following form:

$$\sigma(\beta_i)^{\text{Ridge}} = \sigma\sqrt{\left(\boldsymbol{V}\left(\boldsymbol{\Sigma^2}+\boldsymbol{\lambda}\right)^{-2}\boldsymbol{\Sigma^2 V^T}\right)_{ii}} \tag{II.31}$$

where $\boldsymbol{\lambda} = \lambda \boldsymbol{I}$. The derivation of equation II.30 and II.31 is done in a similar manner to equation II.28 and II.29 which is derived in [4].

C. Resampling and Bias-variance tradeoff

1. Reasmpling methods

There exist a multitude of different resampling methods, but the common trope is repeatedly refitting a model by drawing(often randomly) samples from a larger data set. By repeatedly drawing out samples from a larger set the goal is to utilize the central limit theorem and limit the correct value for a large number of runs, but also compare models between different sets of the same data.

K-fold cross validation is based around the principle of dividing a data set into k equally sized(if possible) folds with $k \leq$ (length of the data). Than k-1 of the folds are used as training data in linear regression while one fold is used as test data. Thereafter the prediction error and mean among other things are evaluated. This is redone k times but with a different fold for the test data each time such that all k-folds are used as test data. By the central limit theorem the evaluated values are than decent approximations to the real value. Another important aspect would be to see how the results differ. Albeit not essential the data set should be shuffled before diving into folds to avoid an unbalanced representation of the data set.

2. Bias-variance tradeoff

When creating a model an important application is the ability to predict using the model. The term overfitting is often used when describing the action of fitting a model to close to the data such that the model no longer describes the underling function $f(\boldsymbol{x})$. The bias-variance tradeoff is closely related to overfitting and underfitting data. In such a way that our model

must be complex enough to have low bias error while avoiding large variance errors.

The MSE from equation II.9 can be rewritten as

$$MSE(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \frac{1}{n}\sum_i (f_i - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2 + \frac{1}{n}\sum_i (\tilde{y}_i - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2 + \sigma^2. \tag{II.32}$$

In equation II.32 the first term is the bias squared term whereas the second is the variance term, and the last is the variance of the inevitable error from equation II.1. The derivation is given in section C in the appendix.

The bias error is the error which reveals the difference between the model and the actual function. For large bias errors the model has a large deviation from underlying function f. High bias can cause the model to miss important relations in the data. The bias error is large in underfitting.

The variance error is the error that reveals how sensitive the model is to the noise. For high variances the model is not only fitted on the function f, but also on the error in the data set. If the noise were changed the model would change drastically. This error is most important when overfitting.

The main goal of a model is to find the ideal combination of bias and variance error such that the error is minimized, while also being able to predict important features of the underlying function f.

III. METHOD

All code can be found in my github at https://github.com/JonOttesen/FYS-STK4155/tree/master/Project1 and `regression.py` is the general module for the regression analysis, `results.py` is the program producing the results for the Franke function and `terrain.py` is the file which produces the results for the terrain data. The renaming python files `latex_print.py` is used for printing latex tables wheras `testing.py` is a testing file and is not used in producing results.

A. Preparations

There are two sets of data used during this project. The first is self-generated through the 2D Franke function with added noise normally distributed, while the other set is terrain data found at https://github.com/CompPhysics/MachineLearning/tree/master/doc/Projects/2019/Project1/DataFiles and the datafile used is the `SRTM_data_Norway_2.tif` file.

The self-generated data is generated with the Franke function on a $n \times n$[2] meshgrid with $x, y \in [0, 1]$ selected by random from a uniform distribution, and each point has a added normally distributed noise with $\mu = 0$ and $\sigma = 1$ in the z-direction. Thus the function used in the data-generation is

$$f_F(x, y) = F(x, y) + \mathcal{N}(0, 1) \tag{III.1}$$

with $F(x, y)$ being the Franke function. Note that I used the *seed 42* when generating all the random numbers, this was to simulate a real-life data set where the measured data is static.

In the terrain data I was given all the data points, thus I only had to generate the grid. Before doing that I cut away the last row and column such that my data set was of 3600×1800 data points. I further decrease the size of the data to 400×200 by taking the average value of 9×9 non overlapping grids inside the data set.

With a reduced data set I created the x and y grid by creating a grid of 400 linearly spaced

[2] It's not requirement for grid to be $n \times n$, however making the grid $n \times m$ with $m \neq n$ would not have any noticable effect on the results for a reasonable sized n and m.

x-values and 200 linearly spaced y-values with $x, y \in [0, 1]$.

The last preparatory step before the analysis is to create a design matrix from the x,y datapoints in the meshgrid. This is done by flattening the meshgrids using `np.ravel` and the design matrix is created by polynomial combinations of x and y up to a given degree n. A code snippet of the creation of X is shown here:

```
for i in range(1, k + 1):
    q = int((i)*(i + 1)/2)
    for k in range(i + 1):
        X[:,q+k] = x**(i-k) * y**k
```

For a showing of the order of the different polynomial print `polynomial_str` parameter from the `regression` class.

For all design matrices the SVD is calculated by scipy and `full_matrices = False`.

With the SVD calculated the regression method OLS is calculated by equation II.28 while ridge is calculated by equation II.29. Lasso is calculated by sklearn by `sklearn.linear_model.Lasso`.

B. Regression analysis

For future reference when talking about the ideal λ-value I am referring to the λ which minimized the MSE and maximized the R^2 error in k-fold cross validation for the excluded fold. This is done by calculating the error estimates for the excluded fold for m-different λ-values which remain constant throughout the k-fold. Than the λ which minimized the mean error in the excluded folds is the 'ideal' λ. In table I I have tried to illustrate how the error estimate may differ for the same fold depending on the λ-value. In table I the ideal λ would be number 4 since mean is the smallest.

λ-values	Fold 1	Fold 2	Fold 3
λ_1	1	2	2
λ_2	2	2	2
λ_3	1	2	1
λ_4	1	1.5	1

Tabell I: A illustration of how the error estimate can differ depending on the fold and the λ value. The ideal λ would here be λ_4.

1. *Franke function*

Most of the actual data analysis is pretty similar between the different data sets. In the Franke function data set the first step was to calculate the OLS, ridge and lasso models for the entire design matrix with 5^{th} degree polynomial complexity. The resulting models was than used to calculate the MSE and R^2 error by equation II.9 and II.10 respectively. The error estimates was calculated both with respect to the data set and to the real Franke function without noise. How the λ values used for ridge lasso was found is explained in the next couple of paragraphs. The confidence intervals for β was also calculated by equation II.30 for OLS and equation II.31 for ridge regression. In lack of a proper way to calculate the confidence interval for lasso, equation II.31 was used.

The next step was to split the data sets into training and test data using the `test_train_split` function from sklear. The ratio for the split used was 70% training data and 30% test data. The training data was than used to fit linear regression models for OLS, ridge and lasso with the MSE and R^2 errors calculated using the test data and the test section of the Franke function without noise. Further for ridge and lasso regression the MSE and R^2 error estimates was calculated for a multitude of different λ-values, and the λ giving the minimum was found. This λ is the same used when fitting the entire data set in the previous paragraph. The confidence intervals for β in OLS was also calculated by equation II.30, the same was done for ridge and lasso, but with equation II.31 and their respec-

tive minimum λ-values.

Now the resampling technique k-fold cross validation was implemented. The β-coefficients, MSE and R^2 values were stored for each exclusion of a fold. The error quantities were calculated both with respect to the test data and training data, and stored. Finally the mean and variance for each measured quantity was calculated. This was done for $k = 10$ folds.

The next step was to use k-fold cross validation with 5-folds to calculate the MSE and R^2 for test and train data. This was done for different polynomial degrees from 0 to 15. This was repeated 50 times for each degree but with random folds and the mean of the 50 runs for each polynomial degree was plotted. The result is thus a plot of the training error and test error pr polynomial degree. This was also done for a larger data set with 400×200 data points.

The next step is to create a plot of the bias and variance. This is done by recreating the data set 100 times pr polynomial degree and creating a model each time for the training data. With 100 different models the variance in the bias-variance tradeoff was calculated by taking the variance between the $\tilde{\boldsymbol{z}}$ in the test data. The bias was than calculated by taking the mean squared difference between the Franke function and the mean of all the $\tilde{\boldsymbol{z}}$-models for the test set as shown i equation II.32.

In the last part the ideal λ-value pr complexity is used to create models for the training data in the splitted data set. This is done for polynomials from 3 to 15 degree complexity for OLS, ridge and lasso regression. The error estimates is than calculated between the model and the test data, but also between the model and the test section of the real Franke function. These models are also used to calculate the error estimates for the entire data set and finally the entire data set is used in the creation of these models with the ideal polynomial degree and λ.

