Regression analysis and resampling methods

FYS-STK4155: Project 1

Kim-Roger Grønmo
http://github.com/kimgronmo/fys-stk4155-p1

October 11, 2022

Abstract

We investigate the suitability of several regression methods for building models to reproduce data from the Franke function and also real world terrain data. The methods used include Ordinary Least Squares (OLS) regression, Ridge regression and Lasso regression. Varying degrees of polynomials are used to fit the data in our models. We also use techniques such as bootstrap and cross-validation to further assess these models. In particular we investigate the bias-variance trade off as a function of model complexity.

We find that the Franke function is best represented using OLS regression with cross validation, as it has the lowest mean square error(mse) at 0.005914 versus 0.0062 for Ridge regression and 0.01131 for LASSO. The real world data terrain set is better represented using OLS with a mse at 0.000507, versus 0.00055 for Ridge and 0.000923 for LASSO regression.

Our investigation reveal that the results from the Franke function is most likely due to our predicted data being situated in a region where bias dominates the mean square error, and we would need to use more complex models than the ones currently being investigated to better fit the data sets to the models.

1 Introduction

We are living in a world with abundant access to data. As our ability to collect and visualize this data increases, our inherent need to understand which causes and effects lies behind what we observe in the real world become more apparent. With the advent of both modern computers and mobile technology we have the tools needed to both collect and analyze the data generated by the real world.

Imagine that we wanted to calculate the average annual salary for a person. It is quite apparent that there a great many factor that influence what a person earns. A few example can be age, gender, level of education, degree earned, years of work experience, the general job market within the given field and so on. In order to try to model these kind of situations with statistics one often chooses linear regression methods.

A basic premise of regression modelling is that the observations we collect, the response, is dependent on some predictor variables. For linear regression models it is assumed that there is a linear relationship between these predictor variables and the responses they generate. We then have the ability to make models which uses this linearity to make a design matrix to predict the effect these factors have on the response variable. We can use linear algebra to use such a design matrix in an optimal combination with the given factors to generate responses that predict real world obseervations. The ability to minimize the error between these predictions and observations using modern computers make Linear Regression Models popular.

In this report we will study the Franke function with an added noise term using regression models. We will also study real world terrain data using the same models and methods. We start by giving an introduction to the statistical theory and methods we will apply. We will then show some of our results and conclude with a discussion of them. The report ends with a bibliography. Note that there is no appendix with the codes being used as they are too large to fit comfortably into such a report. An interested reader can download both this report and supporting code and data set from the following github address:

http://github.com/kimgronmo/fys-stk4155-p1

The github repository includes a test run file with results printed to screen, and some selected figures in the Results folder.

2 Theory

The general references for this section are reference [1] and [2].

For the least squares method we are evaluating a situation in which p characteristics of n samples are measured. The *response* is denoted $\mathbf{y}$: a vector with size n. The measured characteristics, denoted the predictors, we will organize in a matrix $\mathbf{X}$ of size $n \times p$. This matrix is referred to as the *design matrix.*

In order to explain the relationship between the response and the predictor variables we will use a function $\mathbf{y}(\mathbf{X})$. When there is a linear relationship between $\mathbf{X}$ and $\mathbf{y}$ then the single response can be written as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{1}$$

where $\boldsymbol{\varepsilon}$ is the deviation of the linear model $\mathbf{X}\boldsymbol{\beta}$ and the response $\mathbf{y}$. The error $\boldsymbol{\varepsilon}$ is assumed to have a normal distribution, $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2)$. $\boldsymbol{\beta}$ is a vector containing the linear regression coefficients β_i. In our model we denote the generated response variables as $\tilde{\mathbf{y}}$, Then

$$\tilde{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta} = \mathbf{y} - \boldsymbol{\varepsilon}.$$

For our model we want to calculate $\boldsymbol{\beta}$ in such a way that the error $\boldsymbol{\varepsilon}$ gets minimized.

