Regression analysis and resampling methods

Daniel Pinjusic and Gudmund Gunnarsen

October 10, 2020

Abstract

In this project, we present different linear regression methods using polynomials in two variables as a model, and compare their predictions on the Franke function as well as terrain data from a region outside of Stavanger, Norway. We find that the OLS regression method yields the most accurate predictions for the Franke function for polynomials of degree $p < 11$, while Ridge regression gives good results up to and including $p = 25$. When using the different methods of regression on real terrain data, we find that the ridge method made good approximations, but it is unclear wheater or not the OLS and lasso methods is were good, due to conflicting results.

1 Introduction

In today's world the importance of data is bigger than ever, and so it is of great interest to develop methods that can shift through this data and find the relationships between seemingly independent sets of variables, something which would be very difficult for a human to do. With the help of machine learning methods, it is possible to train a computer to find these relationships, creating a model that can then be used for making predictions.

Consider for example the "Boston Housing Dataset"[1]. It contains 14 different attributes that are not seemingly related to each other, so drawing any type of conclusions about their dependability would be non-trivial, however, by using machine learning methods, it is possible to find the relationships between all of the attributes and even make predictions on future events, such as how future housing prices would change with a decreasing level of nitrogen oxide concentration, or how an increase in crime per capita would change the price of the property tax.

In this project, we explore linear regression, using Ordinary Least Squares, Ridge and Lasso regression on two sets of data. We compare the accuracy of these predictive methods and the dependence on factors such as presence and magnitude of noise, number of data points, model complexity, and hyperparameters. The two sets of data we use are the Franke function, were we will be generating a limited set of data and adding some noise, and terrain data from a region close to Stavanger, Norway. These two sets of data only have two variables, x and y coordinates, but depending on model selection, the complexity can easily be tweaked. As such, these data sets are convenient for the purpose of an introductory view of machine learning methods. The model used in this project is polynomials in x and y up to degree p.

First we will present basic theory around linear regression, introducing Ordinary Least Squares, Ridge and Lasso regression, as well as reviewing important concepts such as the Bias-Variance tradeoff and resampling techniques. Then we will explain the data sets

used, one generated by the Franke function and the other terrain data from somewhere near Stavanger, Norway. Next we will describe our model before finally presenting the results of our numerical analysis, comparing accuracy through the R^2 score, MSE, and confidence intervals before discussing our result.

2 Theory

Following is an introduction to the theory upon which the methods of linear regression we use in this project is derived. We follow calculations closely from the lecture notes found in [4], but restructure to fit this project.

2.1 Linear Regression

The basic problem to be solved in this project is that of fitting a function to data. Supposing that there is some data set $\boldsymbol{y}$ that has been measured with corresponding data points $\boldsymbol{x} = \{x_0, x_1, \ldots, x_{n-1}$, the objective is to find a relation between the data points and the data set through a function, such that other data points may be predicted. In this project we work with terrain height as the data set $\boldsymbol{y}$, which corresponds with data points $\boldsymbol{x}$ determining the coordinates of the given height.

The approach of linear regression for a continuous set of data is to assume there is a function $f(x)$ that determines the data points, but with some underlying noise ϵ such that $y_i = f_i + \epsilon_i$, for each point x_i. Then, linear regression makes the assumption that there is some approximation $\tilde{\boldsymbol{y}} = \mathbf{X}\boldsymbol{\beta}$ of $\boldsymbol{y}$, where $\mathbf{X}$ is called the design matrix. The design matrix is composed of various functions of the data set $\boldsymbol{x}$. It is an $n \times k$ matrix, where n is the number of data points and k is the number of regression parameters, contained in the vector $\boldsymbol{\beta}$. The regression parameters determine how the function depends on each element x_i, and the form of the design matrix elements depends on some outside understanding of the problem, which is referred to as the model.

2.2 Ordinary Least Squares

To fit the data with the least amount of error, we must define a function that determines the error. It is common to use the mean squared error or MSE as an estimate of the error of our approximation, defined as

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

This yields the mean of the square of the deviations between the true data $\boldsymbol{y}$ and the predicted data $\tilde{\boldsymbol{y}}$. For a more accurate prediction, the MSE will be smaller, and thus it is a good tool to determine model accuracy. Ordinary Least Squares regression uses the MSE as a straightforward cost function, a function that describes the error of our approximation, which we need to minimize with respect to the regression parameters $\boldsymbol{\beta}$. Thus the cost function is identical to the MSE,

$$C(\mathbf{X}, \boldsymbol{\beta}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = \frac{1}{n}(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}). \tag{2}$$

To find the optimal $\boldsymbol{\beta}$ that minimizes the cost function, we differentiate equation (2) with respect to $\boldsymbol{\beta}$ and find its root, that is finding

$$\frac{\partial}{\partial \boldsymbol{\beta}}(C(\mathbf{X}, \boldsymbol{\beta})) = 0$$

Reverting briefly to sum notation, we assume an $n \times p$ design matrix, such that there are p predictors and $\boldsymbol{\beta}$ is a vector of p elements. The minimization problem then reads

$$\begin{aligned} 0 &= \frac{\partial}{\partial \beta_j} \frac{1}{n} \sum_{i=0}^{n-1} (y_i - x_{i,0}\beta_0 - x_{i,1}\beta_1 - \dots x_{i,p-1}\beta_{p-1})^2 \\ &= -\frac{2}{n} \sum_{i=0}^{n-1} x_{i,j}(y_i - x_{i,0}\beta_0 - x_{i,1}\beta_1 - \dots x_{i,p-1}\beta_{p-1}), \end{aligned}$$

which in matrix notation reads

$$0 = -2\mathbf{X}^T(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}). \tag{3}$$

Solving for $\boldsymbol{\beta}$ yields $\hat{\boldsymbol{\beta}}$, the optimal $\boldsymbol{\beta}$ such that the MSE is minimized:

$$\hat{\boldsymbol{\beta}}_{\text{OLS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}\boldsymbol{y}. \tag{4}$$

With this $\hat{\boldsymbol{\beta}}$, one can produce the estimated values $\tilde{\boldsymbol{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$ that yield the lowest MSE for a given design matrix, for the specific points $\boldsymbol{y}$.