The last part which was done was to plot the models surface in 3D for both a OLS model and ridge model with the ideal λ.

2. *Terrain data*

Unlike for the Franke data I will begin by calculating the error estimates using k-fold cross validation with 5 folds. This is done for all polynomial orders from 1 to 20 and repeated $N = 10$ times for each polynomial with random folds. In the k-fold the error estimates for both the test and training folds are calculated and in the end both are plotted side by side. The plotted values are the mean of N times.

After k-fold cross validation the data set is splitted into test and training data. The training data is used to create multiple models for ridge and lasso for varying λ values. For each model with different λ error estimates are calculated on test data. The error estimates in test data are than plotted as a function λ. This is done for 14, 15 and 16^{th} degree complexity.

In the next step the evolution of the ideal λ-values are studied. This is done by calculating the ideal λ by k-fold a total of 50 times with randomized folds, than the mean λ-value is calculated with corresponding standard error at 95% confidence. This is repeated for all polynomial degrees between 1^{st} and 15^{th} complexity. Finally the mean λ-values for each complexity is plotted with the error estimates, only the non-zero λ are plotted.

With the mean ideal λ-values calculated, k-fold cross validation is again used to calculate the error estimates for test and training. Ridge regression with the the mean ideal λ-values are than used to create models for all polynomial orders from 1 to 20. The ridge model with the mean ideal λ-value for each complexity is than used to calculate the error estimates for test and training sets. The error estimates from the test and training are than plotted in the same plot. For the polynomial degrees where there were no mean ideal λ-value I used $\lambda = 0$. As earlier the k-fold was done $N = 10$ times with random folds and the average error estimates for test and training was plotted.

In the next part k-fold cross validation was again used but now with $k = 10$ folds. The error

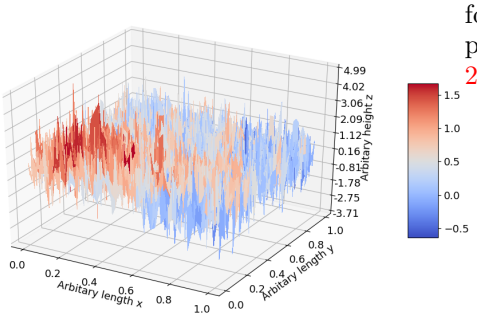
estimates for the OLS, ridge and lasso models for a 15^{th} degree complexity was than plotted in a histogram. The inevitable error from equation II.1 was also calculated, this by equation II.7.

The final part was simply creating a model for both 15^{th} and 5^{th} degree polynomial complexity for the entire data set. Calculating the error estimates and finally plotting colormesh plots for both modeled surfaces. Ridge and lasso regression was not used in the final part.

IV. RESULTS

1. *Franke function*

For most of the analysis, the data set will consist of 81×81 data points except at the end of this subsection when stated otherwise. A plot of the data set of the Franke function with added noise is shown in figure 1. This is the data that will be used for further analysis.



Figur 1: The Franke function with added Gaussin noise with $\mu = 0$ and $\sigma = 1$ for a grid of 81×81 randomly distributed data points taken from a uniform distribution between $[0, 1]$.

The model created by OLS, Ridge and Lasso with a complexity of 5^{th} degree gave the following β-coefficients in table II for the full data set. Note however my choice for λ for both Ridge and Lasso, this choice will be apparent later. The confidence intervals for OLS is caluclated by II.30 while ridge and lasso is calculated by equation II.31, and all σ are multiplied with 1.96 to get the confidence at 95%.

The linear models with the β-coefficients shown in table II gives the following error estimates for MSE and R^2 shown in table III calculated by equation II.9 and II.10. The error estimates shown are both between the model and the data set, but also between the actual Franke function and the model.

So far all the results presented are from creating models on the entire data set. From now on the data set is split at a ratio 70% training and 30% test data. The β-coefficients from the training data is given in table XIV in the appendix. The error estimates are given in table IV and V. In table IV the error estimates is calculated based on the data points used for training while the error estimates in table V is calculated based upon the test section of the data.

So far I have only blindly used two values for λ for both ridge and lasso regression. In figure 2 and 3 the MSE between the test set and a model for varying λ with ridge and lasso regression is plotted respectively. The same applies to figure 24 and 25, but for a wider range of λ-values.

β	OLS	Ridge	Lasso
β_0	0.375 ± 0.401	0.623 ± 0.241	1.1 ± 0.379
β_1	8.04 ± 4.6	4.22 ± 1.7	-0.211 ± 4.23
β_2	4.28 ± 4.52	3.23 ± 1.65	0.537 ± 4.15
β_3	-32.0 ± 22.5	-15.6 ± 4.74	-2.43 ± 20.3
β_4	-10.3 ± 17.1	-1.22 ± 4.17	1.27 ± 15.9
β_5	-13.9 ± 22.9	-12.4 ± 4.74	-4.18 ± 20.6
β_6	35.9 ± 51.3	9.72 ± 6.16	1.3 ± 46.3
β_7	40.2 ± 37.4	10.1 ± 6.84	3.14 ± 34.8
β_8	4.59 ± 36.3	-4.09 ± 6.85	-2.39 ± 34.0
β_9	4.93 ± 52.8	7.01 ± 6.09	1.44 ± 47.4
β_{10}	-6.6 ± 54.3	9.03 ± 6.99	0.325 ± 49.2
β_{11}	-50.2 ± 40.7	-7.62 ± 9.15	0.0 ± 38.3
β_{12}	8.07 ± 37.2	9.88 ± 9.51	0.0 ± 35.1
β_{13}	-19.1 ± 39.4	-5.49 ± 9.28	0.0 ± 37.3
β_{14}	18.2 ± 55.9	10.4 ± 6.82	1.71 ± 50.4
β_{15}	-5.77 ± 21.5	-8.06 ± 4.26	-0.0937 ± 19.6
β_{16}	17.3 ± 18.3	-0.983 ± 6.79	-1.76 ± 17.6
β_{17}	4.8 ± 17.6	-0.287 ± 8.56	0.0 ± 17.0
β_{18}	-10.5 ± 17.6	-6.5 ± 8.66	0.189 ± 17.0
β_{19}	14.8 ± 17.8	6.05 ± 6.81	0.0 ± 17.1
β_{20}	-13.6 ± 22.1	-8.49 ± 4.26	-0.0319 ± 20.0

Tabell II: The β-values for the OLS, Ridge and Lasso regression methods with their respective confidence intervals. For Ridge $\lambda = 4.95 \cdot 10^{-3}$ while for Lasso $\lambda = 6.34 \cdot 10^{-5}$.

Error estimates	OLS	Ridge	Lasso
MSE Franke	0.0053	0.0062	0.0102
MSE Data	0.998	0.999	1.01
R^2 Franke	0.944	0.934	0.892
R^2 Data	0.102	0.101	0.0937

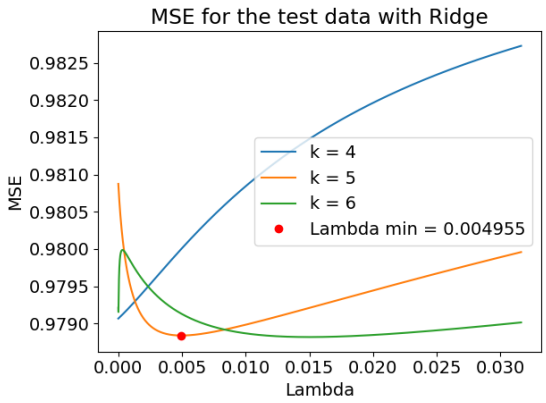
Tabell III: The error estimates MSE and R^2 calculated based upon the model created by the coefficients in table I. The error estimates are calculated both with respect to the actual Franke function and the data set used to create the model.

Error estimates	OLS	Ridge	Lasso
MSE Franke	0.00783	0.00807	0.011
MSE Data	1.01	1.01	1.02
R^2 Franke	0.917	0.915	0.884
R^2 Data	0.0994	0.0979	0.0914

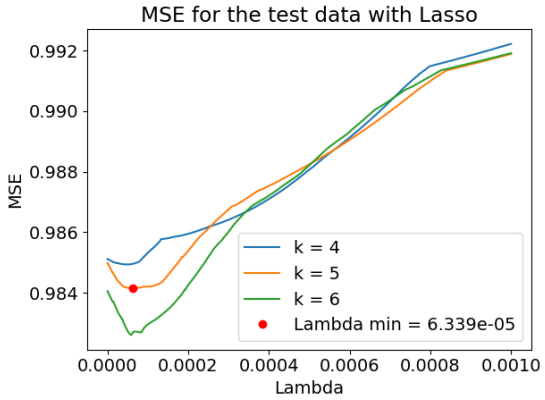
Tabell IV: The error estimates MSE and R^2 calculated based upon the model created by the coefficients in table XIV. The error estimates are for the training section of the data and are calculated both with respect to the actual Franke function and the data set used to create the model.