The expectation value of $\mathbf{y}$ for a given element i is then calculated by

$$\mathbb{E}(y_i) = \mathbb{E}(\mathbf{X_{i,*}}\boldsymbol{\beta}) + \mathbb{E}(\epsilon_i) = \mathbf{X_{i,*}}\boldsymbol{\beta} \tag{2}$$

and its variance is then calculated as

$$\begin{aligned} Var(y_i) = \mathbb{E}([y_i - \mathbb{E}(y_i)]^2) = \mathbb{E}(y_i^2) - [\mathbb{E}(y_i)]^2 \\ = \mathbb{E}([\mathbf{X_{i,*}}\boldsymbol{\beta} + \epsilon_i]^2) - (\mathbf{X_{i,*}}\boldsymbol{\beta})^2 \\ = \mathbb{E}([\mathbf{X_{i,*}}\boldsymbol{\beta}]^2 + 2\epsilon_i\mathbf{X_{i,*}}\boldsymbol{\beta} + \epsilon_i^2) - (\mathbf{X_{i,*}}\boldsymbol{\beta})^2 \\ = (\mathbf{X_{i,*}}\boldsymbol{\beta})^2 + 2\mathbb{E}(\epsilon_i)\mathbf{X_{i,*}}\boldsymbol{\beta} + \mathbb{E}(\epsilon_i^2) - (\mathbf{X_{i,*}}\boldsymbol{\beta})^2 \\ = \mathbb{E}(\epsilon_y^2) = Var(\epsilon_i) = \sigma^2 \end{aligned} \tag{3}$$

For the parameters $\boldsymbol{\beta}$ we also have

$$\mathbb{E}(\hat{\boldsymbol{\beta}}) = \mathbb{E}((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}) = \boldsymbol{X}^T\boldsymbol{X}^{-1}\boldsymbol{X}^T\mathbb{E}(\boldsymbol{Y}) = \boldsymbol{X}^T\boldsymbol{X}^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{\beta} \tag{4}$$

The variance of $\boldsymbol{\beta}$ is

$$\begin{aligned} Var(\hat{\boldsymbol{\beta}}) = \mathbb{E}([\boldsymbol{\beta} - \mathbb{E}(\boldsymbol{\beta})][\boldsymbol{\beta} - \mathbb{E}(\boldsymbol{\beta})]^T) \\ = \mathbb{E}([(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y} - \boldsymbol{\beta}][\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y} - \boldsymbol{\beta}]^T) \\ = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}(\boldsymbol{Y}\boldsymbol{Y}^T)\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \\ = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T + \sigma^2)\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \\ = \boldsymbol{\beta}\boldsymbol{\beta}^T + \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T = \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1} \end{aligned} \tag{5}$$

where $\mathbb{E}(\boldsymbol{Y}\boldsymbol{Y}^T) = \boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T + \sigma^2\boldsymbol{I}_{nn}$. The expression for the variance of $\boldsymbol{\beta}$ can the be used to define a confidence interval for each of the parameters β

2.1 The Design Matrix

In our models we considered different sizes of the design matrix. For a model with polynomial 2 we use two predictors—we will denote them x and y—with the response y. With the intercept included this can be written as:

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_2 & y_2 & x_2y_2 \\ \vdots & \vdots & \vdots & \\ 1 & x_n & x_n & x_ny_n \end{bmatrix}$$

2.2 Ordinary Least Squares

To find the optimal parameters β_i we define a cost function which gives a measure of the spread between the values y_i and the parameterized values $\tilde{y}_i$:

$$C(\boldsymbol{\beta}) = \frac{1}{n}\left\{(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\right\}. \tag{6}$$

In order to find the parameters β_i we will then minimize the spread of $C(\boldsymbol{\beta})$, which means we will aim to solve the problem

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \frac{1}{n}\left\{(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\right\}. \tag{7}$$

which will require

$$\frac{\partial C(\boldsymbol{\beta})}{\partial \beta_j} = \frac{\partial}{\partial \beta_j}\left[\frac{1}{n}\sum_{i=0}^{n-1}\left(y_i - \beta_0 x_{i,0} - \beta_1 x_{i,1} - \beta_2 x_{i,2} - \cdots - \beta_{n-1} x_{i,n-1}\right)^2\right] = 0 \tag{8}$$

This can be written in matrix form as

$$\frac{\partial C(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0 = \boldsymbol{X}^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}). \tag{9}$$

which leads to

$$\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}, \tag{10}$$

then if the matrix $\boldsymbol{X}^T\boldsymbol{X}$ is invertible we have the solution

$$\boldsymbol{\beta} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{11}$$

2.3 Ridge and LASSO regression

If our design matrix $\boldsymbol{X}$ has linearly dependent column vectors, we will not be able to compute the inverse of $\boldsymbol{X}^T\boldsymbol{X}$ and we cannot find the parameters β_i. A way to solve this problem is to add a small diagonal component to the matrix to invert

$$\boldsymbol{X}^T\boldsymbol{X} \to \boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I},$$

where $\boldsymbol{I}$ is the identity matrix. The parameter λ is called a hyperparameter. This is used to adjust the amount of the coefficient shrinkage. We can use the parameter λ to shrink or increase the role of a given parameter β_j. The best lambda for our data set will be the lambda that minimize the cross-validation prediction errors.