We are interested in how accurate the assessment of the regression parameters $\boldsymbol{\beta}$ are. In order to determine the confidence intervals, we need the variance of $\boldsymbol{\beta}$,

$$\text{Var}(\boldsymbol{\beta}) = \mathbb{E}\left[(\mathbb{E}[\boldsymbol{\beta}] - \boldsymbol{\beta})^T(\mathbb{E}[\boldsymbol{\beta}] - \boldsymbol{\beta})\right], \tag{5}$$

which requires the expectation values $\mathbb{E}[\boldsymbol{\beta}]$. For OLS we use equation (4) for

$$\begin{aligned} \mathbb{E}[\hat{\boldsymbol{\beta}}_{\text{OLS}}] =& \mathbb{E}[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{y}] \\ =& (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbb{E}[\boldsymbol{y}] \\ =& (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}}_{\text{OLS}} \\ =& \hat{\boldsymbol{\beta}}_{\text{OLS}}. \end{aligned}$$

Thus $\mathbb{E}[\hat{\boldsymbol{\beta}}_{\text{OLS}}] = \hat{\boldsymbol{\beta}}_{\text{OLS}}$, and this is referred to as OLS being an unbiased regression method. Now we can solve equation (5) for the case of OLS, which can be shown to be

$$\mathrm{Var}(\boldsymbol{\beta}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}.$$

This can be used to find the confidence intervals, for if we assume that $\boldsymbol{\beta}$ has a normal distribution, we can use [] $1.96\sqrt{\mathrm{Var}(\boldsymbol{\beta})}$

2.3 Model Accuracy

The MSE can be represented in another way, to illustrate what is called the Bias-Variance Tradeoff. Starting from the definition of MSE,

$$\begin{aligned}
\mathrm{MSE}(\boldsymbol{y},\tilde{\boldsymbol{y}}) =& \mathbb{E}[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2] \\
=& \mathbb{E}[(\boldsymbol{f}+\boldsymbol{\epsilon}-\tilde{\boldsymbol{y}})^2 \\
=& \mathbb{E}[(\boldsymbol{f}+\boldsymbol{\epsilon}-\tilde{\boldsymbol{y}}+\mathbb{E}[\tilde{\boldsymbol{y}}]-\mathbb{E}[\tilde{\boldsymbol{y}}])^2 \\
=& \mathbb{E}[(\boldsymbol{f}-\mathbb{E}[\tilde{\boldsymbol{y}}])^2]+\mathbb{E}[(\mathbb{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})^2]+\mathbb{E}[\boldsymbol{\epsilon}^2] \\
&+\mathbb{E}[2(\boldsymbol{f}-\mathbb{E}[\tilde{\boldsymbol{y}}])(\mathbb{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}}+\boldsymbol{\epsilon})]+\mathbb{E}[2(\mathbb{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}})\boldsymbol{\epsilon}] \\
=& \mathbb{E}[(\boldsymbol{f}-\mathbb{E}[\tilde{\boldsymbol{y}}])^2]+\mathbb{E}[(\tilde{\boldsymbol{y}}-\mathbb{E}[\tilde{\boldsymbol{y}}])^2]+\mathbb{E}[\boldsymbol{\epsilon}^2] \\
&+\mathbb{E}[\boldsymbol{\epsilon}]\,(2(\boldsymbol{f}-\mathbb{E}[\tilde{\boldsymbol{y}}])+2(\mathbb{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}}))+2(\boldsymbol{f}-\mathbb{E}[\tilde{\boldsymbol{y}}])\mathbb{E}[\mathbb{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}}] \\
=& \mathbb{E}[(\boldsymbol{f}-\mathbb{E}[\tilde{\boldsymbol{y}}])^2]+\mathbb{E}[(\tilde{\boldsymbol{y}}-\mathbb{E}[\tilde{\boldsymbol{y}}])^2]+\mathbb{E}[\boldsymbol{\epsilon}^2].
\end{aligned} \tag{6}$$

In the final line of equation (6), we use that $\mathbb{E}[\boldsymbol{\epsilon}] = 0$ since the noise is assumed to have a mean of 0, and that $\mathbb{E}[\mathbb{E}[\tilde{\boldsymbol{y}}]-\tilde{\boldsymbol{y}}] = 0$. If the noise had a nonzero mean, we could absorb it into $\boldsymbol{f}$. Finally we note that $\mathbb{E}[\boldsymbol{\epsilon}^2] = \mathbb{E}[(\boldsymbol{\epsilon}-\mathbb{E}[\boldsymbol{\epsilon}])^2 = \sigma^2$, the variance of $\boldsymbol{\epsilon}$, and $\mathbb{E}[(\tilde{\boldsymbol{y}}-\mathbb{E}[\tilde{\boldsymbol{y}}])^2] = \mathrm{Var}(\tilde{\boldsymbol{y}})$. With this, the MSE can be written as

$$\mathbb{E}[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2] = \mathbb{E}[(\boldsymbol{f}-\mathbb{E}[\tilde{\boldsymbol{y}}])^2]+\mathrm{Var}(\tilde{\boldsymbol{y}})+\sigma^2 \tag{7}$$