Error estimates	OLS	Ridge	Lasso
MSE Franke	0.00783	0.0079	0.0106
MSE Data	0.981	0.979	0.984
R^2 Franke	0.917	0.916	0.888
R^2 Data	0.101	0.103	0.098

Tabell V: The error estimates MSE and R^2 calculated based upon the model created by the coefficients in table XIV. The error estimates are for the test section of the data and are calculated both with respect to the actual Franke function and the test section of the data set used to create the model.



Figur 2: The MSE between the test set and the predicted model using Ridge regression for different polynomial orders and λ values. The λ which gives the minimum MSE for a 5^{th} order polynomial is highlighted.



Figur 3: The MSE between the test set and the predicted model using Lasso regression for different polynomial orders and λ values. The λ which gives the minimum MSE for a 5^{th} order polynomial is highlighted. The model is calculated using 1001 iterations pr λ-value.

Using the resampling method k-fold cross validation with 10-folds the MSE calculated between the model and the excluded fold is shown as histograms in figure 4, 5 and 6 for OLS, ridge and lasso respectively.

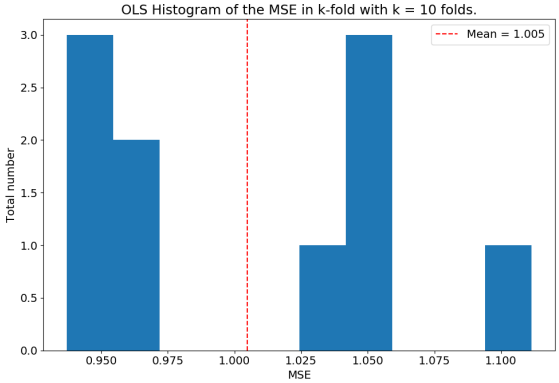
For 10-folds with OLS the MSE and R^2 scores from the test data in k-fold are

$$\begin{aligned} \text{MSE} &= 1.005 \\ R^2 &= 0.0937. \end{aligned}$$

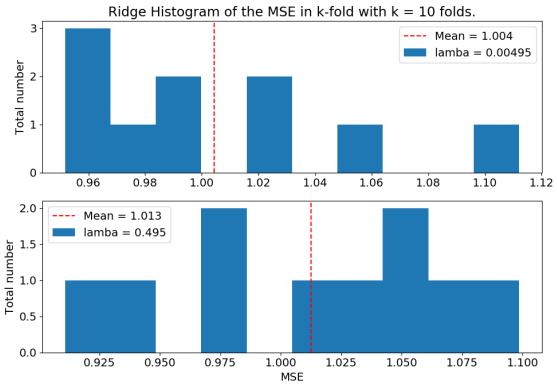
The standard deviation of the inevitable error from equation II.1 is calculated for each exclusion of a fold in k-fold cross validation. The mean of all these standard deviations is

$$\sigma = 1.0007 \pm 0.005 \qquad \text{(IV.1)}$$

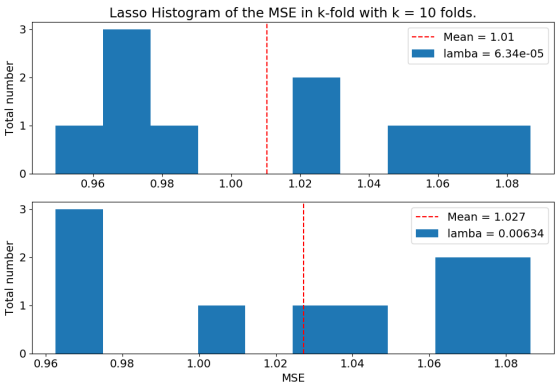
with 95% confidence interval. The mean β-coefficients of the models from k-fold is given in table XV and the error is the standard error of all the calculated β_i-coefficients at a 95% confidence interval.



Figur 4: The MSE between the models created and their respective excluded folds in k-fold cross validation for 10 folds using OLS.



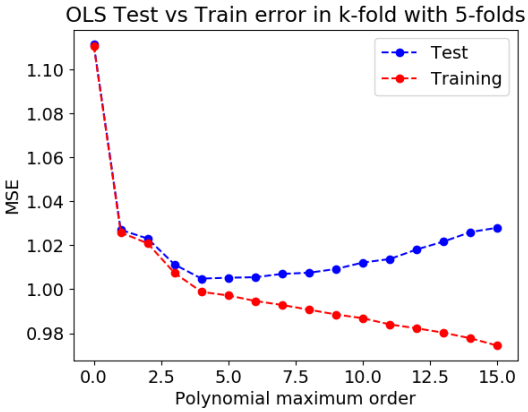
Figur 5: The MSE between the models created and their respective excluded folds in k-fold cross validation for 10 folds using Ridge with the λ-values given in the plot.



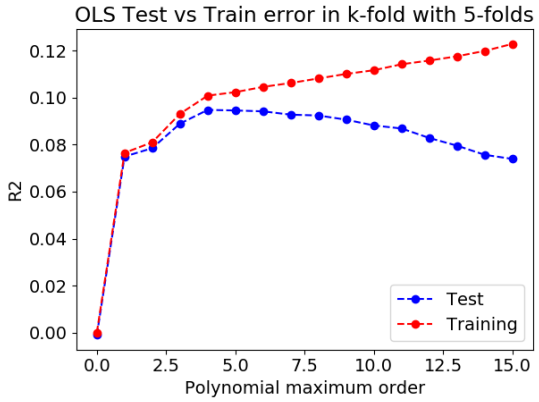
Figur 6: The MSE between the models created and their respective excluded folds in k-fold cross validation for 10 folds using Lasso with the λ-values given in the plot.

In figure 7 the train and test errors from k-fold cross validation with $k = 5$ folds is plotted as function of polynomial complexity. The plotted values are the mean of 50 runs pr polynomial complexity but with random folds, the MSE is than the mean of the 50 runs. The same plot but for R^2 is shown in figure 8.

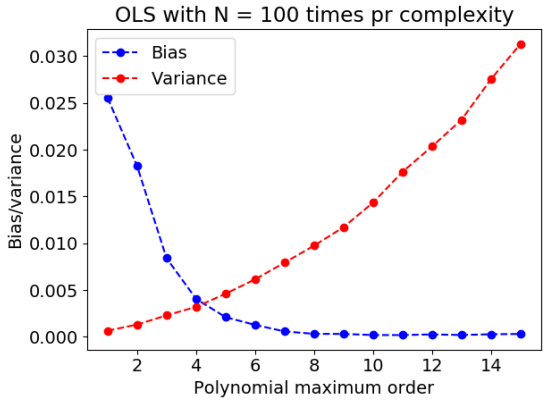
Further in figure 9 the bias and variance is plotted against the model complexity. The bias and variance is calculated between 100 models created by re randomizing the noise 100 times and fitting the OLS model on a train set. The bias and variance is calculated on the test section of the model.



Figur 7: A plot of the MSE from the test and training folds in k-fold cross validation with $k = 5$ folds for the Franke data using OLS. The values are the average of $N = 50$ runs with randomized folds.

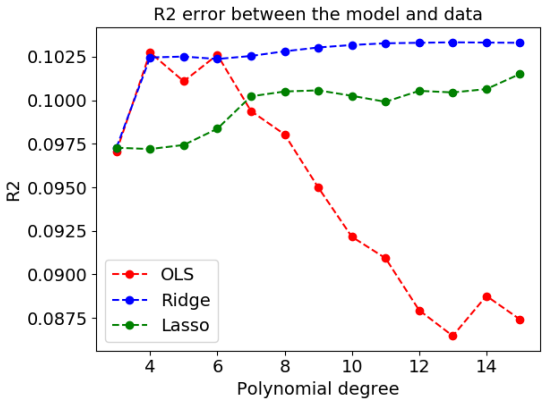


Figur 8: A plot of the R^2 score from the test and training folds in k-fold cross validation with $k = 5$ folds for the Franke data. The values are the average of $N = 50$ runs with randomized folds.