The new cost function to be optimized is then

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

which leads to the Ridge regression minimization problem where we require that $||\boldsymbol{\beta}||_2^2 \leq t$, where t is a finite number larger than zero. The norm-2 vector is defined as

$$||\boldsymbol{x}||_2 = \sqrt{\sum_i x_i^2}.$$

By defining

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

we have a new optimization equation

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

which leads to Lasso (least absolute shrinkage and selection operator) regression. The norm-1 is defined as

$$||\boldsymbol{x}||_1 = \sum_i |x_i|.$$

It can then be shown (see reference [1]) that the best fits are given by:

$$\boldsymbol{\beta}_{\text{Ridge}} = (\mathbf{X}^T\mathbf{X}+\lambda\boldsymbol{I})^{-1}\mathbf{X}^T\mathbf{y}. \tag{12}$$

For LASSO regression we used the built in functions of Scikit-Learn and thus did not solve for the optimal $\boldsymbol{\beta}$ manually

2.4 The bias variance trade-off

We found the parameters $\boldsymbol{\beta}$ by optimizing the mean squared error via the cost function

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

By using a more compact notation

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

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

This is now rewritten in terms of the bias, the variance of the model $\tilde{\boldsymbol{y}}$ and the variance of $\boldsymbol{\epsilon}$.

$$\mathbb{E}\left[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2\right] = \mathbb{E}\left[(\boldsymbol{y}-\mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2\right] + \mathrm{Var}\left[\tilde{\boldsymbol{y}}\right] + \sigma^2,$$

Which can then be rewritten as

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

The expected prediction error consist of the sum of the irreducible error (variance of the new test target) and the mean square error. The mean square error is broken up into a bias component and variance component. As the model complexity increases the variance tend to increase and squared bias decreases.

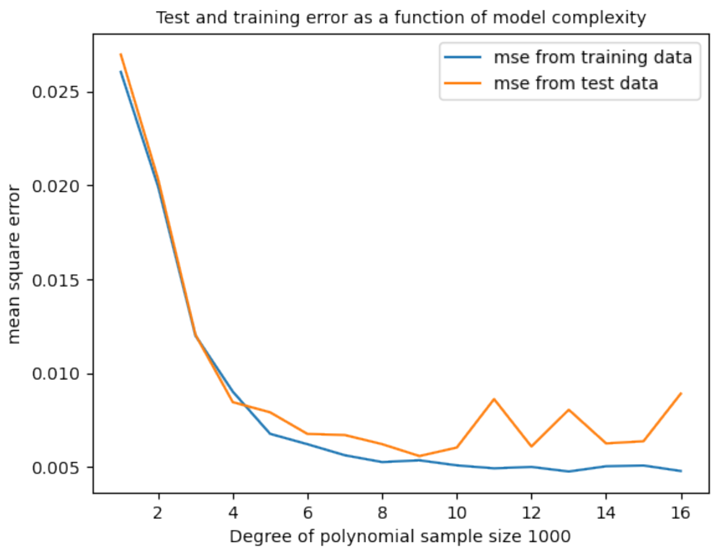


Figure 1: Figure 2.11 based on data from the Franke function

The variance refers to the amount by which our model would change if we estimated it using a different training data set. Different training data sets will result in a different estimate. If a method has high variance then small changes in the training data can result in large changes in the model. More flexible statistical methods have higher variance in general. This can lead to a higher chance that a flexible model will overfit the training data, i.e. it learns the noise of the training data.

We can observe this effect in the given Figure 1 . Note that when the sample size increases this effect is lowered.

3 Methods

3.1 Resampling methods

The bootstrap method is a statistical technique for estimating quantities about a population by averaging estimates from multiple smaller data samples. These samples are constructed by drawing observations with replacement from our training set.

The cross validation method splits up a data set into several mutually exclusive subsets. One of these subsets will in turn take the role of test set and the remaining union of subsets constitutes the training set.

