University of Oslo

FYS-STK4155

Helle Bakke

Project 1: Regression Analysis

October 7, 2019

Contents

Abstract

The Ordinary Least Squares (OLS) method, Ridge and Lasso regression were studied in detail in order to estimate generated and real data. By evaluating the fit of polynomials of varying degree, the mean squared error (MSE), R^2 score and bias-variance trade-off were used to assess the models. When investigating the bias and variance as functions of model complexity, it became clear that over-fitting was not possible with either data set. It was found that more complex models generally performed well on the data, with the OLS method fitting a 5^{th} degree polynomial performing best. Ridge regression with $\lambda = 0.0001$ fitting the same polynomial degree performed almost equally well, while Lasso regression with the same hyperparameters simplified the model to some extent.

1 Introduction

Machine learning is a hot topic in both science and industry these days. It is the study of algorithms and statistical models that enables computers to perform tasks without explicit instructions. The concept of machine learning can be a bit abstract and overwhelming, hence it is necessary to start at the basics to better understand what it comprises. In this project we study different regression methods in detail, including the Ordinary Least Squares (OLS) method, Ridge regression and Lasso regression. We study how to fit polynomials to a specific two-dimensional function called the *Franke function*. Our models are then assessed by employing resampling techniques. Finally, we introduce real terrain data and try to reproduce it using our developed methods.

The structure of this report is as follows. We start by introducing the different regression methods, statistical analysis tools, data sets and algorithm in Section 2. In Section 3 we present and analyse our findings. Section 4 contains a discussion of the results, and Section 5 gives a conclusion and suggestions for future work.

2 Method

2.1 Regression methods

Regression is a statistical measurement to determine the strength of the relationship between a dependent variable $\hat{z}$ and a set of independent variables $\hat{x}$. Given the data set $\hat{z} = [z_0, z_1, ..., z_{n-1}]$ and $\hat{x} = [x_0, x_1, ..., x_{n-1}]$, a function describing the relationship between z_i and x_i can be estimated with

$$z_i = \tilde{z}_i + \epsilon_i \tag{1}$$

where $\tilde{z}_i$ is any function $f(x_i)$ and ϵ_i is the error term. In linear regression, $\hat{\tilde{z}}$ is expressed in terms of the *design matrix* $\hat{X}$ and the *regression parameters* $\hat{\beta}$ as

$$\hat{\tilde{z}} = \hat{X}\hat{\beta} \tag{2}$$

and is the *prediction* of $\hat{z}$. The design matrix $\hat{X}$ has dimension $(n \times p)$, where n is the number of samples and p is the number of features measured in all samples. $\hat{\beta}$ is a vector of length p. Equation (1) can then be written in matrix form as

$$\hat{z} = \hat{\tilde{X}}\hat{\beta} + \hat{\epsilon} \tag{3}$$

2.1.1 Ordinary Least Squares (OLS)

In regression analysis, the OLS method estimates the unknown parameters in a linear regression model. When solving a linear problem, the goal is to find the optimal parameters β_i. Instead of solving equation (2) as a linear algebra problem, we can minimise the residual sum of squares (RSS) given by

$$RSS = \sum_{i=0}^{n-1}(z_i - \tilde{z}_i)^2 \tag{4}$$

We rewrite the RSS on matrix form, such as

$$RSS = (\hat{z} - \hat{\tilde{z}})^T(\hat{z} - \hat{\tilde{z}}) = (\hat{z} - \hat{X}\hat{\beta})^T(\hat{z} - \hat{X}\hat{\beta}) \tag{5}$$

Next, we take the derivative of the residual sum of squares with regards to β_j

$$\begin{aligned}\frac{\partial[RSS(\hat{\beta})]}{\partial\beta_j} &= \frac{\partial}{\partial\beta_j}\left[\sum_{i=1}^{n-1}(z_i - x_{i,0}\beta_0 - x_{i,1}\beta_1 - x_{i,2}\beta_2 - ... - x_{i,p-1}\beta_{p-1})^2\right] \\ &= -2\sum_{i=1}^{n-1} x_{ij}(z_i - x_{i,0}\beta_0 - x_{i,1}\beta_1 - x_{i,2}\beta_2 - ... - x_{i,p-1}\beta_{p-1})\end{aligned}$$

where $j = 0, 1, ..., p-1$. We can rewrite this into a more compact form, such as

$$\frac{\partial[RSS(\hat{\beta})]}{\partial\hat{\beta}} = -\hat{X}^T(\hat{z} - \hat{X}\hat{\beta}) \tag{6}$$

where the factor 2 disappears as a result of the minimisation[1]. Further, we set $\frac{\partial[RSS(\hat{\beta})]}{\partial\hat{\beta}} = 0$ and obtain an expression for $\hat{\beta}$ given by

$$\hat{\beta} = (\hat{X}^T\hat{X})^{-1}\hat{X}^T\hat{z} \tag{7}$$

In addition to finding the coefficients that best fit the data, the OLS method also finds the *unbiased* coefficients. This means that all variables are treated equally. Which independent variable is more important is not considered, and the OLS method simply finds the set of β-values that result in the lowest residual sum of squares.

2.1.2 Ridge regression

Ridge regression is a variant of linear regression. It is similar to the OLS method, but includes a "penalty" term that increases the residual sum of squares. The Ridge coefficients are calculated as

$$\hat{\beta}^{Ridge} = \underset{\beta}{\operatorname{argmin}}\left[\sum_{i=1}^{n}(z_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}\beta_j^2\right] \tag{8}$$

where $\lambda \geq 0$ is a called a *hyperparameter*, and controls the amount of shrinkage. We see that by including the λ-term, we put a penalty on the β-values.

[1]By removing the factor of 2 from the equation, only the steepness of the function changes, not the actual minimum.

It is worth noting that the penalty term does not include the intercept β_0. The reason for this, is that by putting a constraint on the intercept we would make the entire method depend on the origin chosen for $\hat{z}$ (Hastie et al., 2005). By regulating the intercept, β_0 would deviate in a way such that if we were to add a constant c to each z_i, the predictions would not shift by the same amount c.

The criterion in equation (8) in matrix form can be written as

$$RSS = (\hat{z} - \hat{X}\hat{\beta})^T(\hat{z} - \hat{X}\hat{\beta}) + \lambda\hat{\beta}^T\hat{\beta} \tag{9}$$

We differentiate the RSS in equation (9) with regards to β_j and obtain the expression

$$\hat{\beta} = (\hat{X}^T\hat{X} + \lambda\hat{I})^{-1}\hat{X}^T\hat{z} \tag{10}$$

where $\hat{I}$ is the identity matrix of dimension $(p \times p)$.

2.1.3 Lasso regression

Lasso (least absolute shrinkage and selection operator) regression is similar to Ridge regression, but with an important difference. While Ridge regression performs L_2 regularization through the penalty term, Lasso regression performs L_1 regularization. This is seen in the Lasso estimate defined as

$$\hat{\beta}^{Lasso} = \text{argmin}\left[\frac{1}{2}\sum_{i=1}^{n}(z_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}|\beta_j|\right] \tag{11}$$

where the penalty is equal to the absolute value of the magnitude of coefficients. This constraint makes the solutions non-linear in z_i, hence there is no closed-form solution as in the OLS method and Ridge regression. Due to the nature of the penalty term, increasing values of λ causes some of the β-coefficients to be exactly zero (and eliminated). On the other hand, when $\lambda = 0$ no coefficients are eliminated, and the problem reduces to equation (7). The latter is true for Ridge regression as well. λ controls the strength of the penalty, and can be seen as the amount of shrinkage (Hastie et al. (2005); Glen (2015)).

2.2 Regression analysis

So far, we have introduced the different regression methods used in this project. In order to investigate how well they perform on the data, it is

important to introduce various statistical tools. With these tools, we can determine how well a single regression method performs by itself or for different polynomial degrees, as well as how different regression methods perform compared to each other.

2.2.1 Mean squared error

The mean squared error (MSE) is a measure of the quality of a predicted variable, and is defined as

$$MSE(\hat{z}, \tilde{\hat{z}}) = \frac{1}{n} \sum_{i=0}^{n-1} (z_i - \tilde{z}_i)^2 \tag{12}$$

The MSE gives an average of the squared error between the analytical and predicted values, and is always non-negative. The result should be as close to zero as possible for the model to be considered good.

2.2.2 R^2 score

The R^2 score is a statistical measure indicating how much of the variation of a dependent variable can be explained by the variation of an independent variable. In regression analysis, the R^2 score is used to explain how well the model is predicting the actual values. It is defined as

$$R^2(\hat{z}, \tilde{\hat{z}}) = 1 - \frac{\sum_{i=0}^{n-1} (z_i - \tilde{z}_i)^2}{\sum_{i=0}^{n-1} (z_i - \bar{z}_i)^2} \tag{13}$$

with the mean value being calculated as

$$\bar{z} = \frac{1}{n} \sum_{i=0}^{n-1} \tilde{z}_i \tag{14}$$

The nominator in equation (13) is recognised as the residual sum of squares (RSS), and explains the variation between the actual and predicted variables. The denominator explains the total variation, where the average of the predicted values are subtracted from the actual values. The R^2 score should be as close to 1 as possible, indicating that 100% of the variation in the dependent variable can be explained by the independent variable.

2.2.3 Confidence interval

In statistics, a confidence interval represents an estimated range of values that is likely to contain the true value of an unknown population parameter. Confidence intervals are used to address the uncertainty of an estimated variable. In regression analysis, it is useful to study the confidence interval of the β-parameters as we want to know how confident we can be in our models. In this project, we calculate the 95% confidence interval, an interval that covers 95% of a normal curve. That means that the probability of observing a value outside this area is less than 5% (Sullivan).

