Regression Analysis and Resampling Methods

Stian Aannerud, Marius Børvind, Dennis Fremstad and Andreas Wetzel

October 11, 2022

In this project we study the Ordinary Least Squares, Ridge and Lasso regression model to make models of some fiducial data generated by the Franke function, before we move on to real data, which consists of altitudes. We also explore re-sampling techniques such as Bootstrap and cross-validation in order to improve our model. We find that, given the size of the data, OLS provides the best sufficient fit to describe the model with a mean square error of $MSE \approx 9$. LASSO and RIDGE also performed similarly where LASSO reached an MSE of $10^{1.72}$ at polynomial degree 16 and $\lambda = 6.16 \cdot 10^{-4}$, while RIDGE got an MSE of $10^{1.2}$ at polynomial order 17 and $\lambda = 4.83 \cdot 10^{-12}$.

1 Introduction

The aim of this project is to study various regression models and methods of resampling. We will consider Ordinary Least Squares (OLS), Ridge and LASSO regression models, the Bootstrapping resampling technique, and k-fold cross validation. This will lead us to discuss how model setups lead to under-/overfitting, displayed by the Bias-variance trade-off which we discuss. We will test our method on data we generate using the Franke function with added stochastic noise, before applying our findings to model real cartographic data.

We start off by describing background information in the Theory section regarding how the data is approached and how this leads to different regression models and resampling methods. Then in our Method section we cover our numerical implementation of these while also introducing the data which we test and apply our methods to.

All code used in this project can be found at our GitHub page `https://github.com/frdennis/FYS-STK4155`.

2 Theory

Before performing the linear regression methods, we start by assuming that our data $\mathbf{y}$ can be described by a continuous function $f(\mathbf{x})$ and a normal distribution error $\boldsymbol{\epsilon}$ with mean 0 and variance σ^2, $\epsilon \sim N(0, \sigma^2)$,

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

We then approximate the function $f(\mathbf{x})$ with a model,

$$\tilde{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta}, \tag{2}$$

where $\mathbf{X}$ is called the design matrix. It has dimension $n \times p$ where n is the number of samples and p is the number of features. The vector $\boldsymbol{\beta}$ contains the unknown parameters. By defining a so called *Cost function* $C(\mathbf{X}, \boldsymbol{\beta})$ to determine how well our model fits the data, we can gauge the effectiveness of our parameters $\boldsymbol{\beta}$. Attempting to minimize this function will also give us an idea of how to improve our parameters, or in the case of the simple models used in this project we can outright solve for the optimal parameters $\hat{\boldsymbol{\beta}}$. The simplest model we will use comes from defining the Cost function to be the mean squared error (MSE), leading to the Ordinary Least Squares (OLS) method. The mean squared error is defined as

$$MSE(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \tilde{y}_i)^2, \tag{3}$$

where $\tilde{y}_i$ is the predicted value of element i of the sample and y_i is the true value of that element. The cost function can therefore be written

$$\begin{aligned} C(\mathbf{X}, \boldsymbol{\beta}) &= \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \tilde{y}_i) = \mathbb{E}[(\mathbf{y} - \tilde{\mathbf{y}})^2] \\ &= \frac{1}{n} \left\{ (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \right\}, \end{aligned} \tag{4}$$

where we have inserted for $\tilde{\mathbf{y}}$. The optimal parameters $\hat{\boldsymbol{\beta}}$, which give the best model, can then be determined by minimizing this expression. That is done by differentiating with respect to $\boldsymbol{\beta}$, keeping $\mathbf{X}$ and $\mathbf{y}$ constant, and setting the resulting expression to zero. Solving for $\boldsymbol{\beta}$ then gives the following solution,

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{5}$$

Properties which will become important in this project are the expectation value, $\mathbb{E}$, and variance of the data $\mathbf{y}$ and the optimal parameters $\hat{\boldsymbol{\beta}}$. These can be expressed as,

$$\mathbb{E}[y_i] = \sum_j x_{ij}\beta_j = \mathbf{X}_{i,*}\boldsymbol{\beta}, \tag{6}$$

$$\mathrm{Var}(y_i) = \sigma^2, \tag{7}$$

$$\mathbb{E}[\hat{\boldsymbol{\beta}}] = \boldsymbol{\beta}, \tag{8}$$

$$\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}. \tag{9}$$

Here, Equation 6 and 7 show the expectation value and variance of an element i of the model $\tilde{\mathbf{y}}$. The notation $\mathbf{X}_{i,*}$ means the sum over all elements in row i of $\mathbf{X}$. The derivation of these expressions can be found in Appendix A.

One with a keen eye might have spotted that each element of $\mathbf{y}$ is normally distributed with mean $\mathbf{X}_{i,*}\boldsymbol{\beta}$ and variance σ^2, $y_i \sim N(\mathbf{X}_{i,*}\boldsymbol{\beta}, \sigma^2)$. We see that the expectation value of the optimal parameters $\hat{\boldsymbol{\beta}}$ are simply $\boldsymbol{\beta}$. That means $\hat{\boldsymbol{\beta}}$ are unbiased.