Both these methods were implemented using tools from the sklearn library.

These methods were used to give better estimates to the mean square error, bias and variance for several model complexities. Further results are given in the github adress: http://github.com/kimgronmo/fys-stk4155-p1

3.2 The Franke function

The Franke function $f(x, y)$ is defined by:

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

The data set is generated using this function with an added stochastic noise term using the normal distribution $N(0, 1)$.

3.3 Data preprocessing

In order to make our data sets suitable for the algorithms we want to use, we must do some data preprocessing. The features will be scaled in such a way to avoid outlier values. When we do this the accuracy and error analysis will be less sensitive to extreme data values.

Mathematically, this involves subtracting the mean and divide by the standard deviation over the data set, for each feature:

$$x_j^{(i)} \to \frac{x_j^{(i)} - \overline{x}_j}{\sigma(x_j)},$$

where $\overline{x}_j$ and $\sigma(x_j)$ are the mean and standard deviation, respectively, of the feature x_j. This ensures that each feature has zero mean and unit standard deviation. In our data sets we set the standard deviation to 1 and scaled the data by subracting the mean using numpy.

3.4 Error Analysis

A common way to check how close a predicted response value is to the observed value is by calculating the Mean Square Error (MSE). It is defined by:

$$\mathrm{MSE}(\hat{y}, \hat{\tilde{y}}) = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \tilde{y}_i)^2, \tag{14}$$

How well future samples are likely to be predicted by the model can be measured by R^2, the coefficient of determination, where the best possible score is 1.0

$$R^2(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = 1 - \frac{\sum_{i=0}^{n-1} (y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n-1} (y_i - \bar{y})^2}, \tag{15}$$

where the mean value of $\boldsymbol{y}$ is defined by

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

4 Results and Discussion

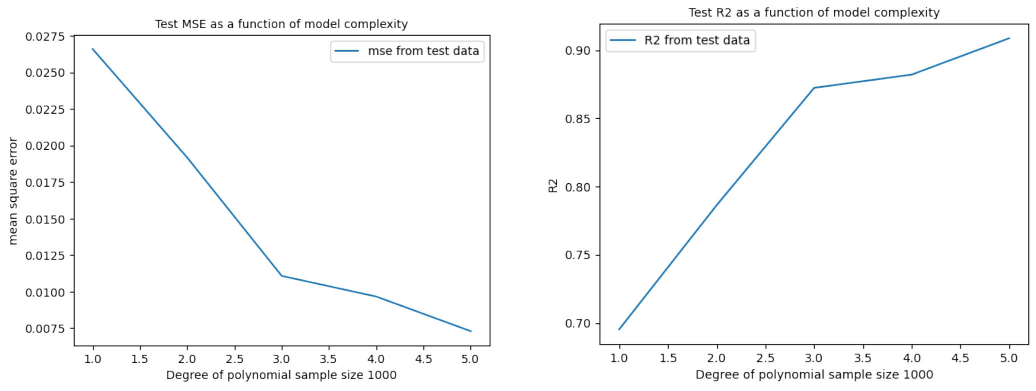


Figure 2: MSE for test data(left) and R2 scores for test data(right) versus degree of polynomial

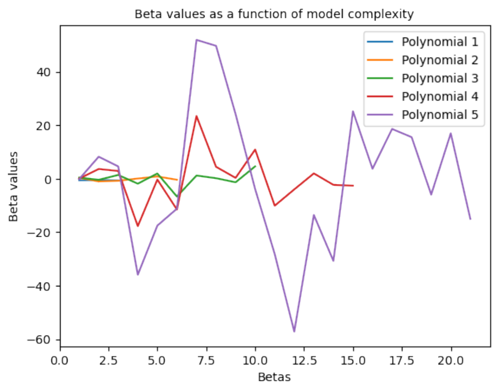


Figure 3: Beta values versus degree of polynomial

We can see from Figure 2 and 3 that mse decreases and R2 score increases with model complexity.

As we can see in Figure 4 to Figure 6 when the model complexity increases (higher degrees of polynomials are used to fit data), we get a bias-variance trade off. The mean square error which is largely dominated by the bias decreases as the complexity of the model increases. This comes at a cost of increased variance.