The confidence interval for β_j is defined as

$$(\beta_j - Z^{1-\alpha}\sigma_{\beta_j}, \beta_j + Z^{1-\alpha}\sigma_{\beta_j}) \tag{15}$$

where $Z^{(1-\alpha)}$ is the $(1-\alpha)$ percentile of the normal distribution (also called the Z-score) and σ_{β_j} is the standard deviation. The standard deviation can be expressed in terms of the variance as

$$\sigma_{\beta_j} = \sqrt{\mathrm{Var}(\beta_j)} \tag{16}$$

where

$$\mathrm{Var}(\beta_j) = (\hat{X}^T\hat{X})^{-1}_{jj}\sigma^2 \tag{17}$$

The subscripts jj refer to the diagonal elements of $\hat{X}^T\hat{X}$, and σ^2 is calculated as

$$\sigma^2 = \frac{1}{n-p-1}\sum_{i=0}^{n-1}(z_i - \tilde{z}_i)^2 \tag{18}$$

Since we are calculating the 95% confidence interval, $\alpha = 0.025$ (which is half of area not covered by 95% of the normal distribution). The value of Z is then looked up in a Z-score table, giving

$$Z^{(1-0.025)} = 1.96$$

The confidence interval can then be written as

$$(\beta_j - 1.96[(\hat{X}^T\hat{X})^{-1}_{jj}]^{1/2}\sigma, \beta_j + 1.96[(\hat{X}^T\hat{X})^{-1}_{jj}]^{1/2}\sigma) \tag{19}$$

2.3 Resampling

Resampling is the process of repeatedly drawing samples from a data set and fitting each sample to the given model. The goal of resampling is to gain information about the fitted model that would not be available from fitting only once. The result of resampling is often a more accurate estimate of the parameter, since the model is trained multiple times on samples that are varying. The objective is to create a model using *training* data, and then predictions can be made on the new data. Resampling methods can enable us to see how well a regression method performs on the data, which is pretty amazing. It is, however, important to keep in mind that resampling methods can be computationally expensive as they require repeatedly fitting of the model on n different subsets of the same data.

There are a variety of different resampling techniques to choose from, but in this report we keep the focus on *bootstrapping*.

2.3.1 Bootstrap

Bootstrapping is a resampling method that involves drawing samples from a data set. In this project, we start by splitting the data set into training and test data. With the bootstrap method, we always keep the test data constant, while the training data is resampled. Each sample is randomly chosen, meaning that we in principle are able to pick the same data points several times (for every bootstrap). Then, we choose a regression method and assess the model using the different methods described in previous sections of this report.

2.3.2 Bias-variance tradeoff

There are a number of considerations to make when fitting a model. In general, we want the model to fit the training data well. However, it is important that it does not fit too well, as that would cause the model to have a high variance due to overfitting. That means that the model makes a curve that is too complex when fitting the data. If the model is too simple, it has a high bias due to underfitting, meaning that the model makes too many assumptions about the data. Because of these trait, it is crucial that we design a model that balances simplicity and complexity in order to find a midpoint between the extremes (Wikipedia, 2017).

The bias-variance decomposition for mean squared error proceeds in the following. With the true data generated from the model in equation (7), where $\tilde{\hat{z}} = \hat{X}\hat{\beta}$, the $\hat{\beta}$-parameters can be found by optimising the MSE via the cost function

$$C(\hat{X}, \hat{\beta}) = \frac{1}{n}\sum_{i=0}^{n-1}(z_i - \tilde{z}_i)^2 = \mathbb{E}[(\hat{z} - \tilde{\hat{z}})^2] \tag{20}$$

The expectation value can be further written as

$$\begin{aligned}
\mathbb{E}[(\hat{z} - \hat{\tilde{z}})^2] =& \mathbb{E}[(f + \hat{\epsilon} - \hat{\tilde{z}})^2] \\
=& \mathbb{E}[(f + \hat{\epsilon} - \hat{\tilde{z}} + \mathbb{E}[\hat{\tilde{z}}] - \mathbb{E}[\hat{\tilde{z}}])^2] \\
=& \mathbb{E}[(f - \mathbb{E}[\hat{\tilde{z}}])^2 + (\mathbb{E}[\hat{\tilde{z}}] - \hat{\tilde{z}})^2 + 2(f - \mathbb{E}[\hat{\tilde{z}}])(\mathbb{E}[\hat{\tilde{z}}] - \hat{\tilde{z}}) \\
& + 2[(f - \mathbb{E}[\hat{\tilde{z}}])\hat{\epsilon}] + 2[\hat{\epsilon}(\mathbb{E}[\hat{\tilde{z}}] - \hat{\tilde{z}})] + \hat{\epsilon}^2] \\
=& \mathbb{E}[(f - \mathbb{E}[\hat{\tilde{z}}])^2] + \mathbb{E}[(\mathbb{E}[\hat{\tilde{z}}] - \hat{\tilde{z}})^2] + 2\,\mathbb{E}[(f - \mathbb{E}[\hat{\tilde{z}}])(\mathbb{E}[\hat{\tilde{z}}] - \hat{\tilde{z}})] \\
& + 2\,\mathbb{E}[\{(f - \mathbb{E}[\hat{\tilde{z}}])\hat{\epsilon}\}] + 2\,\mathbb{E}[\{\hat{\epsilon}(\mathbb{E}[\hat{\tilde{z}}] - \hat{\tilde{z}})\}] + \mathbb{E}[\hat{\epsilon}^2]
\end{aligned}$$

We know that the expectation value $\mathbb{E}[\hat{\epsilon}] = 0$ because the noise is distributed as $\mathcal{N}(0, \sigma^2)$. We can find the expectation value of $\hat{\epsilon}^2$ since

$$\mathbb{E}[x^n] = \begin{cases} 0 & \text{if } n \text{ is odd} \\ \sigma^2(n-1) & \text{if } n \text{ is even} \end{cases} \tag{21}$$

giving $\mathbb{E}[\hat{\epsilon}^2] = \sigma^2$. This leaves us with the following calculation

$$\begin{aligned}
\mathbb{E}[(\hat{z} - \hat{\tilde{z}})^2] =& \mathbb{E}[(f - \mathbb{E}[\hat{\tilde{z}}])^2] + \mathbb{E}[(\mathbb{E}[\hat{\tilde{z}}] - \hat{\tilde{z}})^2] + 2\,\mathbb{E}[(f - \mathbb{E}[\hat{\tilde{z}}])(\mathbb{E}[\hat{\tilde{z}}] - \hat{\tilde{z}})] \\
=& \mathbb{E}[(f - \mathbb{E}[\hat{\tilde{z}}])^2] + \mathbb{E}[(\hat{\tilde{z}} - \mathbb{E}[\hat{\tilde{z}}])^2] + 2f\,\mathbb{E}[\hat{\tilde{z}}] - 2f\,\mathbb{E}[\hat{\tilde{z}}] \\
& - 2(\mathbb{E}[\hat{\tilde{z}}])^2 + 2(\mathbb{E}[\hat{\tilde{z}}])^2 + \sigma^2 \\
=& \mathbb{E}[(f - \mathbb{E}[\hat{\tilde{z}}])^2] + \mathbb{E}[(\hat{\tilde{z}} - \mathbb{E}[\hat{\tilde{z}}])^2] + \sigma^2 \\
=& \frac{1}{n}\sum_i (f_i - \mathbb{E}[\hat{\tilde{z}}])^2 + \frac{1}{n}\sum_i (\tilde{z}_i - \mathbb{E}[\hat{\tilde{z}}])^2 + \sigma^2
\end{aligned}$$

where we have used that $\mathbb{E}[f] = f$ and $(\mathbb{E}[\hat{\tilde{z}}] - \hat{\tilde{z}})^2 = (\hat{\tilde{z}} - \mathbb{E}[\hat{\tilde{z}}])^2$. Finally, we can write the above expression as

$$\mathbb{E}[(\hat{z} - \hat{\tilde{z}})^2] = \text{Bias}^2(\hat{\tilde{z}}) + \text{Var}(\hat{\tilde{z}}) + \sigma^2 \tag{22}$$

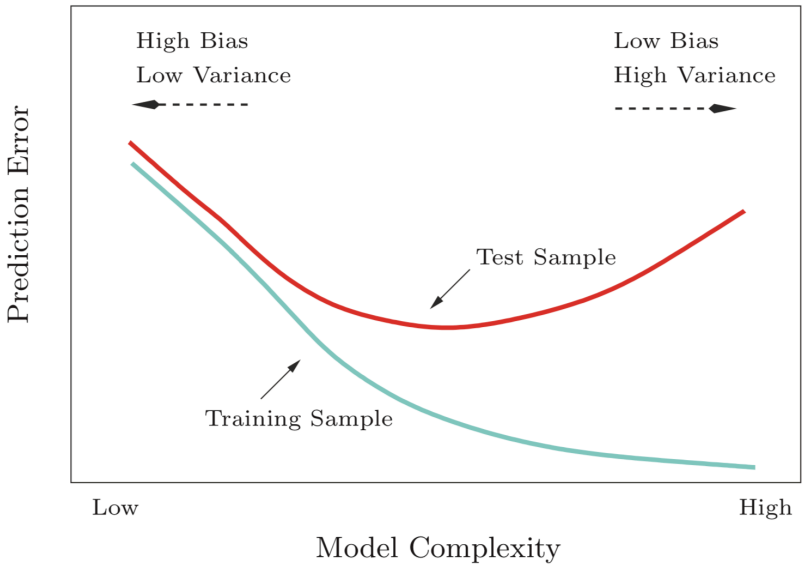

Figure 1: Predicted error of the training and test data as a function of model complexity. Figure taken from Hastie et al. (2005).

where we see which terms account for the bias, variance and *irreducible error*. The irreducible error results from the noise in the data. The simplest way of interpretating the bias-variance tradeoff is by considering Figure 1. The figure shows that as bias increase, variance decrease and vice versa. We observe that low model complexity results in high bias and low variance, while high model complexity results in low bias and high variance. This means that in order to have a model that is neither underfitted nor overfitted, we need a model complexity (i.e. polynomial degree) that is not too high or too low.

2.4 Data sets

We have used two different data sets in this project. First, we study the two-dimensional Franke function given by the equation

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

Franke's function has two Gaussian peaks, each of different height, and a characteristic smaller dip. The second data set is a real digital terrain of the Oslo area obtained from the website `https://earthexplorer.usgs.gov/`. Figure 2 illustrates surfaces of both the Franke function and the real terrain data.