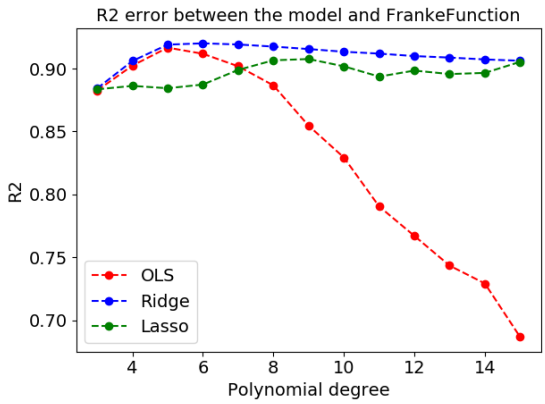


Figur 9: A plot of the Bias^2 and Variance for the Franke data by re randomizing the noise in the data set while keeping the data points constant. The Bias^2 and Variance is calculated based on the models fitted on the test section.

The final part of this subsection will resolve around finding the optimal solution for solving the problem. Therefore in figure 10 and 11 the R^2 score for OLS, ridge and lasso is plotted against the corresponding model complexity. The model is created using a training set whereas the error is calculated using the test data. Further in figure 10 the error estimate is calculated with the real Franke function on the test section of the data while figure 11 shows the error estimate for the data set. Corresponding figures for the MSE is shown in figure 27 for the data set and 26 when comparing to the real Franke function. The minimum MSE and highest R^2 scores is found in table VI with the corresponding polynomial complexity and λ-parameters. For lasso and ridge regression the λ-parameter is my calculated ideal λ for the training set and is plotted in figure 28.



Figur 10: The R^2 between a test set from the data set and the regression methods OLS, ridge and lasso using the "optimal" λ-values for each polynomial degree and method from figure 28.



Figur 11: The R^2 score between a test set and the regression methods OLS, ridge and lasso using the "optimal" λ-values for each polynomial degree and method from figure 28. The test set here is for the real Franke function without noise, but it correspond to the part of the data set omitted in training the model.

Error estimates	OLS	Ridge	Lasso	Degree
MSE Franke	0.00783	0.0075	0.00868	5, 6, 9
MSE Data	0.979	0.978	0.98	4, 13, 15
R^2 Franke	0.917	0.92	0.908	5, 6, 9
R^2 Data	0.103	0.103	0.102	4, 13, 15

Tabell VI: The lowest MSE and highest R^2 estimate with corresponding polynomail degree from figure 10, 11, 26 and 27. The λ-values used for ridge is $\lambda = 0.002458$ and $\lambda = 0.03091$ and for lasso $\lambda = 7.2605 \cdot 10^{-6}$ and $\lambda = 3.7727 \cdot 10^{-5}$ for Franke and data respectively.

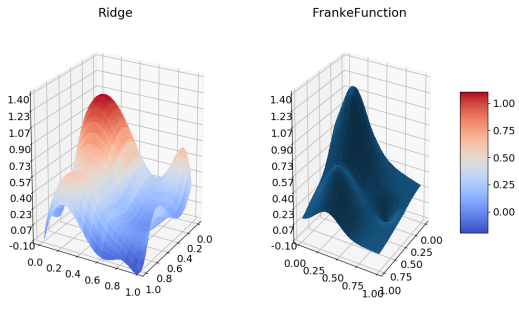
Error estimates	OLS	Ridge	Lasso
MSE Franke	0.009233	0.008609	0.009017
MSE Data	1.001	0.9985	1.001
R^2 Franke	0.9023	0.9089	0.9046
R^2 Data	0.09922	0.1012	0.09854

Tabell VII: The MSE and R^2 scores for the entire data set with the best performing models from table VI for OLS, ridge and lasso regression.

Using the models for OLS, ridge and lasso with the best test data scores shown in table VI i.e: 4^{th} order for OLS, 13^{th} order for ridge and 15^{th} order for lasso. The MSE and R^2 error estimates for the entire data set for those models is shown in table VII. When the models are created on the entire data set the error estimates are shown in table VIII. The ridge model is plotted in figure 12 whereas the two remaining models from table VI is plotted in figure 30 and 29.

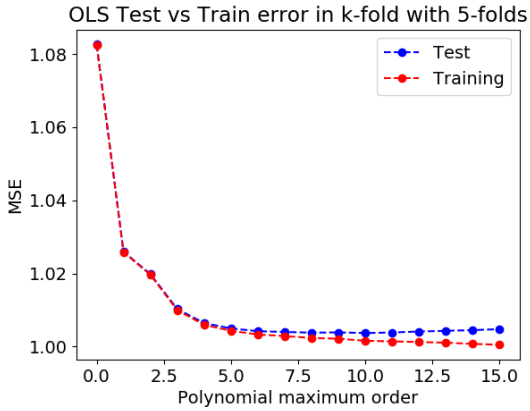
Error estimates	OLS	Ridge	Lasso
MSE Franke	0.007161	0.006443	0.007304
MSE Data	0.9994	0.9971	1.001
R^2 Franke	0.9242	0.9318	0.9227
R^2 Data	0.1004	0.1025	0.09915

Tabell VIII: The MSE and R^2 scores for the entire data using the models with best performance in the test data errors from table VII for OLS, ridge and lasso regression. The models are here used on the entire data grid not just the test or training section.



Figur 12: A 3D plot of the 13$^{\text{th}}$ order Ridge model with $\lambda = 0.03091$ and the Franke function.

The final figure of the Franke functon data is figure 13 and this figure is a plot of the test and training MSE error from k-fold cross validation similar to figure 7. The only difference is that the data set used is of size 400×200. This calculation is also repeated N=50 times for random folds.



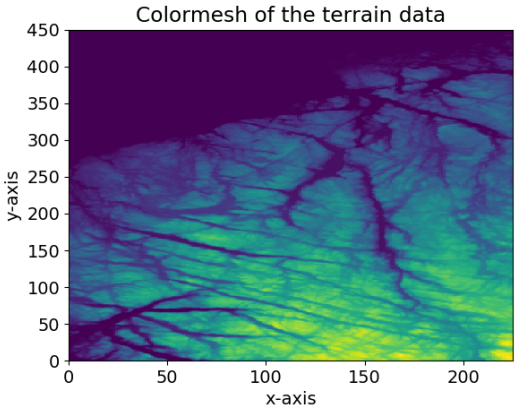
Figur 13: A plot of the MSE from the test and training folds in k-fold cross validation with $k = 5$ folds for the Franke data using OLS. The values are the average of $N = 50$ runs with randomized folds. Unlike figure 7 the data-set here is of size 400×200.

2. *Terrain data*

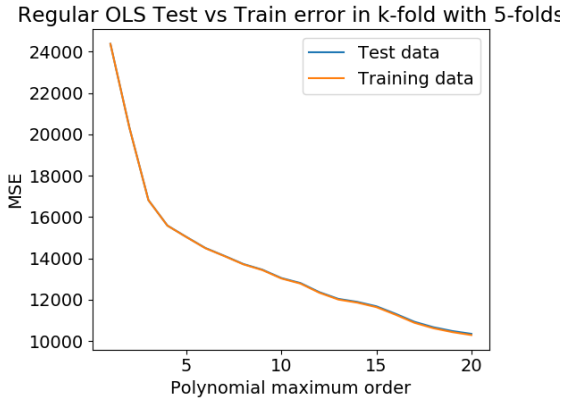
Unlike the previous subsection I will not be including any tables about the β-parameters and their confidence intervals. All of this can be found in `terrain.py` in my github.

A colormesh plot of the terrain data is shown in figure 14. As in the figure and for the major parts of this subsection of the data set is down sampled from 3600×1800 to 400×200 if not stated otherwise. The downsampling is done by taking the average of 9×9 adjutant data points.

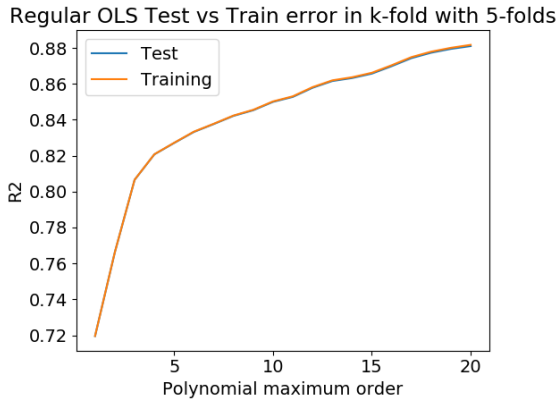
To determine the complexity of the model I will begin by finding the MSE and R^2 scores by k-fold cross validation with 5-folds a total of $N = 10$ times for random folds. In figure 15 and 16 the mean MSE and R^2 from k-fold run N-times are plotted against the used polynomial complexity. The error estimates are for the mean errors in the test and train data in k-fold. In both cases OLS is used.