The first term of equation (7) is called the bias. This is how much the model varies from the underlying function values. For simple models, we expect a large bias, as the model is incapable of showing all the details of $\boldsymbol{f}$. The second term is the variance of the model itself, and this tends to increase for higher model complexity, as there are more regression parameters to take into account. Trying to fit them all will generally lead to overfitting, which means that the model is inaccurate when predicting values outside of the training data set. Finally, there is the third term, the variance of the noise itself.

Another way other than MSE to assess the accuracy of a prediction on a data set is the R^2 score function. It is defined as

$$\mathrm{R}^2(\boldsymbol{y},\tilde{\boldsymbol{y}}) = 1-\frac{\sum_{i=0}^{n-1}(y_i-\tilde{y}_i)^2}{\sum_{i=0}^{n-1}(y_i-\bar{y})^2}, \tag{8}$$

where $\bar{y}$ is the mean of the data points $\boldsymbol{y}$. The R^2 score function describes essentially how accurate the prediction is relative to a prediction which simply assumes the mean value for all possible data points. If $\mathrm{R}^2(\boldsymbol{y},\tilde{\boldsymbol{y}}) = 1$, then the prediction corresponds exactly with the true values, while if $\mathrm{R}^2(\boldsymbol{y},\tilde{\boldsymbol{y}}) = 0$ then the prediction is no better than the mean.

2.4 Ridge and Lasso regression

If the matrix $\mathbf{X}^T\mathbf{X}$ is singular, meaning its column vectors are linearly dependent, the matrix has no inverse and equation (4) will not be able to produce optimal regression parameters. This problem leads to the usage of Ridge regression. An ad-hoc solution is to add a small number λ along the diagonal of the matrix, such that

$$\hat{\boldsymbol{\beta}}_{\mathrm{R}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}\boldsymbol{y}, \tag{9}$$

approximating the solution while side stepping any singularities. This also corresponds with a different cost function

$$C(\mathbf{X}, \boldsymbol{\beta}, \lambda) = \frac{1}{n}(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}. \tag{10}$$

The value λ is referred to as a hyperparameter, and it serves a purpose beyond simply assisting in removing the singularity in that it can reduce degrees of freedom, by reducing the regression parameters that are already small. Minimizing equation (10) with respect to $\boldsymbol{\beta}$ yields equation (3) with an additional term,

$$0 = -2\mathbf{X}^T(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}) + 2\lambda\boldsymbol{\beta},$$

which when solved for $\boldsymbol{\beta}$ returns equation (9), with $\hat{\boldsymbol{\beta}}_{\mathrm{R}}$ as the optimal predictors of Ridge regression for a given λ. Ridge regression does not minimize the MSE for the given points $\boldsymbol{y}$, because of the added regularization term $\lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$. However, it is important to note that while OLS will by definition always find the smallest MSE for a given data set, the objective of linear regression is to use the given data set to predict other data sets. In the case of finding a continuous function like in this work, it could be that OLS produces large fluctuations between data points, when the true data might be an almost straight line between points. Thus, Ridge regression and other biased regression schemes might actually yield a lower MSE, but only on data that is not in the original training set.

A similar cost function could be defined as

$$C(\mathbf{X}, \boldsymbol{\beta}) = \frac{1}{n}(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda||\boldsymbol{\beta}||_1, \tag{11}$$

where

$$||\boldsymbol{x}||_1 = \sum_i |x_i| \tag{12}$$

is the norm-1 of the vector $\boldsymbol{x}$. equation (11) is the cost function that corresponds with Lasso regression [6]. It yields a non-analytic expression for $\hat{\boldsymbol{\beta}}$ and so has to be solved using iterative methods. We use the inbuilt functionality in the SciKit Learn [5] library using Python for Lasso regression in this work.

2.5 Resampling

A useful tool to more precisely and consistently assess model accuracy is resampling. Resampling means to make a new data set, consisting of parts of the total data set. By resampling, one can compare different parts of the data set, and thus discover information about how the model varies when changing the data. This is important as it is desirable to have a model that is stable with respect to differing data sets, as it will be more generally applicable. There are many types of resampling techniques, but we will look at and use two, bootstrap and k-fold cross validation.

Bootstrap is a resampling method introduced in 1979 by B. Efron [2]. The bootstrap method works as follows: In a set of randomly determined data points $X = \{x_1, x_2, \ldots, x_n\}$, one randomly draws elements into a new data set $X^* = \{x_1^*, x_2^*, \ldots, x_n^*\}$, where a given x_i might be represented multiple times in the X^* set. One then calculates the function of interest on this resampled data set, before repeating the process a number of times. The purpose is to find statistical properties of a function of the randomly determined data set X, but without knowing the probability density function on X. As an approximation, the bootstrap method assumes that the frequency of a given data point x_i in X represents the probability density function of X. Thus, through repeated resampling one can draw statistical knowledge about the function of X.

In the case of linear regression, one resamples the data points to produce a design matrix, and evaluates the optimal predictors for each resampling. With this, a better estimate for the error, bias and variance for example can be made, which can be especially important for smaller data sets.