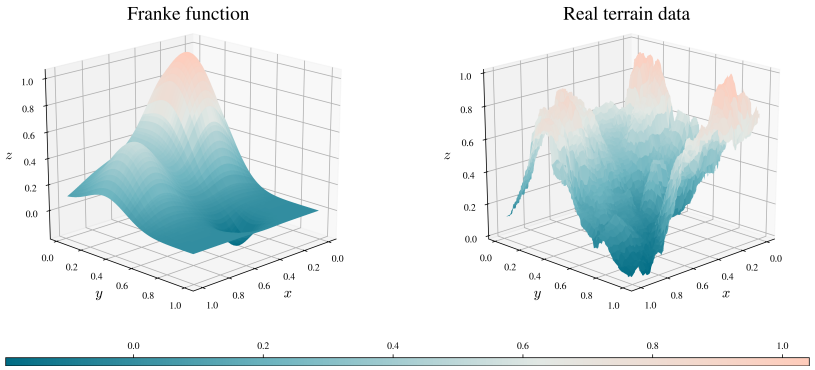

Figure 2: Analytical Franke function and real terrain data plotted as surfaces.

2.5 Algorithm

So far, we have introduced regression methods, statistical analysis tools and the resampling method used in this project. Our main objective is to study how the different regression methods perform on the given data sets. In order to do this, we need to set up various functions for calculating the design matrix $\hat{X}$, $\hat{\beta}$-values, mean squared error (MSE), R^2 score and confidence interval.

As an example, following equation (3) for a 2^{nd} order polynomial, the design matrix has the form

$$\hat{X} = \begin{bmatrix} 1 & x_0 & y_0 & x_0^2 & x_0y_0 & y_1^2 \\ 1 & x_1 & y_1 & x_1^2 & x_1y_1 & y_1^2 \\ 1 & x_2 & y_2 & x_2^2 & x_2y_2 & y_2^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n-1} & y_{n-1} & x_{n-1}^2 & x_{n-1}y_{n-1} & y_{n-1}^2 \end{bmatrix} \tag{23}$$

The $\hat{\beta}$-values are then calculated using either the OLS method in equation (7), Ridge regression in equation (10), or using the functionalities of Scikit-

Learn to perform Lasso regression. This is done for each bootstrap, where the MSE and R^2 score are calculated as well. The bootstrap method is set up as follows:

```
def Bootstrap(X, z, n_boots):
  # split into training and test data
  X_train, X_test, z_train, z_test = train_test_split(X, z, test_size=0.2)

  # arrays for storing mse for training and test data
  mse_train = np.zeros(n_boots)
  mse_test  = np.zeros(n_boots)

  # arrays for storing mse for training and test data
  r2s_train = np.zeros(n_boots)
  r2s_test  = np.zeros(n_boots)

  for k in range(n_boots):
    # create random indices for every bootstrap
    random_index = np.random.randint(X_train.shape[0], size=X_train.shape[0])

    # resample X_train and z_train
    X_tmp = X_train[random_index,:]
    z_tmp = z_train[random_index]

    # obtain beta values for training data
    beta_train = BetaValues(X_tmp, z_tmp)

    # calculate z_predict for training and test data
    zpred_test  = np.dot(X_test, beta_train)
    zpred_train = np.dot(X_tmp, beta_train)

    # calculate MSE for training and test data
    mse_train[k] = MeanSquaredError(z_tmp, zpred_train)
    mse_test[k]  = MeanSquaredError(z_test, zpred_test)

    # calculate r2 score for training and test data
    r2s_train[k] = R2Score(z_tmp, zpred_train)
    r2s_test[k]  = R2Score(z_test, zpred_test)
```

where *train_test_split* is a built-in function in the Scikit-Learn library for splitting the data into training and test samples. Here, the test sample is 1/5 of the total data. The *BetaValues*-function calculates the β-values based on the preferred regression method, and the *MeanSquaredError*- and *R2Score*-functions calculate the MSE and R^2 score using equations (12) and (13). These functions have been benchmarked, see Section 6. The MSE and R^2 score are then calculated by taking the average of the train- and test-arrays after the bootstrapping. The β-values providing the lowest MSE in the test-array is then used to calculate the final prediction of $\hat{z}$, as well as the confidence interval.

3 Results

3.1 The Franke function

We have generated a data set using the Franke function with an added stochastic noise that is normally distributed as $\mathcal{N}(0, 0.1)$. $x, y \in [0, 1]$ are

randomly distributed and sorted, and each vector has length 100. The bootstrap method has been run for 1000 bootstraps, where we have stored the β-values that corresponds to the lowest mean squared error in order to plot the final prediction $\hat{\tilde{z}}$. In order to compare the regression methods, we have created tables and figures that contain results from all regression methods.

Bias-variance tradeoff

In order to deduce important information about our models, we need to analyse the bias-variance tradeoff. Only then are we able to opinionate which model provides the lowest mean squared error, as well as observe what measures we need to take in order to obtain a result that is neither overfitted nor underfitted. It should be noted, however, that with this data set it is extremely difficult to obtain an overfitted result as we are not testing polynomials of sufficiently high degree. Figure 3 shows the bias and variance as functions of model complexity. At the left end of the figure, the bias is high while the variance is low, meaning that the model is underfitted and has a higher MSE. We observe that the variance is constant at ~ 0 for all polynomial degrees, hence we never get an overfitted model with the polynomials tested[2]. Since the variance is constant and the bias is decreasing with increasing model complexity, the MSE for the test data will also decrease in a similar fashion.

Figures 4 and 5 show the bias and variance as functions of model complexity for Ridge and Lasso regression, respectively. We have tested the two regression methods for different values of the hyperparameter λ, as seen in the figure. Again, we observe that the variance is ~ 0 for all polynomial degrees, while the bias decrease as model complexity increase. For a 5^{th} degree polynomial, the bias is lowest when λ is at its lowest value. This is true for both Ridge and Lasso regression, meaning that we will probably find the lowest MSE for these hyperparameters.

Mean squared error

Figure 6 shows the mean squared error as a function of model complexity for the OLS method, a plot inspired by Figure 1. Even though it is intuitive that a 1^{st} order polynomial is not a good fit to our more complicated data set, we have included it in order to explicitly show that this is true. We see that

[2]In reality, the variance is not exactly 0, but rather a small number close to 0.

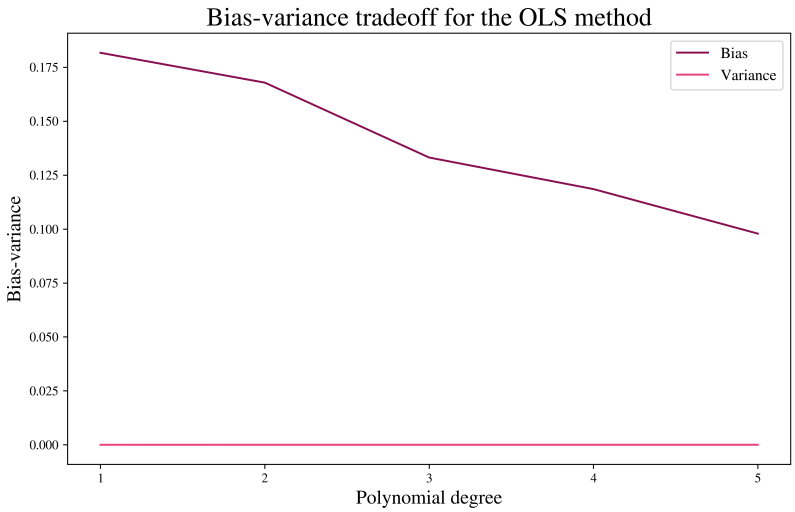

Figure 3: Bias-variance tradeoff as a function of model complexity for the Ordinary Least Squares method.

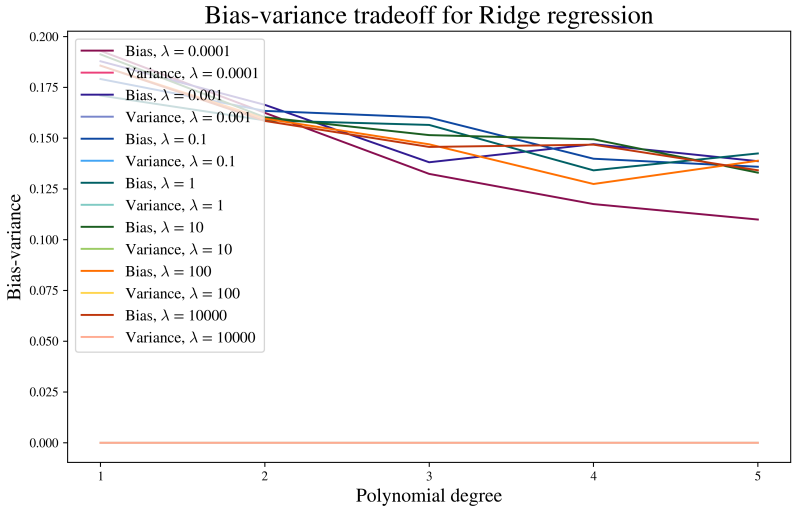

Figure 4: Bias-variance tradeoff as a function of model complexity for Ridge regression with different λ-values.

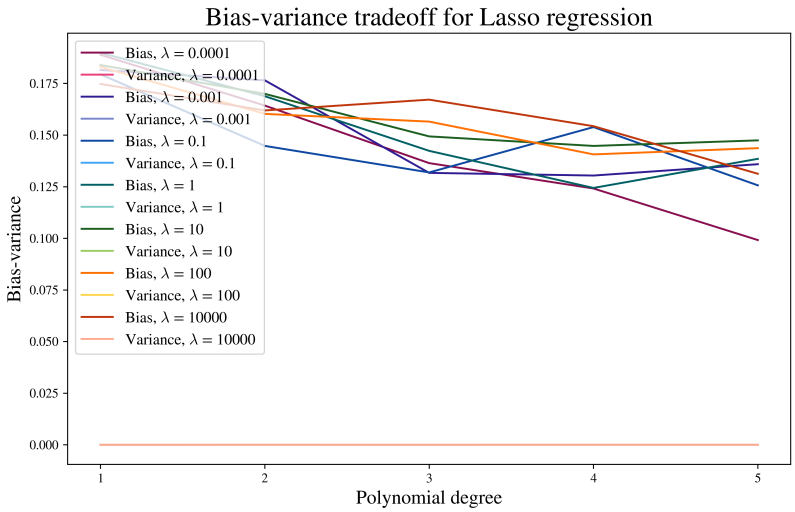

Figure 5: Bias-variance tradeoff as a function of model complexity for Lasso regression with different λ-values.

the MSE for the test data is low, which is in accordance with the training data. If the model was overfitted, the MSE of the test data would start to increase, as shown in Figure 1.