Figur 14: Colormesh plot of the terrain data used from file "SRTM_data_Norway_2" after being resampled to a size of 400×200.



Figur 15: A plot of the MSE from the test and training folds in k-fold cross validation with $k = 5$ folds for the terrain data using OLS. The values are the average of $N = 10$ runs with randomized folds.



Figur 16: A plot of the R^2 score from the test and training folds in k-fold cross validation with $k = 5$ folds for the terrain data using OLS. The values are the average of $N = 10$ runs with randomized folds.

Based on figure 15 and 16 there exist no ideal polynomial complexity where further increase in complexity would have negligible effect. Thus I will be choosing a complexity of maximum 15^{th} degree[3]. The average MSE and R^2 error for $N = 10$ runs with random folds for a 15^{th} degree polynomial in k-fold with 5 folds can be found in table IX.

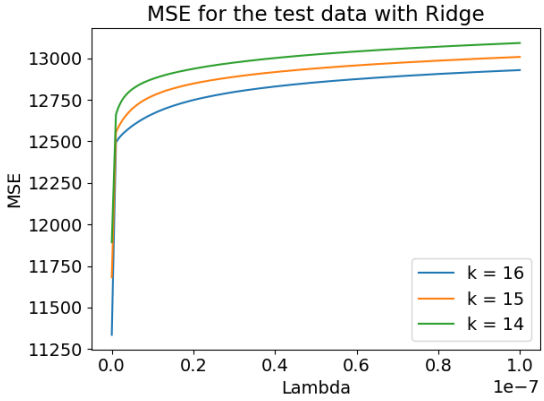
As done with the Franke function the data is splitted into training and test data. This split used to find the optimal λ parameter for ridge and lasso. The result of this is seen in figure 17 and 18 where the MSE in the test data is plotted against the corresponding λ parameter. Further increase in λ-values than demonstrated would prove pointless since it would not improve the accuracy of the model for a 15^{th} degree polynomial complexity. However for lower polynomial complexity $\lambda > 0$ does in fact have a positive effect on the error estimates of the test data.

In figure 19 the mean λ values which minimized the test error in k-fold with 4 folds ran $N = 50$ times is plotted against the model complexity. The error is the standard error $\sigma/\sqrt{N}$ at 95% confidence of the mean λ from k-fold. Notice how the standard error doesn't take into account the number of folds used in k-fold. This is because the λ value is not the mean of the λ-values which minimized to error in each fold, but rather the λ where the sum of the test errors were minimized. In figure 19 where $\lambda = 0$ has not been plotted, thus stopping at 12 degree polynomials.

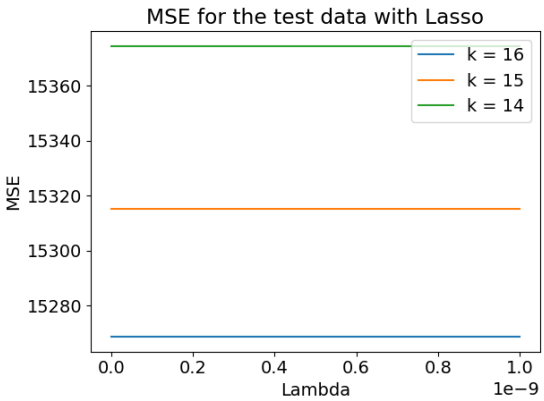
[3] I will later learn to regret this decision, so many hours running programs

Error estimates	OLS
MSE Test	11677.3
MSE Training	11636.6
R^2 Test	0.8657
R^2 Training	0.8662

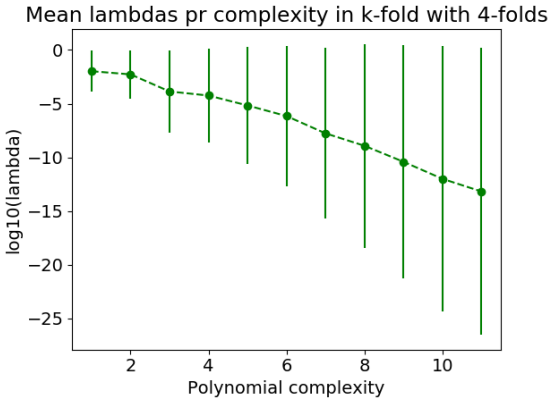
Tabell IX: The error estimates MSE and R^2 from figure 15 and 16 for 15^{th} degree polynomial.



Figur 17: A plot of the mean MSE as a function of λ for the test folds in k-fold cross validation for $k = 5$ folds using ridge regression with the terrain data, with $\lambda \in [10^{-9}, 10^{-7}]$ logarithmic spaced.



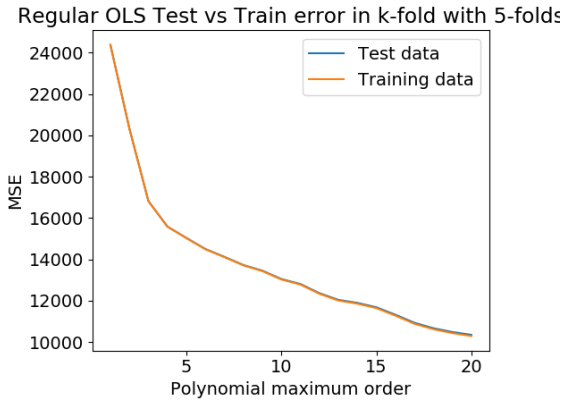
Figur 18: A plot of the mean MSE as a function of λ for the test folds in k-fold cross validation for $k = 5$ folds using lasso regression with the terrain data, with $\lambda \in [10^{-11}, 10^{-9}]$ logarithmic spaced.



Figur 19: The λ-values which minimizes the mean of the test errors in k-fold cross validation with 4-folds using ridge regression. The plotted λ-values are the mean of N = 50 k-fold runs with randomized folds pr complexity.

Using the λ-values from figure 19 and recreating figure 15 with ridge, the final result is plotted in figure 20. Further a comparison of the test MSE scores from ridge and OLS with the same folds is shown in table X. Note however that after 12^{th}

polynomials $\lambda = 0$ and differences are because of round off errors in the calculations.



Figur 20: A plot of the MSE from the test and training folds in k-fold cross validation with $k = 5$ folds for the terrain data using Ridge regression with the optimal λ values from figure 19. The values are the average of $N = 10$ runs with randomized folds.

Complexity	OLS	Ridge
1	24382.158	24382.158
2	20328.86	20328.86
3	16819.655	16819.655
4	15586.651	15586.651
5	15035.038	15035.035
6	14499.407	14499.4
7	14119.283	14119.277
8	13723.398	13723.39
9	13449.225	13449.216
10	13045.589	13045.57
11	12804.559	12804.518
12	12361.396	12361.394
13	12035.657	12035.657
14	11890.92	11890.92
15	11677.324	11677.324
16	11317.505	11317.505
17	10927.481	10927.481
18	10664.592	10664.591
19	10478.644	10478.645
20	10343.452	10343.45

Tabell X: The error estimates for the MSE in the test data from figure15 and 20 for all polynomials from 1 to 20.

Using a 15^{th} degree complexity to create the model the MSE and R^2 error estimates for the entire data set is shown in table XI and the inevitable error is $\sigma = 107.98423$. Using a simple train test split the error estimates is shown in table XII and the inevitable error for the train test case is $\sigma = 107.98418$.

Using k-fold cross validation on the entire data set with $k = 5$ folds the MSE and R^2 error estimates is given in table XIII, and figure 21 is a histogram of MSE for k-fold cross validation with 10 folds for OLS. The same histogram but for ridge and lasso regression with $\lambda = 10^{-8}$ is shown in figure 31 and 32.

Error estimates	OLS	Ridge	Lasso
MSE Data	11641.0	12700.0	15349.0
R^2 Data	0.86612	0.85393	0.82346

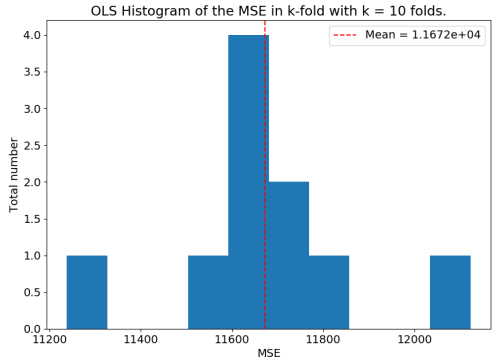
Tabell XI: The error estimates MSE and R^2 from figure 15 and 16 for 15^{th} degree polynomial.