One issue concerning resampling using the bootstrap method is that certain samples might randomly be picked out more than others, leading to those samples being overrepresented in the overall data and causing a bias in the overall model. A way to overcome this issue is using k-fold cross validation. K-fold cross validation means to split the data set into k different groups or folds, and then picking out individual folds to use as test data, while training the model on the remaining ones.

This has the advantage of utilizing all available data as test and training in equal amounts. Throughout these different permutations of the folds, one can validate the type of model run. In the case of Ridge and Lasso regression, for instance, various hyperparameters can be tested and the corresponding MSE can be compared, using all the available data and thus allowing for a more consistent result. The natural limit of k-fold cross validation is leave one out cross validation, where the number of folds equals the number of data points. This rules out all possible randomness in the validation, where every single data point but one is used to train the model and it is tested on the point left out, for each point in the data. However, this is very computationally intensive, as for every single data point there will be a unique fit of the model. In general, both bootstrap and k-fold cross validation are techniques that require a lot of computational power.

2.6 Numerical Test Data

In this project we look at two sets of data, one generated by the Franke function [3], and one set of terrain data. The Franke function is an exponential function in two dimensions that is convenient when testing interpolation and fitting algorithms,

$$
\begin{aligned}
f(x,y) =& \frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4}-\frac{(9y-2)^2}{4}\right)+\frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49}-\frac{(9y+1)}{10}\right) \\
&+\frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4}-\frac{(9y-3)^2}{4}\right)-\frac{1}{5}\exp\left(-(9x-4)^2-(9y-7)^2\right).
\end{aligned} \tag{13}
$$

This function can be Taylor expanded as a polynomial in x and y, and so we expect that modeling this function with lower order polynomials in x and y should be a good approximation.

The terrain data was downloaded from [4], but is originally from [7]. It is height data collected from a region outside of Stavanger, Norway.

2.7 Model

To model the 2-dimensional Franke function and the terrain data, the design matrix is produced as a polynomial fit using polynomials in x, y and the products of the two variables. For a given order p of a polynomial in two variables, the number of terms is $\frac{(p+1)(p+2)}{2}$, as for a given p, the polynomial expression of order $p+1$ will have $p+2$ new possible terms, $x^{p+1}, x^p y, x^{p-1}y^2, \ldots, y^{p+1}$, all of order $p+1$. For polynomials of order 0, that is a constant, there is only 1 term. Thus, the number of terms is the arithmetic sequence $1+2+\ldots+(p+1)$, the sum of which is given as above. In this project we define the row vectors in $\mathbf{X}$ as For this project we will be splitting the data in into training and testing sets. We will be using a 0.2 ratio, which means that training data will be 80% of all data, and the remaining 20% will be used as the testing set. We will also be scaling our data.

$$
\boldsymbol{x}^T = \left[1, x, y, x^2, y^2, \ldots, x^p, x^{p-1}y, \ldots, y^p\right] \tag{14}
$$

for a desired polynomial degree p.

3 Results

Here we present the results of our linear regression analysis of both the Franke function and the terrain data. For all results, unless stated otherwise, a precision of $n = 1000$ has been used.

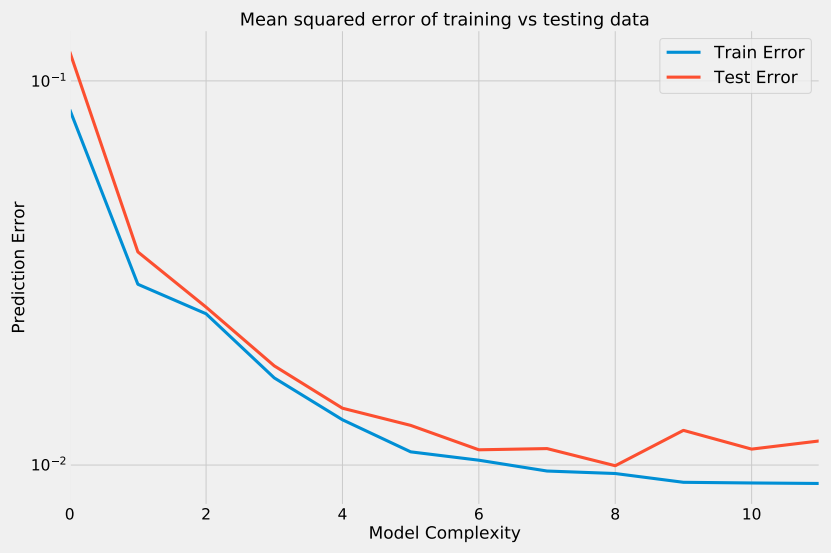

Figure 1: Plot showing the MSE error of testing data versus training data with increasing complexity. Data used is split from the Franke function.