Figures 7 and 8 show the mean squared error plotted as a function of model complexity for Ridge and Lasso regression, respectively. Since both Ridge and Lasso regression includes a hyperparameter λ, we have tested the models for several different values of this parameter. This results in figures of many individual graphs, hence the training MSE is plotted in a strong colour with the test MSE plotted as the corresponding pastel colour for easier reading. Both figures show that the smallest λ-value results in the lowest MSE. An interesting observation is that there is a relatively large difference between the lowest λ-value and the larger ones for 5^{th} degree polynomials. For the Ridge model, this difference starts to develop for 3^{rd} degree polynomials, while in the Lasso model the difference does not become apparent before larger polynomial degrees. This shows that smaller λ-values are necessary in order to minimise the mean squared error. As λ gets smaller, we approach the mean squared error of the OLS method, since both Ridge and Lasso regression reduces to this method when $\lambda = 0$.

Table 1 shows the mean squared error for polynomials up to 5^{th} degree for

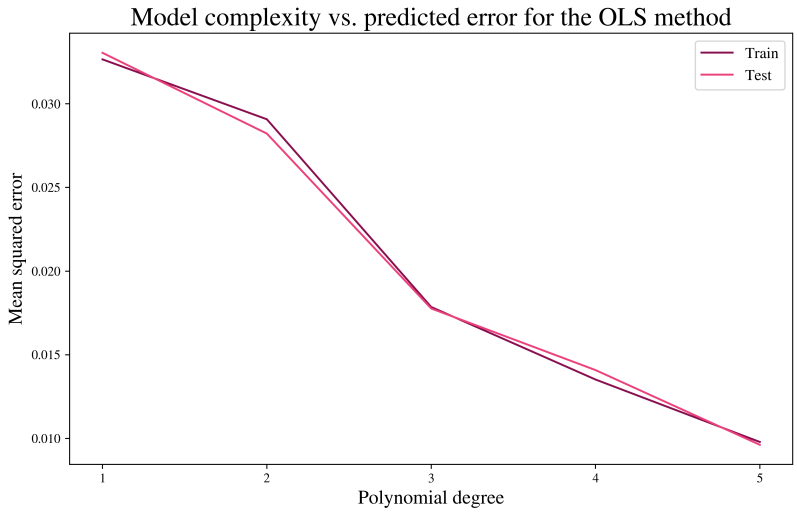

Figure 6: Model complexity vs. predicted error for the Ordinary Least Squares method.

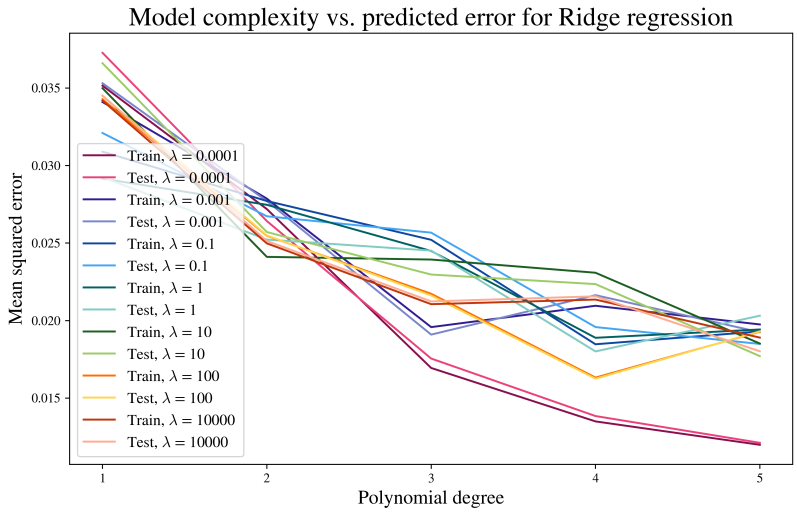

Figure 7: Model complexity vs. predicted error for Ridge regression. The model is tested for 7 different hyperparameters λ ranging between $[0.0001, 10000]$.

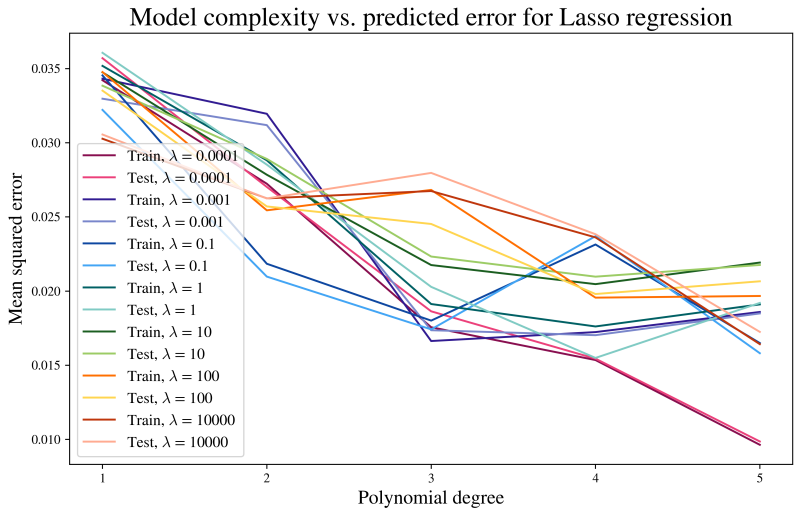

Figure 8: Model complexity vs. predicted error for Lasso regression. The model is tested for 7 different hyperparameters λ ranging between $[0.0001, 10000]$.

the OLS method, as well as Ridge and Lasso regression for the λ-parameter that provides the lowest MSE (as seen in Figures 7 and 8). In general, there are no large differences in the MSE for either method. As the above figures have already shown, fitting of higher order polynomials provides the lowest MSE, while the models are not overfitted.

R^2 score

Figure 9 shows the R^2 score as a function of model complexity for the OLS method. As the polynomial degree increase, the R^2 score increases as well. Figures 10 and 11 show the R^2 score as a function of model complexity for Ridge and Lasso regression, respectively. As with the MSE figures, we have included several different values of the λ-parameter. Both figures show that the R^2 score increases with polynomial degree, but also (in most cases) with decreasing λ. It is clearly seen that for higher order polynomials, the R^2 score is best when $\lambda = 0.0001$. In Ridge regression this is true for almost all polynomial degrees, while in Lasso regression this does not become apparent until we reach 4^{th} order polynomials and higher.

Table 1: Mean squared error for the OLS method as well as Ridge and Lasso regression with $\lambda = 0.0001$.

Degree	MSE_{OLS}	MSE_{Ridge}	MSE_{Lasso}
1^{st}	0.033	0.037	0.036
2^{nd}	0.028	0.026	0.027
3^{rd}	0.018	0.018	0.019
4^{th}	0.014	0.014	0.015
5^{th}	0.010	0.012	0.010

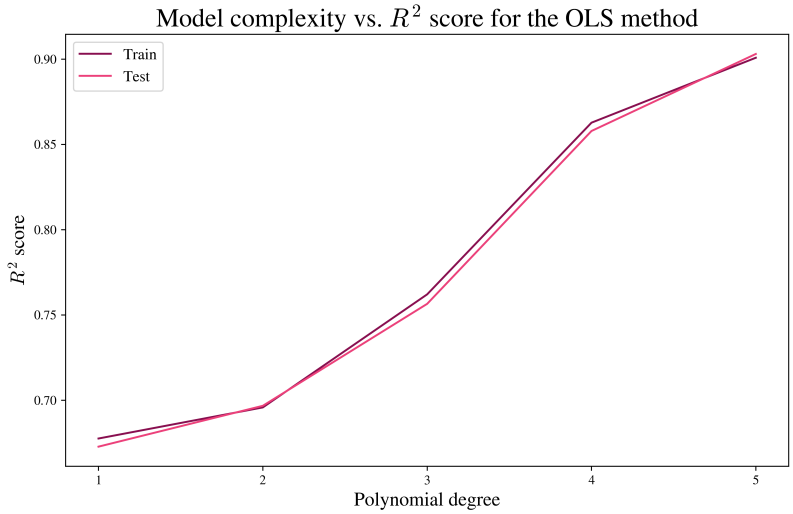

Figure 9: R^2 score as a function of model complexity for the OLS method.

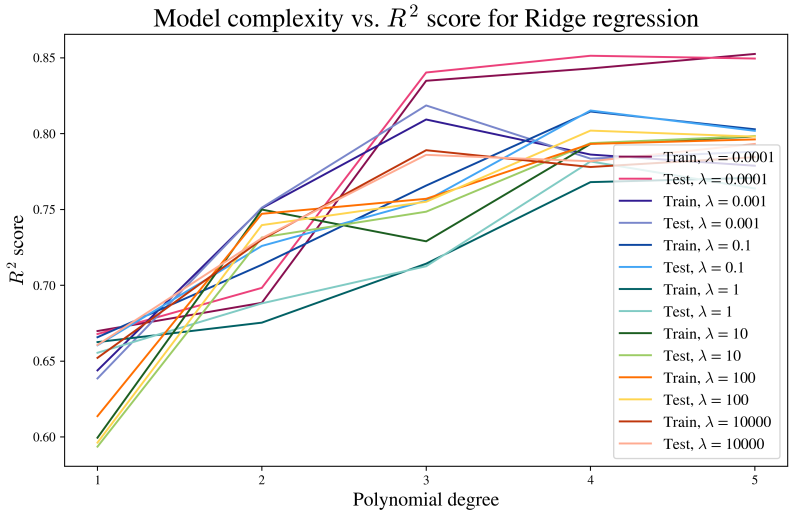

Figure 10: R^2 score as a function of model complexity for Ridge regression.