Error estimates	OLS	Ridge	Lasso
MSE Data	11632.0	12722.0	15305.0
R^2 Data	0.86617	0.85363	0.82392
MSE Data	11676.0	12813.0	15467.0
R^2 Data	0.86581	0.85274	0.82224

Tabell XII: The error estimates MSE and R^2 for a OLS model created using the entire terrain data set with a 15^{th} degree complexity. The λ-values used for ridge and lasso regression is $\lambda = 10^{-8}$ for both.

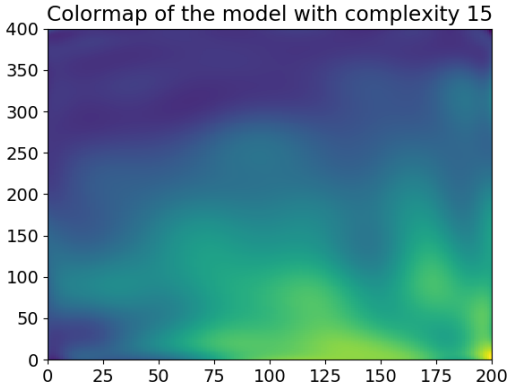
Error estimates	OLS	Ridge	
MSE Data	11700.0 ± 362.0	12700.0 ± 463.0	154(
R^2 Data	0.866 ± 0.00471	0.853 ± 0.00533	0.82
MSE Data	11600.0 ± 90.1	12700.0 ± 113.0	153
R^2 Data	0.866 ± 0.00117	0.854 ± 0.00131	0.82

Tabell XIII: The mean error estimates MSE and R^2 for the test data with OLS, ridge and lasso using k-fold cross validation with $k = 10$ folds. The λ-values used for ridge and lasso regression is $\lambda = 10^{-8}$ for both. The error shown is the standard deviation with a confidence of 95%.

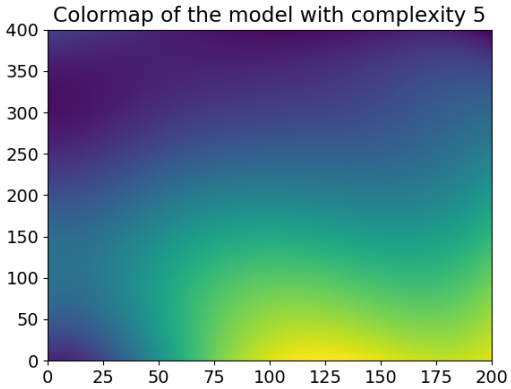


Figur 21: The MSE for the test folds in k-fold cross validation with $k = 10$ folds using OLS.

Using 15^{th} degree complexity for the entire data set a colormap of the model is shown in figure 22. For comparison a 5^{th} degree polynomial fit has the following error estimates $MSE = 15026$ and $R^2 = 0.8272$ and the colorplot is shown in figure 23.



Figur 22: A colormap of the model created using 15^{th} degree polynomial complexity with OLS on the entire data set.



Figur 23: A colormap of the model created using 5^{th} degree polynomial complexity with OLS on the entire data set.

V. DISCUSSION

1. Regression methods

In this subsection I will be comparing the different regression methods not the data sets themselves. This will be carried out in a later subsection.

As expected the OLS models outperformed both ridge regression and lasso regression when fitting the entire data set, and evaluating the models on the same set the models were created. This should come as no surprise since the derivation of OLS is based around minimizing the cost function. From the vector space derivation the OLS model is the closest point in the vector space spanned by $\boldsymbol{X}$ to the data set. It can be seen from table II and table III where the error was smaller than both ridge and lasso.

The problem where OLS may struggle is when you want a predictive model. OLS is very sensitive to over fitting as seen in figure 7. This is however less a problem with a larger data set as seen in figure 13. Ridge and lasso are far less sensitive as seen in figure 10 and 11. It's clear that their strengths lies in the fact that higher order complexity can be utilized without overfitting. As an extension they are also more stable when the data consist of fewer data points.

In my case ridge regression gave constantly better results for both the Franke function and the terrain data than lasso regression did. This may have something to do with how they shrink the β coefficients. As seen in table II lasso regression has a tendency to make coefficients zero and way smaller than ridge with larger λ-values. Therefore lasso regression may be even more stable than ridge for even higher complexity than tested. Figure 11 and 10 may seem to indicate such an observation, since the lasso were either stable or improving while ridge where either stable or degrading.

So far I have only covered how the regression methods behaved for a small data sets. As evident in figure 13 OLS is far more stable for a higher number of data points n. This is also very evident in figure 15 where there is almost no difference between the test and training error. Further from figure 17 and 18 it's clear that in these cases ridge and lasso regression would only worsen the result. This may simply be because the strengths of ridge and lasso regression is that it can utilize higher degree polynomial complexity than OLS without over-fitting.

An interesting result is shown in figure 19 where $\lambda \to 0$ for larger polynomial complexity. This is unexpected as I would expect the ideal λ-values to increase as it does with the Franke function data as seen in figure 28 to keep the model stable. My guess for this is that the data is to advanced for lower polynomials give good estimates, and thus changing λ would be similar to tilting a plane. For more advanced models less and less tilting is required. This is however just a wild guess. It is however clear that the polynomial order is too low to properly make use of the stabilizing qualities of ridge and lasso. Thus making OLS a better option since my used polynomials are rather stable without shrinking.

As a short summary, OLS is the better option for fitting entire data sets or when the complexity and number of data points are high. Ridge and lasso are better options when there are fewer data points and OLS begins to overfit the data since the shrinkage keeps the model stable. For very high complexity, lasso may be the better option than ridge. This is because the shrinkage of λ-values are greater for lasso than for ridge as seen in table II, and figure 17 and 18 shows a more positive trend for lasso than for ridge at higher complexity.

2. Data sets

So far the regression methods haven been the focal point. In this subsection the data sets and their best models will be in focus.

We will begin with the terrain data. It's very clear that from figure 15 that there is very little overfitting when using OLS. This is furt-

her emphasized in table X where the difference between OLS and ridge regression with the ideal λ are practically non existent. Although I will say that ridge regression is smaller from 4^{th} degree up to 12^{th} degree polynomials. After this all ideal *lambda* values were 0. There is however no point in using so low polynomial degrees since higher order are complexity is objectively better as seen by the MSE. Especially since there are no noticeable overfitting. By comparing figure 14 with the models in figure 22 and 23 there is clear that the 15^{th} degree complexity is far more capable of reproducing the original surface.

Lasso regression is even worse than ridge regression in fitting the terrain data. Unlike ridge which gave some (very very minor) improvements in the test error for low polynomial degrees, lasso is objectively the worst option. This is evident in figure 18 where even $\lambda = 0$ gave worse error estimates than OLS.

The terrain data was simply to complex and had to many data points such that OLS could not over-fit. Because of the lack of overfitting ridge and lasso lost the advantage of higher order complexity.

For the Frank function the linear regression which gave the best fit is not as simple as for the terrain data. Unlike the for the terrain data, lasso regression out preformed OLS in one specific case.

I will here look at table VI, VII and VIII. This is simply because the data shown in table VII and VIII are based on model I would have chosen. This is because these models are the models which minimized the error estimate for the test data in table VI. It is worth noting that ridge outperformed both lasso regression and OLS while lasso outperformed OLS with respect to real Franke function in table VIII, however OLS generally outperformed Lasso. A general trend was that ridge regression gave better results when used in predictions wile OLS gave the better result when used on entire data set. There is however worth noting that the difference between the error estimates for all regression methods are very similar.

All regression methods give good results in reproducing the original data, and a point could be made for all of the methods. I would however make a point in noting that finding the optimal λ-values for ridge and lasso is far computationally demanding than using OLS. Further the extra complexity need places more emphasis the computational cost of these two methods. Albeit I believe from table VII and figure 28 that ridge and lasso may be safer options to avoid overfitting, the extra computational power need does make OLS seem like the better option. Time used on finding λ may instead be used on finding the ideal polynomial complexity to avoid overfitting, but if the end goal is to make predictive focused models ridge and lasso seems like the better option.

3. Bias-variance tradeoff

I will end this section as a whole with a small discussion about the bias-variance tradeoff. This part will not resolve around the regression methods but the over-fitting itself.