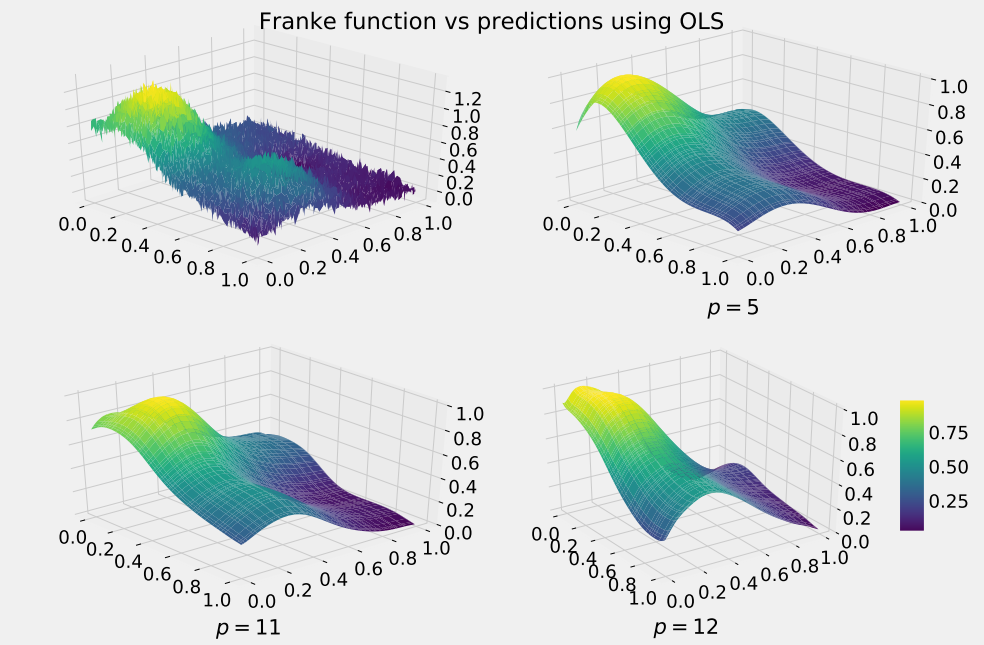

Figure 2: Franke function with noise compared with three predictions of OLS, with a degree 5, 11 and 12 polynomial fit. While there are some small differences between $p = 5$ and $p = 11$, $p = 12$ gives large errors.

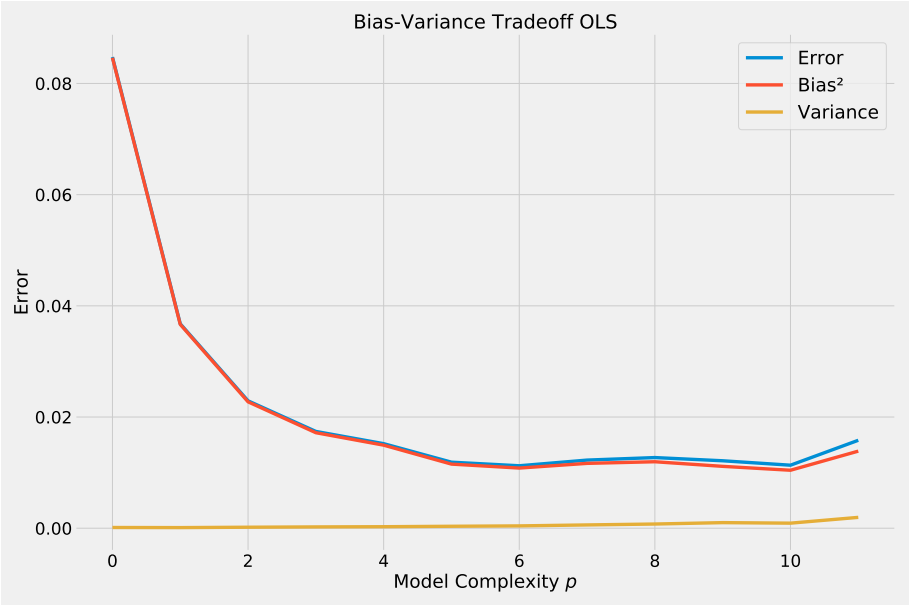

Figure 3: This plot shows the Bias-Variance Tradeoff for OLS on the Franke function, with bootstrap resampling and $n = 10000$. We see that as the model complexity p increases, the Bias lessens but the Variance eventually increases. For higher values of p, the error and variance increases by several orders of magnitude.

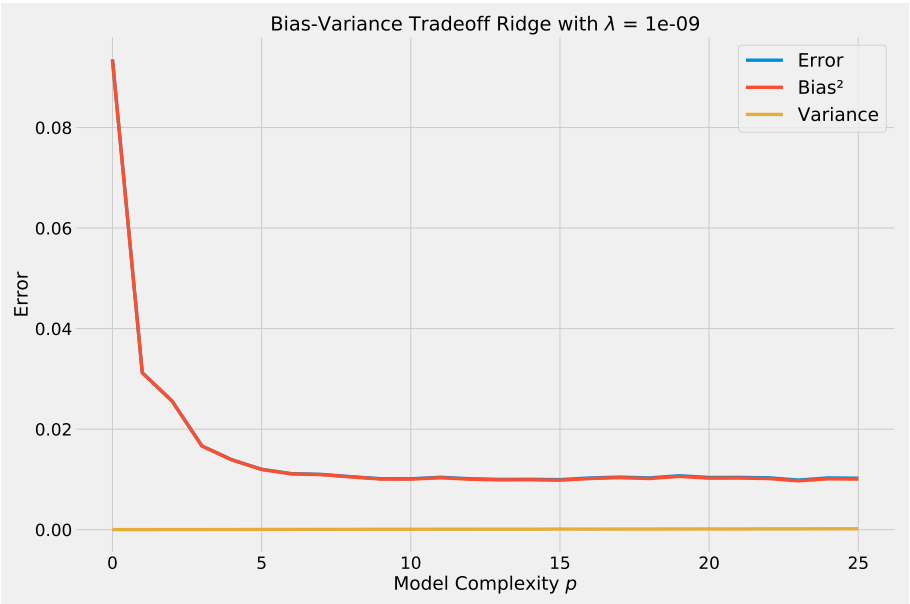

Figure 4: This plot shows the Bias-Variance Tradeoff for Ridge on the Franke function, with bootstrap resampling and $n = 10000$.