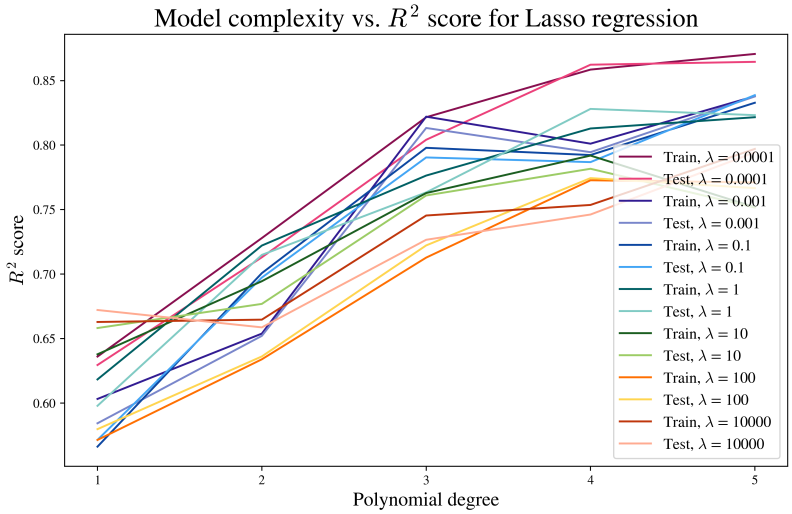

Figure 11: R^2 score as a function of model complexity for Lasso regression.

Table 2: R^2 score for the OLS method as well as Ridge and Lasso regression with $\lambda = 0.0001$.

Degree	R^2_{OLS}	R^2_{Ridge}	R^2_{Lasso}
1^{st}	0.673	0.668	0.630
2^{nd}	0.697	0.698	0.713
3^{rd}	0.757	0.840	0.804
4^{th}	0.858	0.851	0.862
5^{th}	0.903	0.850	0.865

Table 2 shows the R^2 score for the OLS method, as well as Ridge and Lasso regression for the λ-value that proved best from Figures 10 and 11. We observe that higher order polynomials provide better R^2 scores for all regression methods, but we also note that the OLS method for a 5^{th} degree polynomial is better than both Ridge and Lasso regression. This supports the fact that we need to analyse various statistical quantities in order to conclude on the best model.

Confidence interval

Table 3 shows the β-parameters and their 95% confidence interval for the OLS method, Ridge and Lasso regression. Since fitting of a 5^{th} degree polynomial provided the best results, the confidence intervals have been calculated with this fit. The range of each confidence interval describes possible values that the true β-value can be.

Predicted surfaces

Figure 12 shows the Franke function together with the prediction $\hat{\tilde{z}}$ from the OLS method, Ridge and Lasso regression. A 5^{th} degree polynomial has been fitted, and we have chosen $\lambda = 0.0001$ based on the previous analysis in this section. We recall that the prediction of $\hat{z}$ is calculated as

$$\hat{\tilde{z}} = \hat{X}\hat{\beta} \tag{24}$$

We see that the OLS method and Ridge regression perform well on the data, and are able to predict a function that is close to the analytical Franke function. In the Lasso regression surface, we recognise some of the shapes from the Franke function, but in general the prediction is not that close to the

Table 3: Confidence intervals for OLS, Ridge and Lasso regression. 5^{th} degree polynomials are fitted, and for Ridge and Lasso regression $\lambda = 0.0001$.

β_{OLS}	CI_{OLS}	β_{Ridge}	CI_{Ridge}	β_{Lasso}	CI_{Lasso}
0.439	(0.405, 0.473)	0.558	(0.510, 0.605)	0.496	(0.464, 0.528)
7.263	(6.941, 7.585)	8.256	(7.817, 8.695)	7.272	(6.892, 7.651)
3.427	(3.005, 3.848)	-1.165	(-1.669, -0.660)	1.912	(1.535, 2.288)
-34.689	(-36.273, -33.106)	-36.494	(-38.454, -34.534)	-32.810	(-34.677, -30.942)
-11.299	(-12.736, -9.861)	-15.599	(-17.323, -13.875)	-12.424	(-13.780, -11.068)
-15.375	(-17.368, -13.383)	12.606	(10.377, 14.835)	-9.975	(-11.788, -8.162)
53.522	(49.839, 57.204)	52.179	(47.870, 56.488)	47.058	(42.860, 51.256)
42.330	(39.494, 45.167)	51.692	(48.361, 55.023)	43.722	(40.835, 46.609)
14.668	(11.463, 17.873)	20.212	(16.626, 23.798)	16.420	(13.461, 19.379)
11.988	(7.583, 16.393)	-53.297	(-57.954, -48.640)	6.500	(2.509, 10.490)
-32.160	(-36.100, -28.219)	-26.791	(-31.242, -22.340)	-23.765	(-28.124, -19.407)
-51.441	(-54.447, -48.434)	-60.224	(-63.571, -56.877)	-53.054	(-56.227, -49.881)
-5.037	(-8.150, -1.924)	-12.836	(-16.229, -9.442)	-6.861	(-9.744, -3.978)
-27.508	(-30.907, -24.109)	-28.610	(-32.262, -24.958)	-27.645	(-30.844, -24.446)
8.744	(4.170, 13.318)	74.104	(69.493, 78.715)	7.807	(3.688, 11.927)
5.670	(4.100, 7.239)	2.533	(0.809, 4.257)	1.766	(0.068, 3.464)
16.790	(15.442, 18.139)	20.584	(19.139, 22.028)	18.997	(17.560, 20.434)
13.244	(11.833, 14.655)	13.929	(12.448, 15.410)	10.710	(9.411, 12.010)
-9.457	(-10.969, -7.946)	-5.020	(-6.621, -3.418)	-6.178	(-7.571, -4.785)
18.330	(16.828, 19.832)	16.344	(14.787, 17.900)	16.947	(15.518, 18.377)
-9.385	(-11.181, -7.589)	-32.930	(-34.670, -31.190)	-6.750	(-8.365, -5.135)

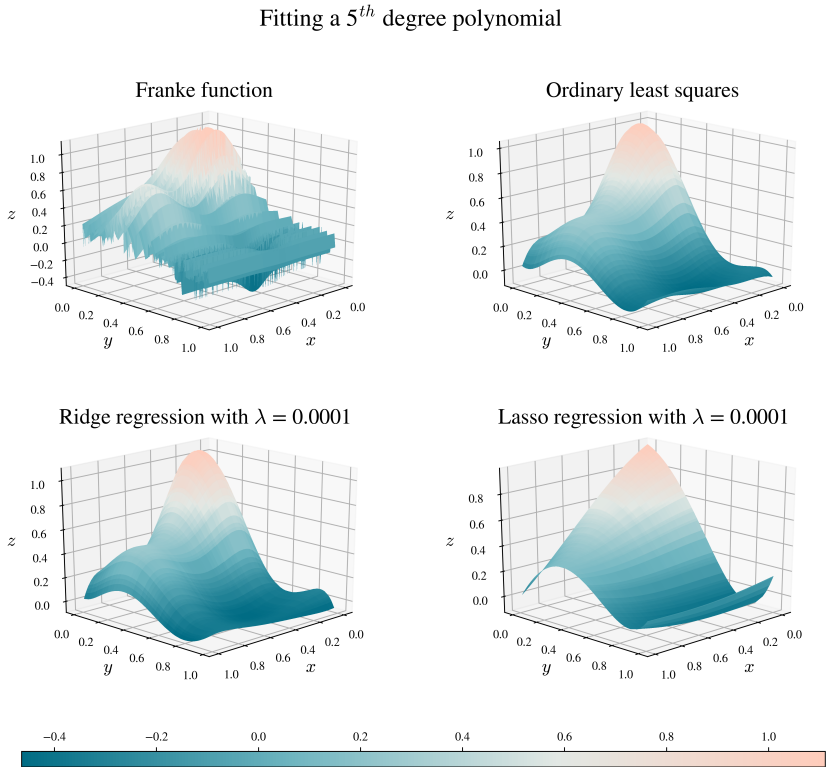

Figure 12: Analytical Franke function together with the prediction $\hat{\tilde{z}}$ for the OLS method, Ridge and Lasso regression when $\lambda = 0.0001$.

actual function. We also see that neither method predicts the noise (which is a good thing, as noise is something we generally do not want in our data). In order for our model to actually predict the noise, we would need to fit a polynomial of sufficiently large degree.

Figure 13 shows the Franke function together with the OLS method, as well as Ridge and Lasso regression for $\lambda = 1$. We see that the increase in λ-value causes the two latter models to perform worse on the data, as we have seen in the above analysis of the bias-variance tradeoff, mean squared error and R^2 score. The most interesting feature of this figure is the Lasso regression, which predicts the data as a flat surface. The reason for this is that all β-values are 0 as a consequence of the L_1 regularisation term in equation (11).

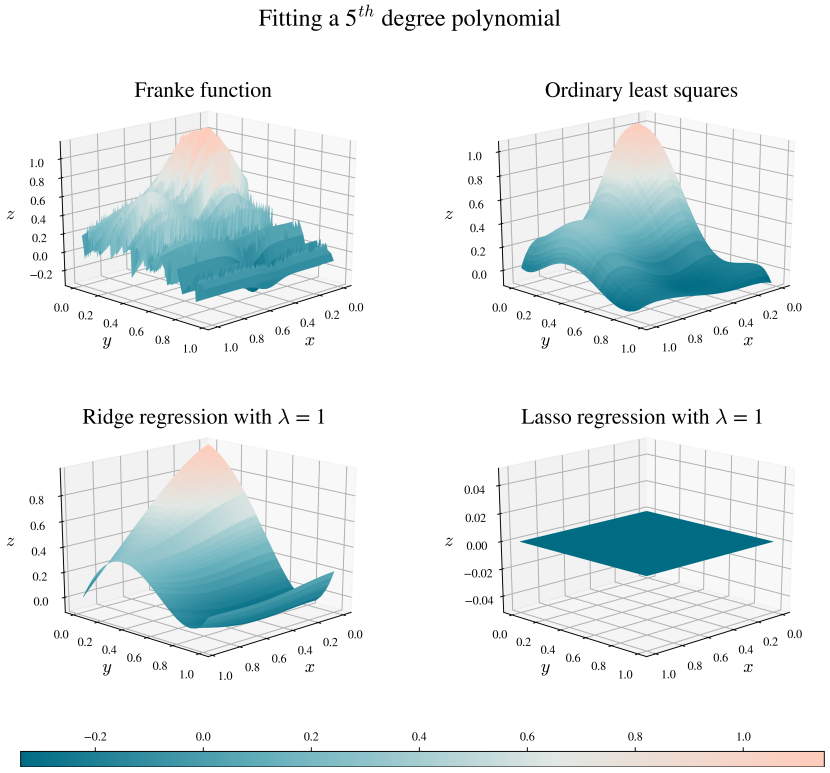

Figure 13: Analytical Franke function together with the prediction $\hat{\tilde{z}}$ for the OLS method, Ridge and Lasso regression when $\lambda = 1$.