To do this figure 7 and 9 will play a central role. As seen in figure 7 the training and test data starts diverging. The reason for this is that the model starts fitting not only to the underlying function, but to the noise of the data set. As seen in figure 9 the variance between multiple predicted models with re randomized noise is increasing, while the bias squared is decreasing. The ideal model would than be when the sum of the bias and variance terms are at the smallest. This is in great agreement with my result showing that polynomials for OLS has smallest error between 4 and 6.

For ridge and lasso the bias-variance curves would differ. They impose a penalty on the coefficients and thus reducing the variance term. This will however impact the bias term in comparison to OLS for the same degree. Thus ridge and lasso is based around the principle that the gain in bias is less than the reduction in variance.

VI. CONCLUSION

OLS outperforms both ridge and lasso regression for larger more complex data sets such as the terrain data. Here the $MSE = 11677$ for a polynomial of 15^{th} degree complexity. For lasso regression the same error estimate gave at best $MSE \approx 15000$ for $\lambda = 0$, whereas for ridge only very small λ-values could give result near that of OLS.

For the smaller data set such as the Franke function OLS has a tendency to over-fit for reasonably low complexity. This is seen from the bias-variance tradeoff where the ideal polynomial degree is around 5^{th}. In these cases the shrinkage for β-parameters in ridge and lasso are more stable and are less prone to overfitting. They therefore works better with higher complexity. When predicting data a high complexity ridge model with $MSE = 0.0086$ and a high complexity lasso model with $MSE = 0.009$ outperformed a 4th order OLS model with $MSE = 0.0092$ when predicting on the real Franke function.

Therefore my conclusion is that OLS works best for larger data sets, while ridge and lasso works best for smaller data sets where overfitting may be problematic.

Tillegg A: Tables

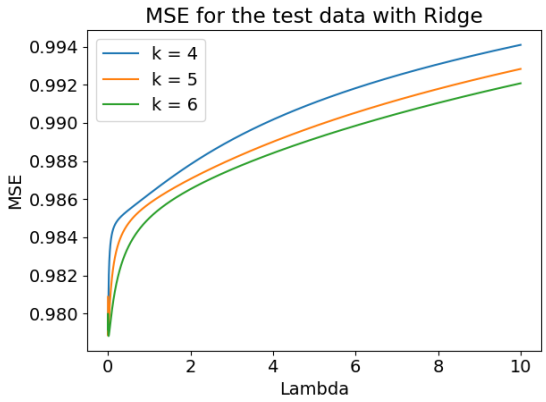
β	OLS	Ridge	Lasso
β_0	0.457 ± 0.469	0.699 ± 0.285	1.05 ± 0.437
β_1	6.04 ± 5.45	3.16 ± 1.94	-0.0837 ± 4.86
β_2	5.09 ± 5.38	2.92 ± 1.92	0.555 ± 4.79
β_3	-23.1 ± 26.8	-12.5 ± 5.22	-2.43 ± 23.4
β_4	-5.23 ± 20.2	1.59 ± 4.5	1.81 ± 18.4
β_5	-23.2 ± 27.3	-13.2 ± 5.24	-4.53 ± 23.7
β_6	15.3 ± 61.2	6.07 ± 5.82	0.639 ± 53.0
β_7	38.0 ± 44.5	8.66 ± 6.47	3.29 ± 40.4
β_8	-2.12 ± 43.1	-6.93 ± 6.5	-3.63 ± 39.3
β_9	29.4 ± 63.1	9.71 ± 5.75	2.27 ± 54.3
β_{10}	16.1 ± 64.8	9.88 ± 6.5	0.875 ± 56.5
β_{11}	-59.4 ± 48.7	-9.13 ± 8.5	0.0 ± 44.9
β_{12}	21.7 ± 44.4	11.1 ± 9.01	0.0 ± 41.1
β_{13}	-24.2 ± 46.8	-6.5 ± 8.68	-0.00225 ± 43.4
β_{14}	-5.41 ± 66.7	8.73 ± 6.38	1.56 ± 57.7
β_{15}	-14.8 ± 25.7	-7.35 ± 4.43	-0.00429 ± 22.7
β_{16}	23.6 ± 22.0	0.152 ± 6.9	-2.27 ± 20.7
β_{17}	0.97 ± 21.1	-1.3 ± 9.03	0.0 ± 20.1
β_{18}	-13.5 ± 21.1	-4.3 ± 9.09	1.05 ± 20.2
β_{19}	19.9 ± 21.2	6.43 ± 6.88	0.0 ± 20.1
β_{20}	-5.79 ± 26.3	-8.38 ± 4.45	-0.216 ± 23.0

Tabell XIV: The β-values for the OLS, Ridge and Lasso regression models on the training data with their respective confidence interval at 95%. For Ridge $\lambda = 4.95 \cdot 10^{-3}$ while for Lasso $\lambda = 6.34 \cdot 10^{-5}$.

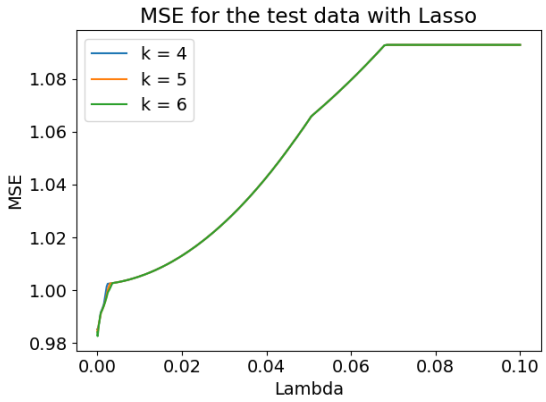
β	OLS
β_0	0.375 ± 0.0435
β_1	8.04 ± 0.535
β_2	4.28 ± 0.536
β_3	-32.0 ± 2.64
β_4	-10.3 ± 2.21
β_5	-13.9 ± 2.44
β_6	35.9 ± 5.76
β_7	40.2 ± 4.49
β_8	4.61 ± 4.73
β_9	4.92 ± 5.05
β_{10}	-6.54 ± 5.79
β_{11}	-50.2 ± 5.05
β_{12}	8.06 ± 5.84
β_{13}	-19.2 ± 4.74
β_{14}	18.2 ± 4.98
β_{15}	-5.8 ± 2.23
β_{16}	17.4 ± 2.38
β_{17}	4.79 ± 1.96
β_{18}	-10.5 ± 2.93
β_{19}	14.8 ± 2.48
β_{20}	-13.6 ± 1.92

Tabell XV: The β-values for the OLS regression method using 10 folds in k-fold cross validation on the training data in a train/test data set. The confidence interval is 95%, and is standard error between the corresponding β_i-coefficients i.e the std for β_0, β_1, β_2 etc.

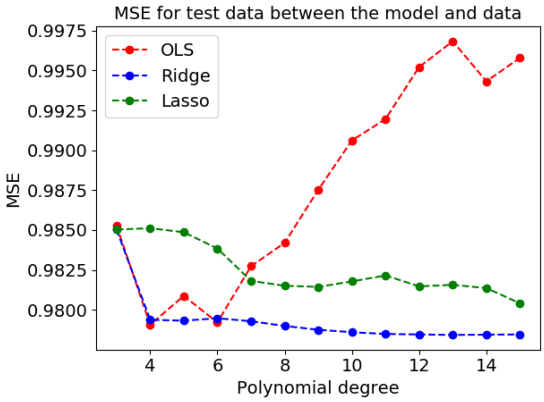
Tillegg B: Figures



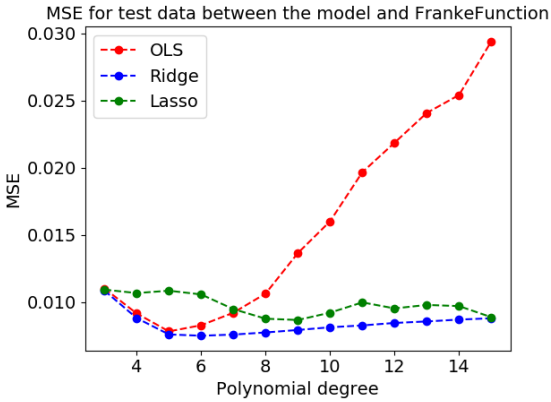
Figur 24: The MSE between the test set and the predicted model using Ridge regression for different polynomial orders and λ values. It's the same plot as figure 2 but for larger λ values i.e $\lambda \in [0, 10]$ logarithmic spaced.