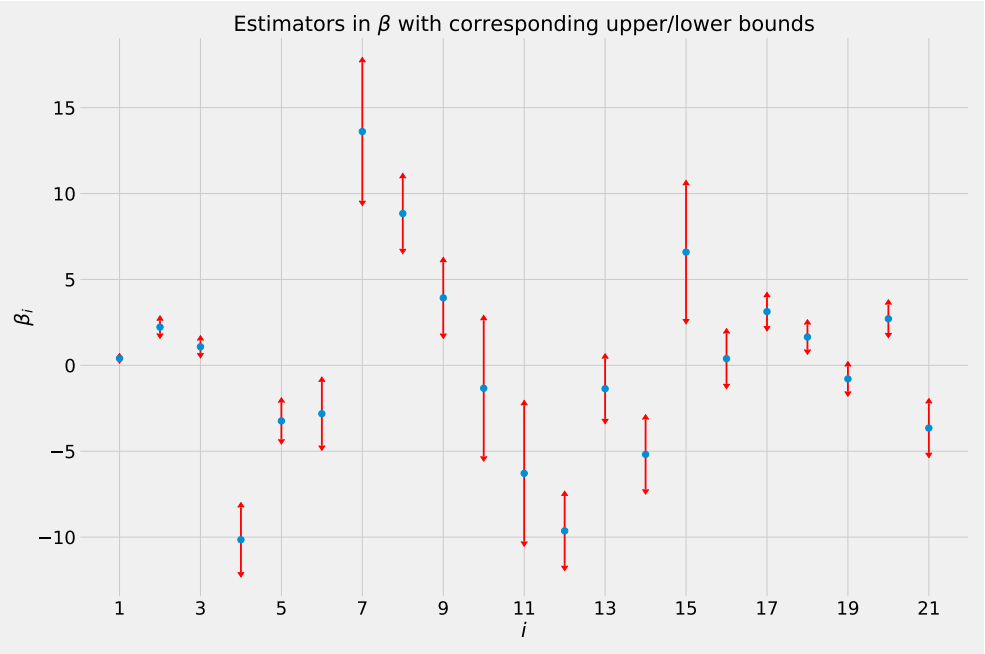

Figure 5: This plot shows the values of β with its corresponding confidence intervals.

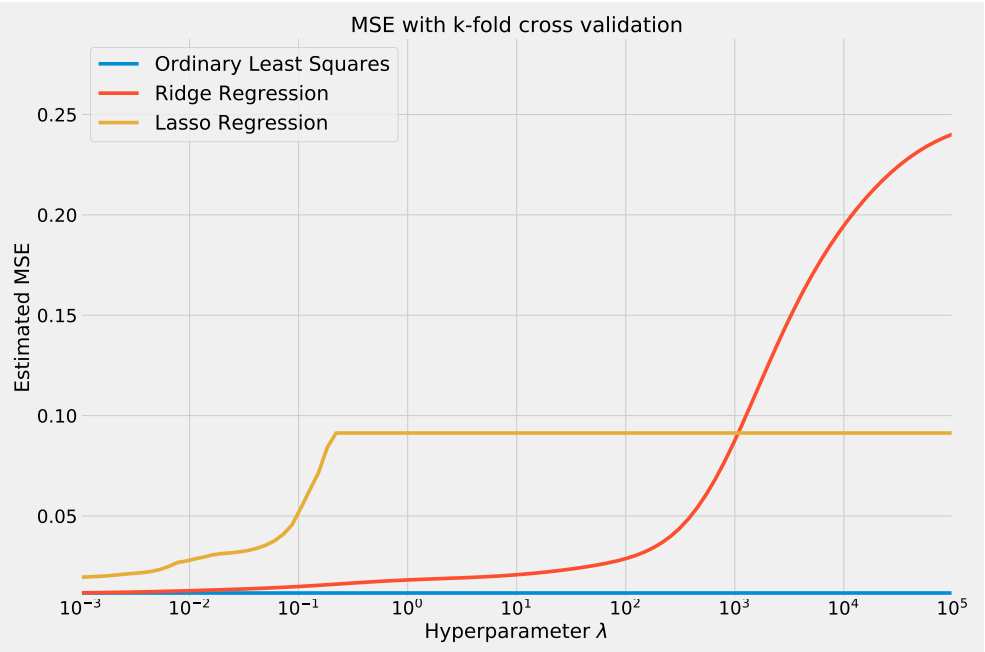

Figure 6: This plot shows the estimated MSE of OLS, Ridge and Lasso regression on the Franke function as a function of the hyperparameter λ. The model complexity is polynomials up to degree $p = 5$, and the estimation is done using k-fold cross validation with 5 folds. We see that OLS is the most accurate regardless of the hyperparameter.