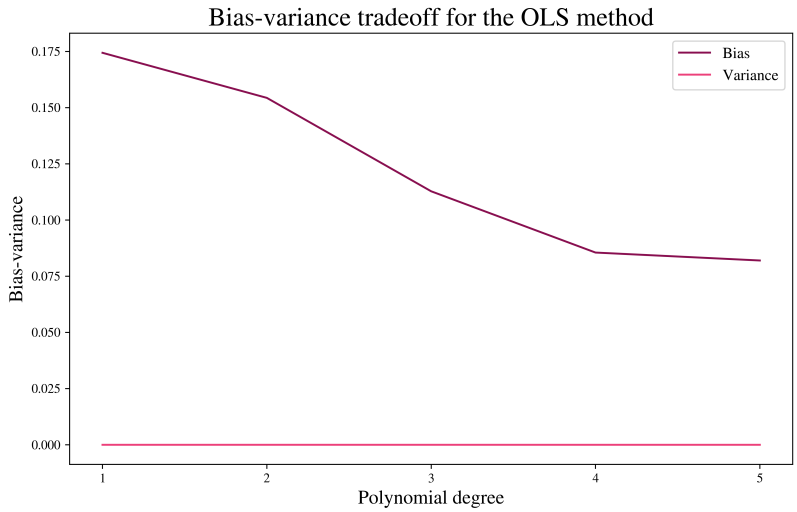

Figure 14: Bias-variance tradeoff as a function of model complexity for the Ordinary Least Squares method.

3.2 Real terrain data

The real terrain data is a large data set of dimension (3600×1800). Running such a large data set with 1000 bootstraps would take a large amount of time, hence we have reduced the data set to the dimension (200×200). Additionally, the terrain data contains large values. In order to obtain results that can be compared to the Franke function data, we have normalised the terrain data using the simple equation

$$\hat{z}_{\text{norm}} = \frac{\hat{z} - \hat{z}_{\text{min}}}{\hat{z}_{\text{max}} - \hat{z}_{\text{min}}} \tag{25}$$

Bias-variance tradeoff

Figure 14 shows the bias and variance as functions of model complexity for the OLS method. Again, we observe that the model is not complex enough to overfit the data. The variance is constant at ~ 0, and the bias decreases with increasing model complexity. The figure can further be used to comment on the behaviour of the mean squared error, where also this quantity will decrease as the polynomial degree increases.

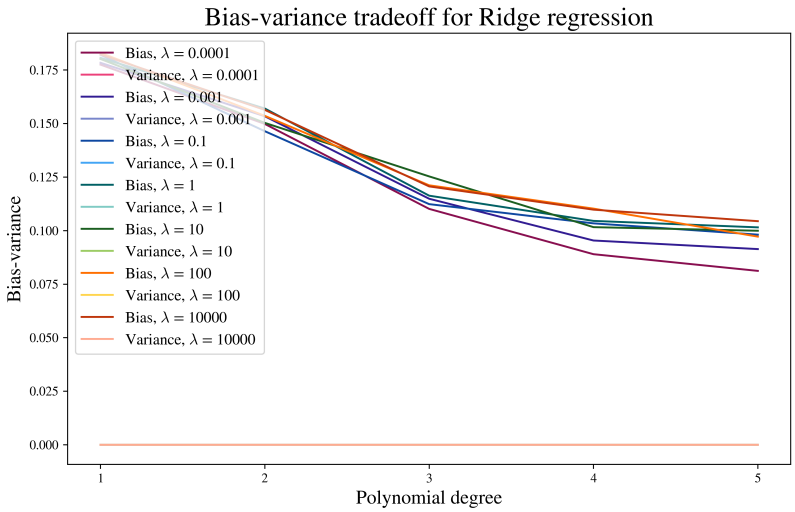

Figure 15: Bias-variance tradeoff as a function of model complexity for Ridge regression, where we have varied the hyperparameter λ.

Figures 15 and 16 show the bias and variance as functions of model complexity for the Ridge and Lasso regression, respectively. We see that the lowest value of λ provides the lowest bias for a 5^{th} degree polynomial, hence these are the hyperparameters that provide the best MSE. As compared to Figures 4 and 5, we see that the decreasing bias slope for the terrain data is more constant than in the generated Franke function data. A reason for this might be because of the noise in the Franke function data, but we are not confident that this is the case.

Mean squared error

Figure 17 shows the mean squared error as a function of model complexity for the OLS method. We see that the MSE decrease for increasing polynomial degree, as we also saw when analysing the mean squared error of the Franke function. The MSE is generally lower compared to Figure 6, where it converges to the minimum value around a 4^{th} order polynomial. However, this is not surprising as we are testing the model on a different data set.

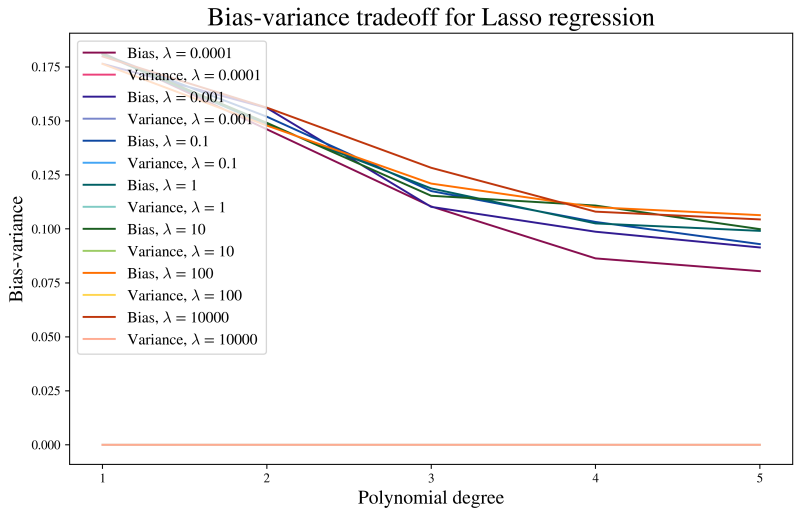

Figure 16: Bias-variance tradeoff as a function of model complexity for Lasso regression, where we have varied the hyperparameter λ.

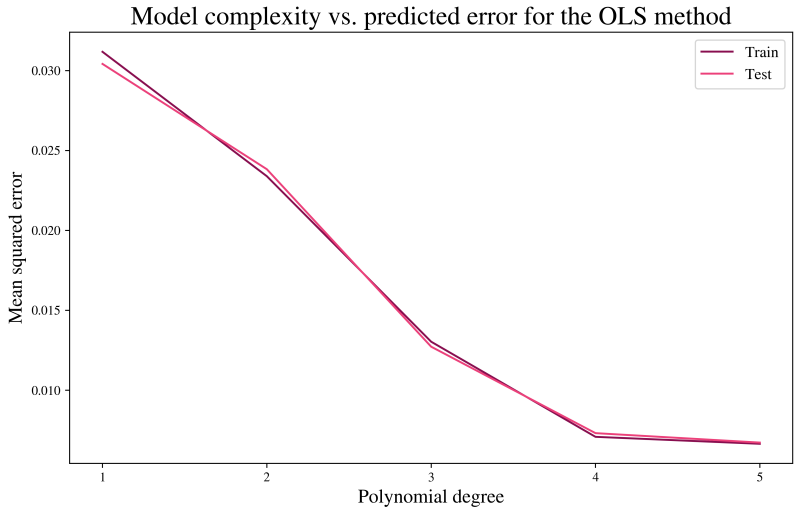

Figure 17: Mean squared error as a function of model complexity for the OLS method.

Figures 18 and 19 show the mean squared error as a function of model

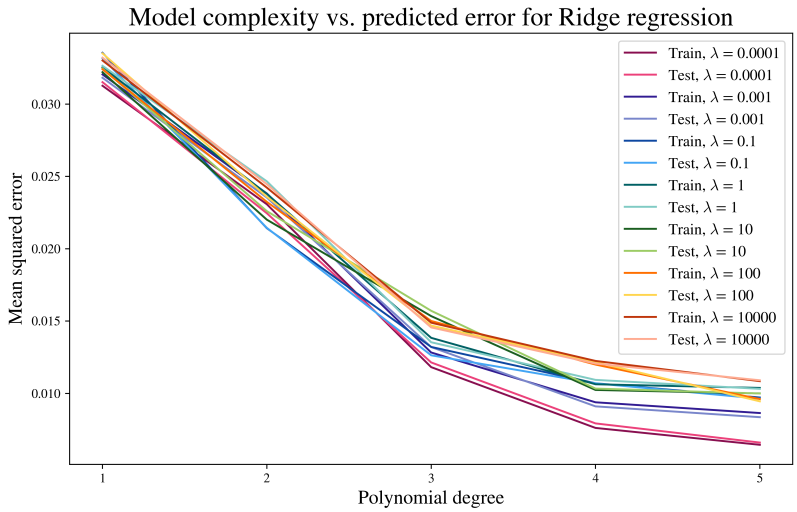

Figure 18: Mean squared error as a function of model complexity for Ridge regression. Different values of λ have also been tested.

complexity for Ridge and Lasso regression, respectively. Different values of the λ-parameter have been tested, where we clearly see that the lowest value of λ provides the lowest MSE for higher order polynomials. For a 4^{th} and 5^{th} degree polynomial, the difference in MSE for the various λ-values is not as apparent as with the Franke function data. Additionally, the MSE seems to decrease equally gradually with model complexity for all λ-values up until 3^{rd} degree polynomials. Then, the MSE for the lowest λ-value deviates from the rest.