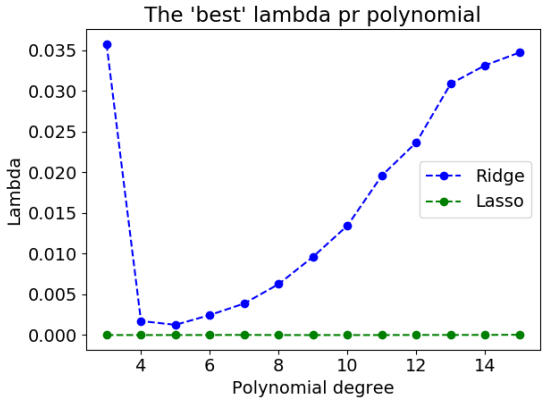
Figur 25: The MSE between the test set and the predicted model using Ridge regression for different polynomial orders and λ values. It's the same plot as figure 3 but for larger λ values i.e $\lambda \in [0, 0.1]$ logarithmic spaced with 1001 iterations.



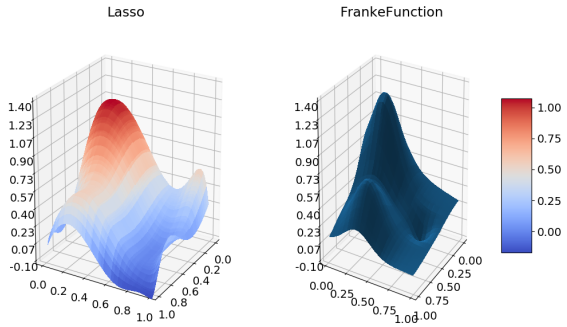
Figur 26: The MSE between a test set from the data and the regression methods OLS, ridge and lasso using the "optimal" λ-values for each polynomial degree and method from figure 28.



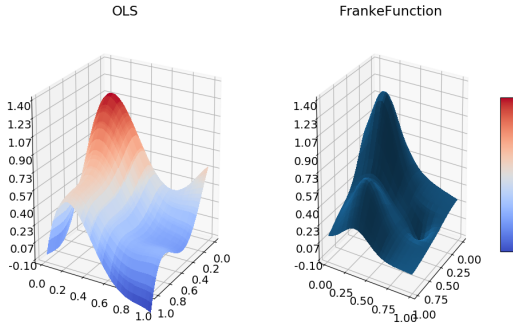
Figur 27: The MSE between a test set and the regression methods OLS, ridge and lasso using the "optimal" λ-values for each polynomial degree and method from figure 28. The test set here is for the real Franke function without noise, but it correspond to the part of the data set omitted in training the model.



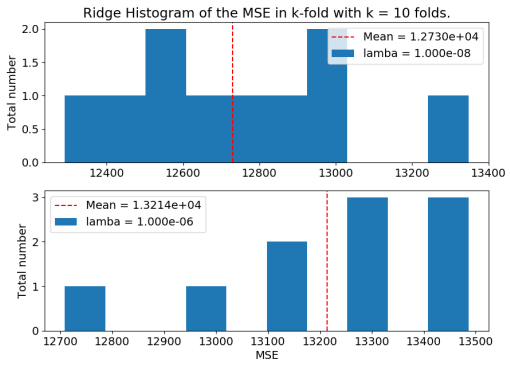
Figur 28: The ideal λ values for the Franke function.



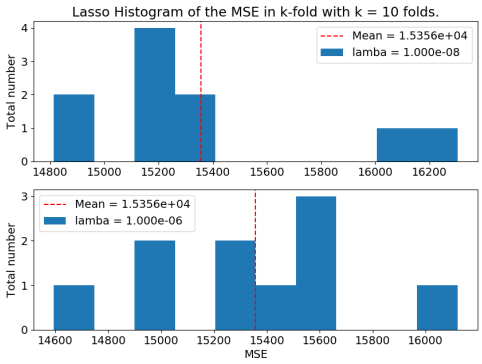
Figur 30: A 3D plot of the 15^{th} order Lasso model from table VII with $\lambda = 3.7727 \cdot 10^{-5}$ beside the Franke function.



Figur 29: A 3D plot of the 4^{th} order OLS model from table VII beside the Franke function.



Figur 31: The MSE for the test folds in k-fold cross validation with $k = 10$ folds for the terrain data and two different λ-values in ridge regression.



Figur 32: The MSE for the test folds in k-fold cross validation with $k = 10$ folds for the terrain data and two different λ-values in lasso regression.

Tillegg C: Proofs

a. Bias-variance tradeoff proof

$$
\begin{aligned}
\mathbb{E}[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2] &= E[(f(\boldsymbol{x})+\boldsymbol{\epsilon}-\tilde{\boldsymbol{y}})^2] \\
&= \mathbb{E}[(f(\boldsymbol{x})-\tilde{\boldsymbol{y}})^2] + 2\mathbb{E}[(f(\boldsymbol{x})-\tilde{\boldsymbol{y}})\boldsymbol{\epsilon}] + \mathbb{E}[\boldsymbol{\epsilon}^2] \\
&= \mathbb{E}[(f(\boldsymbol{x})-\tilde{\boldsymbol{y}})^2] + 2\mathbb{E}[(f(\boldsymbol{x})-\tilde{\boldsymbol{y}})]\mathbb{E}[\boldsymbol{\epsilon}] + \sigma^2 \\
&= \mathbb{E}[(f(\boldsymbol{x})-\tilde{\boldsymbol{y}})^2] + \sigma^2 \\
&= \mathbb{E}[(f(\boldsymbol{x})+\mathbb{E}[\tilde{\boldsymbol{y}}]-\mathbb{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})^2] + \sigma^2 \\
&= [f(\boldsymbol{x})-\mathbb{E}[\tilde{\boldsymbol{y}}]]^2 + \mathbb{E}\left[\tilde{\boldsymbol{y}}-\mathbb{E}[\tilde{\boldsymbol{y}}]\right]^2 + \sigma^2 \\
&= [f(\boldsymbol{x})-\mathbb{E}[\tilde{\boldsymbol{y}}]]^2 + \mathbb{E}\left[\tilde{\boldsymbol{y}}-\mathbb{E}[\tilde{\boldsymbol{y}}]\right]^2 + \sigma^2 \\
&= \mathrm{Bias}^2(\tilde{\boldsymbol{y}}) + \mathrm{Var}(\tilde{\boldsymbol{y}}) + \sigma^2.
\end{aligned}
$$

A couple small comments to the above calculations. For two independent variables $\boldsymbol{x}$ and $\boldsymbol{y}$ the expectation value goes as follows

$$\mathbb{E}[\boldsymbol{x}\boldsymbol{y}] = \mathbb{E}[\boldsymbol{x}]\mathbb{E}[\boldsymbol{y}]. \tag{C.1}$$

This is the case for $\boldsymbol{\epsilon}$, $\tilde{\boldsymbol{y}}$ and $f(\boldsymbol{x})$. Therefore

$$2\mathbb{E}[(f(\boldsymbol{x})-\tilde{\boldsymbol{y}})\boldsymbol{\epsilon}] = 2\mathbb{E}[(f(\boldsymbol{x})-\tilde{\boldsymbol{y}})]\mathbb{E}[\boldsymbol{\epsilon}] = 0 \tag{C.2}$$

since $\mathbb{E}[\boldsymbol{\epsilon}] = 0$. Another unclear term is the following

$$2\mathbb{E}\left[(f(\boldsymbol{x})+\mathbb{E}[\tilde{\boldsymbol{y}}])\,(\tilde{\boldsymbol{y}}-\mathbb{E}[\tilde{\boldsymbol{y}}])\right] = 0 \tag{C.3}$$

which I omitted writing in line six because it's zero. This is zero because

$$\mathbb{E}\left[\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{\boldsymbol{y}}]\right] = \mathbb{E}[\tilde{\boldsymbol{y}}] - \mathbb{E}[\mathbb{E}[\tilde{\boldsymbol{y}}]] \tag{C.4}$$

$$= \mathbb{E}[\tilde{\boldsymbol{y}}] - \mathbb{E}[\tilde{\boldsymbol{y}}] = 0. \tag{C.5}$$

---

[1] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer New York, New York, NY, 2009.

[2] Morten Hjorth-Jensen. Lectures notes in fys-stk4155. data analysis and machine learning: Linear regression and more advanced regression analysis. https://compphysics.github.io/MachineLearning/doc/pub/Regression/html/Regression.html, 2019. [Online; accessed 23-September-2019].

[3] David C Lay. Linear algebra and its applications, 2016.

[4] whuber (https://stats.stackexchange.com/users/919/whuber). The proof of shrinking coefficients using ridge regression through spectral decomposition. Cross Validated. URL:https://stats.stackexchange.com/q/220324 (version: 2017-08-11).

[5] Wikipedia. Regression analysis — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Regression%20analysis&oldid=914514224, 2019. [Online; accessed 14-September-2019].