4 Discussion

4.1 Model Complexity

In figure 1 we see the general trend that is expected, namely that the train error is lower than the test error for all polynomial degrees p, but that the test error begins increasing

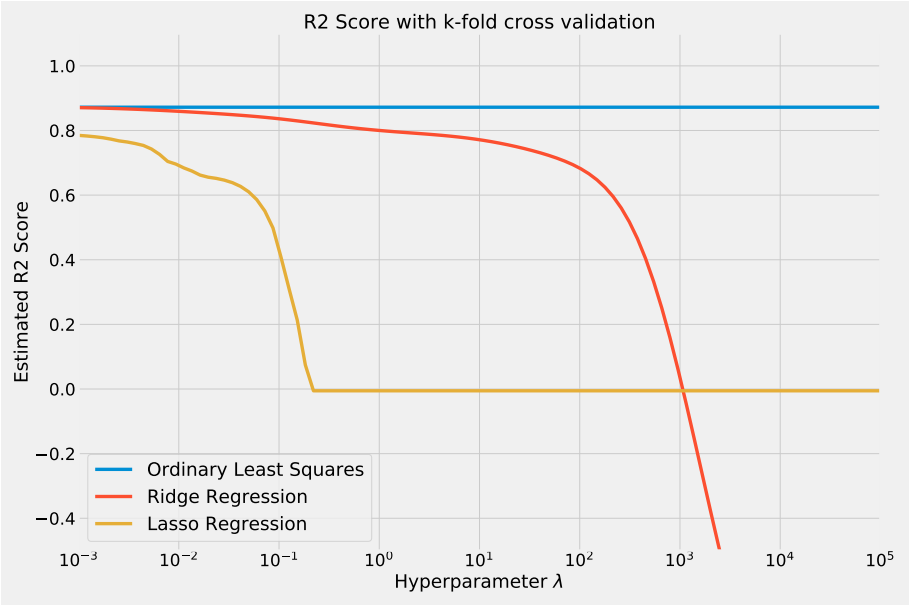

Figure 7: This plot shows the estimated R2 score of OLS, Ridge and Lasso regression on the Franke function as a function of the hyperparameter λ. The model complexity is polynomials up to degree $p = 5$, and the estimation is done using k-fold cross validation with 5 folds. We see that OLS is the most accurate regardless of the hyperparameter.

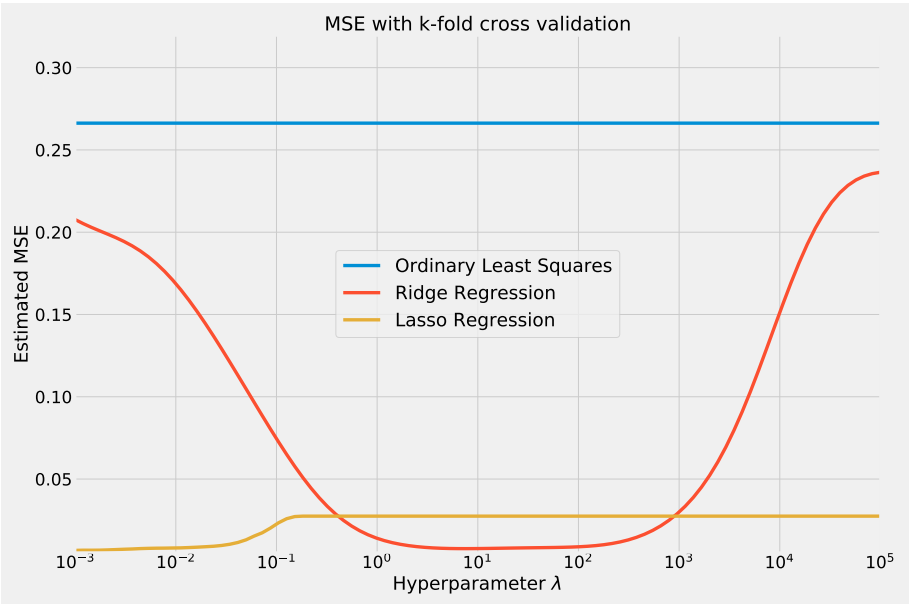

Figure 8: MSE of predicted terrain data.

for higher model complexity, as the variance increases and the model overfits to the training data.

The 3d plots of the Franke function and three OLS fits are plotted in figure 2. Here we can see that $p = 5$ and $p = 11$ seem to yield similar plots, while $p = 12$ becomes warped.