Table 4 shows the mean squared error for the OLS method, Ridge and Lasso regression, with $\lambda = 0.0001$. Compared to Table 1, the MSE for the different regression methods are more similar to each other. We observe that the mean squared error of the Ridge and Lasso regression perform equally well to the OLS method. Based on the values of the table, it is difficult to say which method performs best of the terrain data. However, based on the results from the generated Franke function data, we have reason to suspect that the OLS method will perform best on the terrain data. After all, the variation in the MSE values is quite small.

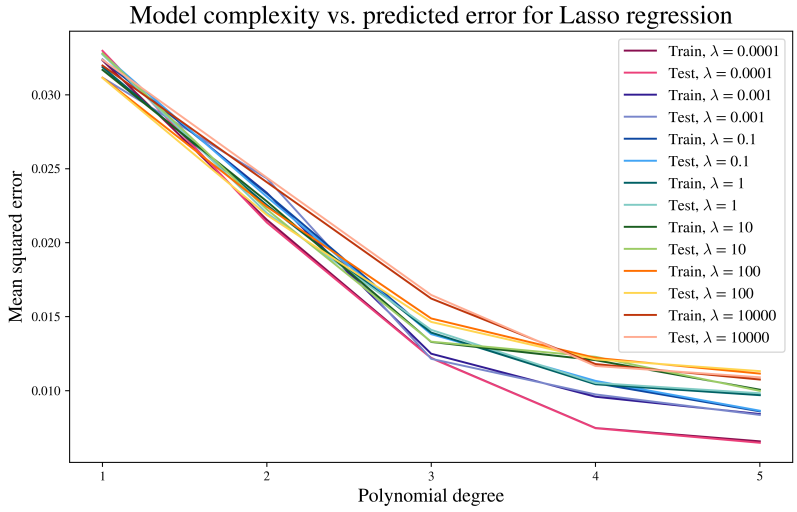

Figure 19: Mean squared error as a function of model complexity for Lasso regression. Different values of λ have also been tested.

Table 4: Mean squared error for the OLS method as well as Ridge and Lasso regression with $\lambda = 0.0001$.

Degree	MSE_{OLS}	MSE_{Ridge}	MSE_{Lasso}
1^{st}	0.030	0.032	0.033
2^{nd}	0.024	0.022	0.021
3^{rd}	0.013	0.012	0.012
4^{th}	0.007	0.008	0.007
5^{th}	0.007	0.007	0.006

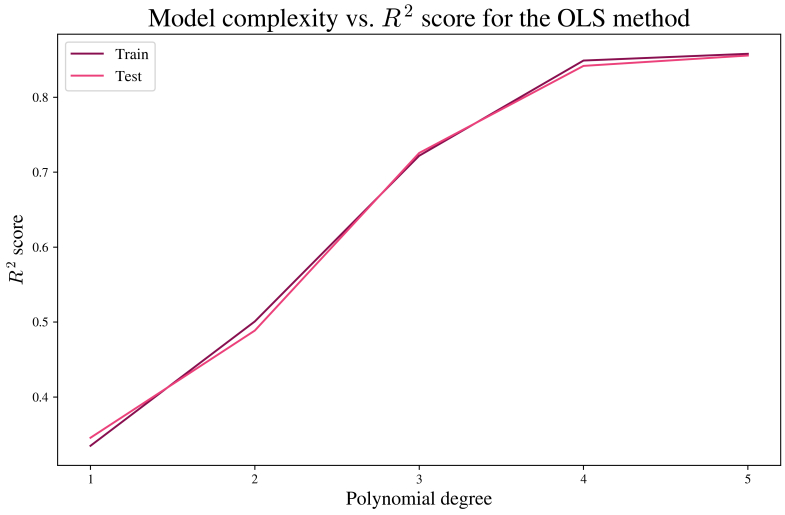

Figure 20: R^2 score as a function of model complexity for the OLS method.

R^2 score

Figures 20, 21 and 22 show the R^2 score as function of model complexity for the OLS method, Ridge and Lasso regression, respectively. The R^2 score increases with increasing model complexity, and we observe that when $\lambda = 0.0001$ the Ridge and Lasso regression methods are almost equal to the OLS method. As we have mentioned before, this is due to the fact that Ridge and Lasso regression boil down to the OLS method when $\lambda = 0$. The smallest λ-value is close to 0, hence the similarities. This is also seen in Table 5, where we have included the R^2 score for all three methods and used that $\lambda = 0.0001$. The table shows that for a 5^{th} degree polynomial, the Ridge method performs best. However, the difference in R^2 score for this polynomial degree is almost equal, and we suspect that the differences in the predictions are not that large.

Confidence interval

Table 6 shows the β-parameters and their 95% confidence interval for the OLS method, Ridge and Lasso regression. Again, we have fitted a polynomial of 5^{th} degree as this provides the best results based on the above analysis.

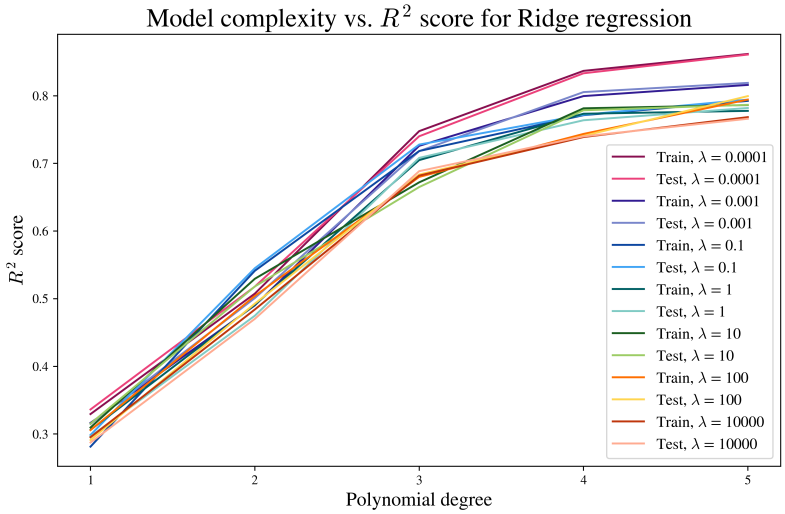

Figure 21: R^2 score as a function of model complexity for Ridge regression. Different values of λ have also been tested.

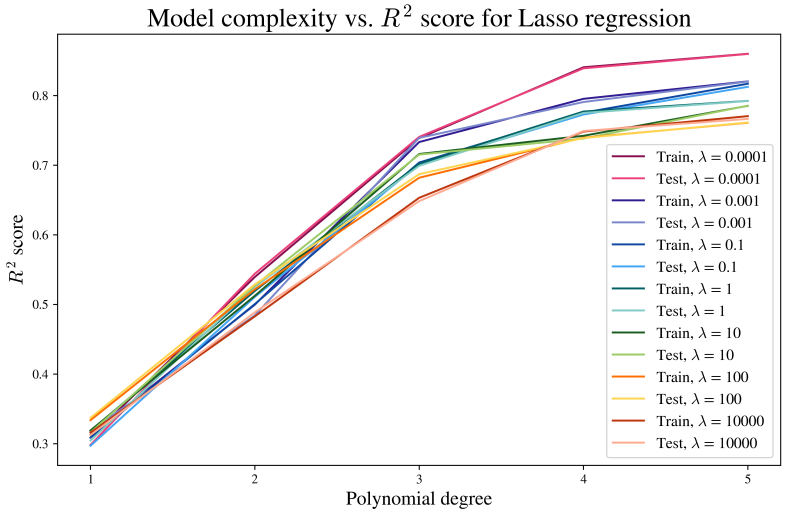

Figure 22: R^2 score as a function of model complexity for Lasso regression. Different values of λ have also been tested.

Table 5: R^2 score for the OLS method as well as Ridge and Lasso regression with $\lambda = 0.0001$.

Degree	R^2_{OLS}	R^2_{Ridge}	R^2_{Lasso}
1^{st}	0.346	0.336	0.298
2^{nd}	0.489	0.518	0.544
3^{rd}	0.726	0.740	0.741
4^{th}	0.842	0.833	0.839
5^{th}	0.856	0.861	0.860

Table 6: Confidence intervals for OLS, Ridge and Lasso regression. 5^{th} degree polynomials are fitted, and for Ridge and Lasso regression $\lambda = 0.0001$.