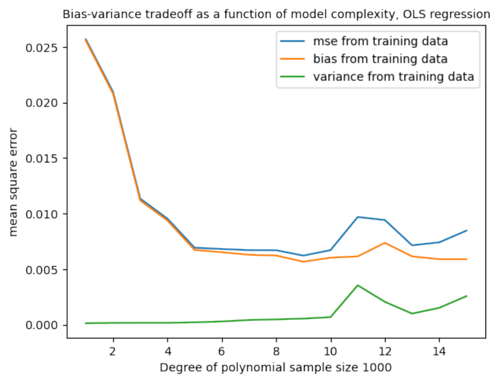
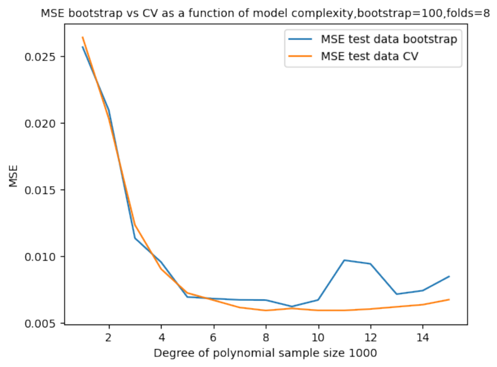


Figure 4: MSE,bias and variance vs degree of polynomial,number of bootstraps=100, noise=0.25,OLS regression



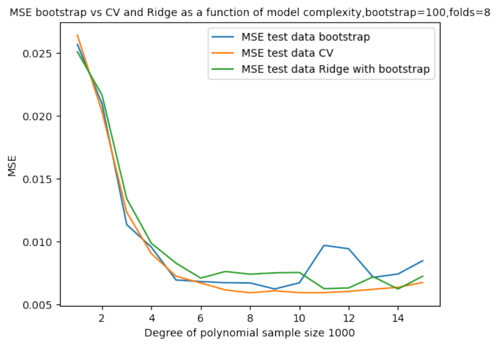


Figure 5: MSE for LASSO versus Ridge for Franke(left) and Terrain(right) versus lambda values

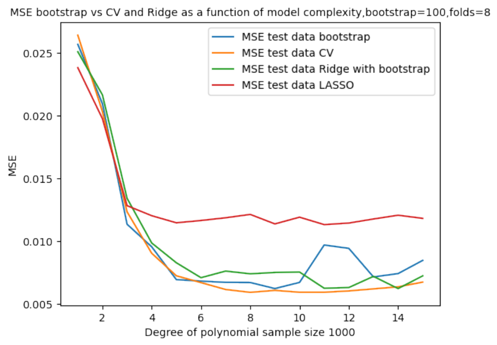


Figure 6: Bias and Variance as a function of model complexity for Franke function

Further results for The Franke Function and the terrain data are included in the github folder. As we can see in Table 1 the model that provides the best fit (lowest mse) for the Franke function is the Ordinary Least Square method with cross validation, with the Ridge method second and OLS, third LASSO in fourth place.

Table 1: Minimum MSE for Polynomials Franke Function

Regression method	**MSE Values**	**polynomial**
RIDGE	0.006215	14
LASSO	0.011314	11
OLS	0.006216	9
OLS with CV	0.005914	8

In table two we can see the result for the terrain data. The OLS method has the lowest mse and provides the best fit, then Ridge second and lasso third.

Table 2: Minimum MSE for Polynomial 5 Terrain Data

Regression method	**MSE Values**	**lambda**
LASSO	0.00092364	0.001
Ridge	0.00055127	0.001
OLS	0.00050764	Na

5 Conclusion

We have have performed linear regression fits for both the Franke function as well as real world terrain data. These fits indicate that ordinary least squares regression with CV are better suited (lower mse) at fitting data than Ridge- and Lasso regression for the Franke function. For the terrain data, the OLS regression method had the lowest mse score and provided the best fit. The reason for this seems to be that for the polynomial fit chosen the MSE are in a region where bias dominates the results and the effect of variance is very small.

We would need to use more complex models than the ones currently being investigated to better fit the data sets to the models. Then variance would matter more for the total error, and methods that aim at reducing this error such as Ridge- and LASSO regression would perform better.

This project looked specifically at polynomials of degree 5. As we can see from these results higher degrees of polynomials might provide better fits for these functions. Especially if the lambda values are adjusted properly.

6 Bibliography

[1] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning: data mining, inference, and prediction. 2009.

[2] Lecture Notes for FYS-STK4155.
https://compphysics.github.io/MachineLearning/doc/web/course.html
Accessed: 11.10.2022