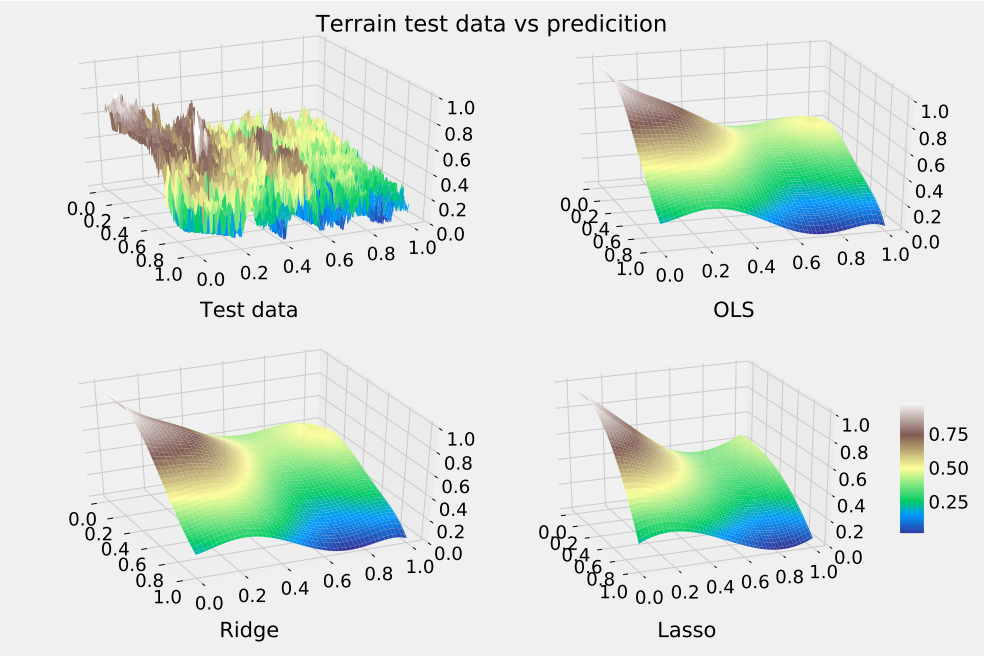

Figure 9: Terrain data compared with the predicted data using OLS, Ridge and Lasso for $p = 5$, ridge $\lambda = 10^-3$ and lasso $\lambda = 10^-4$.

This is reflected in the MSE calculations.

4.2 Bias Variance Tradeoff

As can be seen in figure 3, the bias largely dominates for the model complexity that we work with. However, at polynomial degree $p > 11$, the variance and bias begin increasing, and at $p = 12$ they both have increased by several orders of magnitude for the OLS model on the Franke function. Since this only occurs using OLS, and not with Ridge or Lasso regression, it might be related to issues in trying to invert $\mathbf{X}^T\mathbf{X}$, as Ridge and Lasso regression sidestep this issue by the hyperparameter λ. The higher the complexity, the more likely that the matrix $\mathbf{X}^T\mathbf{X}$ has linearly dependent vectors, and thus is singular and not invertible. Testing with very low values of the hyperparameter using Ridge regression, $\lambda \approx 10^{-9}$ produces variances that stay small even up to $p = 25$ as can be seen in figure 4. Setting an even smaller hyperparameter would be possible, but might also lead to loss of numerical precision. We should expect that in the limit of $\lambda \to 0$, Ridge regression returns results identical to OLS regression.

4.3 Cross validation

In figure 6 we show MSE as a function of the hyperparameter λ, for OLS, Ridge and Lasso. This plot was generated using k-fold cross validation, with 5 folds. Different numbers of folds were tested, but yielded similar results. In the case of the Franke function, OLS tended to give the lowest MSE, but only for model complexity $p < 12$.

In all cases it seems that the function for Lasso regression used in SciKit Learn [5] has some limiting function in place to prevent the R^2 score from dropping appreciably below zero. The Ridge regression MSE using our own code also flattens out at a certain λ,

but at a higher value than Lasso, and the R^2 score function drops to negative values, signifying it to be a worse approximation than simply using the mean value of the data points.

As for model complexity, higher complexity corresponded with substantially increased run time, and for OLS, the bias and variance grew by many orders of magnitude beyond $p = 11$. On the other hand, Ridge regression seems to flatten out and hardly gain any benefit in terms of error after about $p = 10$, as can be seen in figure 4.

4.4 Terrain Data

For the terrain data plotted in figure 9, we see that OLS and Ridge seem to yield best results, while Lasso is slightly less accurate. figure 8 however seems to contradict this, with OLS having a much higher MSE than both Lasso and Ridge. It might be issues with our code, but it might simply be that in figure 9 the model was trained on 80% of the entire terrain data set and tested on the remaining 20%, while in figure 8 we were using a small subset of 100×100 data points. Cross validation is a very time consuming process, and so running it on the approximately 6 million data points in the terrain data would be unfeasible.

5 Conclusion

The purpose of this project was to compare linear regression methods on the Franke function and real terrain data, using k-fold cross validation and bootstrap resampling methods to estimate the MSE and R2 score. Comparing the different models we see that OLS is the most accurate model for lower model complexity of degree $p < 11$, but that for higher complexity, Ridge regression with a small hyperparameter $\lambda \approx 10^{-9}$ ensures a small error even for higher degree polynomials. The terrain data gave unclear results, as there was some seeming disagreement between the cross validation and the visual representation. While the cross validation is quantitative and more trustworthy than visual approximation, the cross validation was only done on a small subset of the terrain data. Ridge regression seems to give good results in both cases.

Future work could include testing on other types of terrain data, and to further compare different hyperparameters using Ridge regression to see that the regression parameters are shrinked properly as they should. Finally, with more time

6 Appendix

Github repository with codes and figures can be found at `https://github.com/Gudmunhg/ML-projects`.

References

[1] URL: `https://www.kaggle.com/c/boston-housing/overview/description`. (accessed: 10.10.2020).

[2] B. Efron. "Bootstrap Methods: Another Look at the Jackknife". In: *Ann. Statist.* 7.1 (Jan. 1979), pp. 1–26. DOI: `10.1214/aos/1176344552`. URL: `https://doi.org/10.1214/aos/1176344552`.

[3] Richard Franke. *A Critical Comparison of Some Methods for Interpolation of Scattered Data*. 1979. URL: `https://calhoun.nps.edu/handle/10945/35052`.

[4] Morten Hjorth-Jensen. *FYS-STK3155/4155 Applied Data Analysis and Machine Learning*. URL: https://github.com/CompPhysics/MachineLearning. (accessed: 10.10.2020).

[5] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[6] Robert Tibshirani. "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288. ISSN: 00359246. URL: http://www.jstor.org/stable/2346178.

[7] USGS. *USGS*. URL: https://earthexplorer.usgs.gov/.