β_{OLS}	CI_{OLS}	β_{Ridge}	CI_{Ridge}	β_{Lasso}	CI_{Lasso}
0.860	(0.847, 0.874)	0.908	(0.890, 0.926)	0.870	(0.851, 0.889)
-1.388	(-1.538, -1.238)	-1.341	(-1.526, -1.156)	-1.017	(-1.214, -0.820)
0.349	(0.200, 0.497)	-0.407	(-0.589, -0.226)	-0.109	(-0.285, 0.067)
-1.298	(-2.016, -0.579)	-2.019	(-2.851, -1.186)	-3.100	(-3.995, -2.205)
-5.431	(-6.000, -4.862)	-4.242	(-4.900, -3.584)	-5.870	(-6.559, -5.180)
5.664	(4.945, 6.383)	8.948	(8.118, 9.779)	7.433	(6.665, 8.200)
16.253	(14.620, 17.886)	17.239	(15.453, 19.024)	19.505	(17.545, 21.464)
1.395	(0.154, 2.636)	1.866	(0.517, 3.214)	7.806	(6.380, 9.232)
19.211	(17.994, 20.428)	9.192	(7.844, 10.540)	6.351	(4.976, 7.727)
-36.465	(-38.084, -34.845)	-35.673	(-37.470, -33.877)	-27.781	(-29.419, -26.143)
-23.221	(-24.958, -21.485)	-22.783	(-24.595, -20.972)	-25.930	(-27.964, -23.895)
4.660	(3.296, 6.024)	1.475	(0.072, 2.877)	0.049	(-1.445, 1.542)
-20.213	(-21.458, -18.969)	-13.954	(-15.275, -12.632)	-24.756	(-26.126, -23.385)
-13.441	(-14.755, -12.128)	2.663	(1.247, 4.078)	15.261	(13.842, 16.680)
55.595	(53.903, 57.286)	43.521	(41.696, 45.347)	28.694	(27.029, 30.359)
8.657	(7.963, 9.352)	7.803	(7.104, 8.502)	9.298	(8.495, 10.101)
2.347	(1.726, 2.968)	5.138	(4.526, 5.749)	3.873	(3.209, 4.537)
1.389	(0.788, 1.990)	-2.340	(-2.951, -1.729)	2.858	(2.224, 3.491)
6.124	(5.540, 6.708)	4.739	(4.133, 5.345)	7.091	(6.468, 7.714)
4.428	(3.841, 5.015)	-4.236	(-4.858, -3.615)	-12.096	(-12.714, -11.478)
-25.422	(-26.084, -24.760)	-16.513	(-17.216, -15.811)	-8.307	(-8.950, -7.663)

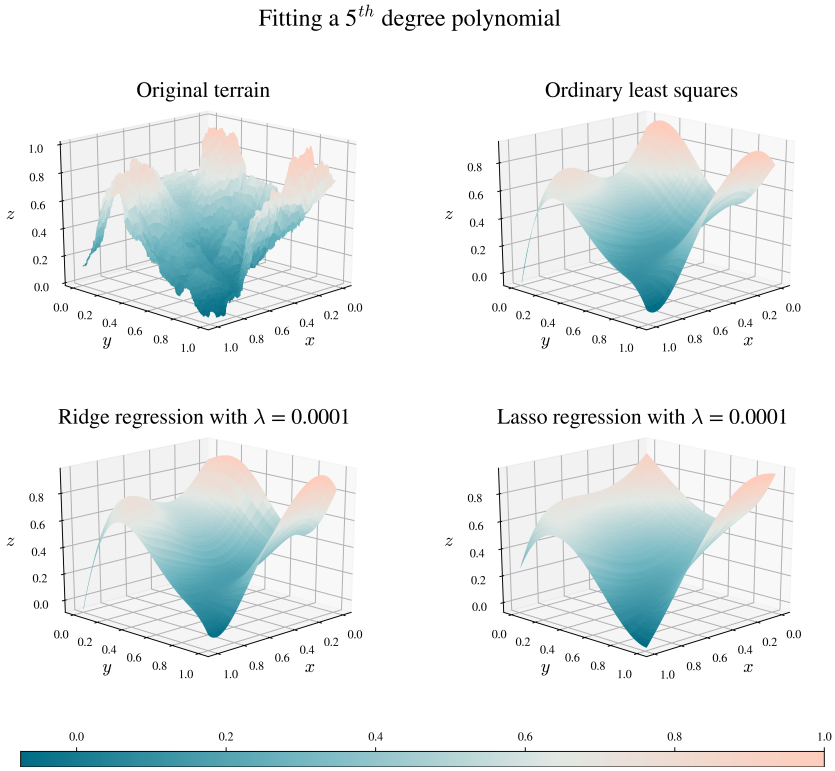

Figure 23: Real terrain data together with the prediction $\hat{\tilde{z}}$ for the OLS method, Ridge and Lasso regression.

Predicted surfaces

Figure 23 shows the real terrain data together with the prediction $\hat{\tilde{z}}$ from the OLS method, Ridge and Lasso regression. A 5^{th} degree polynomial has been fitted, where we have chosen $\lambda = 0.0001$ based on the above analysis. The predicted functions have been calculated using equation (24). We see that the predictions are similar in shape to the real terrain data, without the observed fine structure. The models performing OLS and Ridge regression provides similar results, where we clearly see that they are well fitted (at least as well as we can expect). The model using Lasso regression is not performing equally well on the data, but we can see clear shapes that are starting to resemble the other predicted data.

Figure 24 shows the Franke function together with the OLS methods, as well as Ridge and Lasso regression for $\lambda = 1$. Again, we see that the Lasso

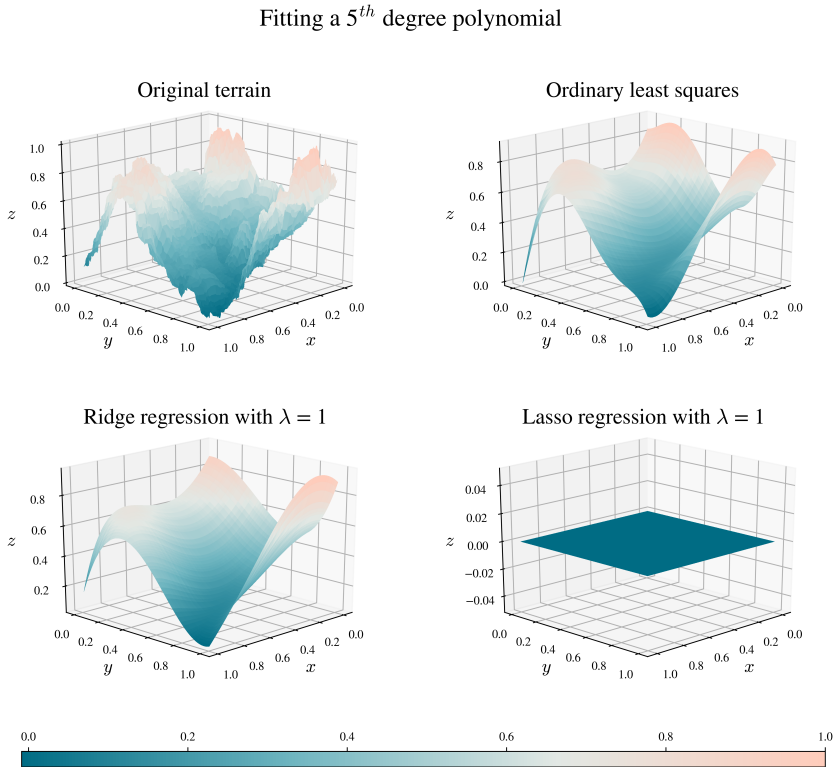

Figure 24: Analytical Franke function together with the prediction $\hat{\tilde{z}}$ for the OLS method, Ridge and Lasso regression when $\lambda = 1$.

regression model performs poorly on the data, eliminating the β-values as they become exactly 0. The Ridge regression model, however, is performing relatively well on the data when $\lambda = 1$. The predicted surface shows resemblance to the original terrain, but it is not as good as it is for lower values of λ.

4 Discussion

After doing statistical analysis on the different regression methods, we see that the OLS method performs best on both the Franke function data and the real terrain data. The Ridge method performs almost equally well, but given the fact that it requires a small λ-value, this is not difficult to understand. λ-values close to 0 will approach the OLS method, until the regression method

reduces down to the OLS method for $\lambda = 0$. With the given data sets, the OLS method is performing well enough for us to obtain good results. At least, this is the case when we fit 5^{th} degree polynomials. In the figures illustrating mean squared error and R^2 score as functions of model complexity, we see that as we approach the highest polynomial degree the graphs start to flatten. This indicates that the MSE and R^2 score converge to their minimum and maximum values as the model complexity increase. However, we know that eventually the bias will decrease and the variance will increase such that the model becomes overfitted. In this project, we could potentially fit a higher degree polynomial and obtain an even better MSE and R^2 score, without overfitting our model. However, the predicted surfaces in Figures 12 and 23 show that fitting a 5^{th} degree polynomial is sufficient in order to prove that our OLS (and also to some extent Ridge) model performs well on the data.

Going into this project, we thought that Lasso regression would produce the best results. The reason for this might be because this was the only method we did not write our own code to calculate, hence the expectations were high. However, after learning more about this particular regression method, we have realised that this method is actually most likely to produce the "worst" results. This is because of the constraint on the penalty term in equation (11). Increasing values of λ causes some of the β-coefficients to be exactly zero, hence they are eliminated. Smaller values of λ means that less β-coefficients are eliminated, which in turn gives us yet another reason to choose the smallest λ.

5 Conclusion

We have created three models using different regression methods to estimate the generated Franke function data and real terrain data. Statistical analysis tools have shown that the OLS method performs best on the data sets, but also that Ridge regression is relatively good when the hyperparameter λ is sufficiently small. We have shown that fitting of higher order polynomials provide better results in terms of the mean squared error and R^2 score. When fitting a 5^{th} degree polynomial, the models performing the OLS method and Ridge regression are well fitted to the analytical data.

Additional work on this subject would include fine tuning of the λ-parameter, i.e. test even smaller values to compare with the OLS method. It would also be interesting to test higher order polynomials, and see if we

could be able to observe overfitting with the given data sets. There are also a number of ways to calculate $\hat{\beta}$ (i.e. SVD), and it could be useful to explore how they perform compared to each other.

This project has been a good introduction to regression analysis and resampling methods, and has provided us with tools to use when we are soon delving into heavier machine learning topics.

6 Appendix

Relevant programs used to solve this project can be found at the GitHub address:

`https://github.com/hellmb/FYS-STK4155/tree/master/Project_1`

Please see the *README.md* file on how to run the code. The GitHub repository also contains a benchmark folder, where relevant quantities such as the β-values, mean squared error and R^2 score have been verified. The developed functions for calculating these quantities have been tested against the built-in functionalities of the Scikit-Learn library.

References

Glen, S.
2015. Lasso regression. `https://www.statisticshowto.datasciencecentral.com/lasso-regression/`. Accessed: 2019-04-10.

Hastie, T., R. Tibshirani, J. Friedman, and J. Franklin
2005. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85.

Sullivan, L.
. Confidence intervals. `http://sphweb.bumc.bu.edu/otlt/MPH-Modules/BS/BS704_Confidence_Intervals/BS704_Confidence_Intervals_print.html`. Accessed: 2019-04-10.

Wikipedia
2017. Bias-variance tradeoff. `https://en.wikipedia.org/wiki/Bias-variance_tradeoff`. Accessed: 2019-04-10.