From Equation 5 we see that the inverse of the matrix $\mathbf{X}^T\mathbf{X}$ is needed to find $\hat{\boldsymbol{\beta}}$. However, this matrix might not be invertible and Equation 5 can therefore not be used directly.

One way to avoid this problem is to use what is called the singular value decomposition (SVD). The matrix $\mathbf{X}^T\mathbf{X}$ is then decomposed into two orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$, and a diagonal matrix Σ as

$$\mathbf{X}^T\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T. \tag{10}$$

The matrices $\mathbf{U}$ and Σ contains the eigenvectors and and eigenvalues of $\mathbf{X}^T\mathbf{X}$ respectively. With the SVD method, we can compute the pseudoinverse of $\mathbf{X}^T\mathbf{X}$, giving a result even where the matrix is normally not invertible.

Another way of dealing with $\mathbf{X}^T\mathbf{X}$ not being invertible, is to add a small parameter λ to the diagonal of the matrix to ensure that the determinant is not zero. The expression for the optimal parameters $\hat{\boldsymbol{\beta}}$ then becomes,

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}. \tag{11}$$

This is also the expression for the optimal parameters $\hat{\boldsymbol{\beta}}$ in the method called Ridge regression. In this method we define a new cost function,

$$C(\mathbf{X}, \boldsymbol{\beta}) = \frac{1}{n}||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2, \tag{12}$$

where the norm-2 is defined as

$$||\boldsymbol{x}||_2 = \sqrt{\sum_i x_i^2}. \tag{13}$$

Minimizing Equation 12 with respect to $\boldsymbol{\beta}$ gives the solution shown in Equation 11.

Here λ is a regularization parameter. Its value depends on the problem and can be found in different ways. An example is to consider several values and choose the one that gives the smallest error. The advantage with the Ridge regression method, is that it decreases the less important parameters of $\boldsymbol{\beta}$.

The last linear regression method we will consider is called Lasso regression. In this method we define the following cost function,

$$C(\mathbf{X}, \boldsymbol{\beta}) = \frac{1}{n}||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1, \tag{14}$$

with the norm-1 defined as

$$||\boldsymbol{x}||_1 = \sum_i |x_i|. \tag{15}$$

For the Lasso regression method, there is no analytical expression for the parameters $\boldsymbol{\beta}$, so we cannot find $\boldsymbol{\beta}$ in the same way as for OLS and Ridge regression. We notice that this method also consist of the parameter λ which means the less important parameters of $\boldsymbol{\beta}$ is again suppressed.

After having fitted a model to the data, the aim is to analyse the accuracy of the model. Two quantities that are used extensively for this purpose are the MSE, shown in Equation 3, and the R^2 score function which is calculated using,

$$R^2(\mathbf{y}, \tilde{\mathbf{y}}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2}. \tag{16}$$

Here, $\bar{y}$ is the mean value of $\mathbf{y}$,

$$\bar{y} = \mathbb{E}[\mathbf{y}] = \frac{1}{n}\sum_{i=0}^{n-1} y_i. \tag{17}$$

We see that the R^2 score function is equal to MSE, only weighted by the difference between the data points and the mean of the data. That means points that are far away from the average value of the data will have less effect on the error analysis.

In appendix B we show that the MSE in equation 4 can be expressed as a sum of three quantities,

$$\mathbb{E}[(\mathbf{y}-\tilde{\mathbf{y}})^2] = (\text{Bias}[\tilde{y}])^2 + \text{var}[\tilde{f}] + \sigma^2, \tag{18}$$

where

$$(\text{Bias}[\tilde{y}])^2 = \frac{1}{n}\sum_{i=0}^{n-1}(f_i - \mathbb{E}[\tilde{\mathbf{y}}])^2 \tag{19}$$

$$\text{var}[\tilde{f}] = \frac{1}{n}\sum_{i}(\tilde{y}_i - \mathbb{E}[\tilde{\mathbf{y}}])^2. \tag{20}$$

Equation B makes it easier to see where the error of the model comes from. The error due to the model being to simple is represented by the bias term, which is the difference between mean value of the data and the mean value of the model. This term therefore describes how much the model misses the expected value of the data. For a very simplified model, the bias will be large and we get what is called underfitting of the data.

On the other hand, we do not want the model to be too complicated. Although this will reduce the error from the bias, it will lead to overfitting, and thus to an increase in the error due to the variance. This is represented by the second term in Equation 18 which describes how each model value deviates from the mean value of the model. In other words, this term represents the error caused by the spread of the model values compared to the expected value of the model.

The last term in Equation 18, σ^2, represents the error due to the variance of the data. Since we do not know this value, there is no way to avoid this error. However, the bias and variance, mentioned above, depends on the model complexity. The combination of these quantities is therefore important when assessing the model. Since the values of both quantities determine whether we get overfitting or underfitting, we are interested in the model where the quantities are equal. This is called the bias-variance trade-off. Later in this report we will discuss how the bias-variance trade-off varies with respect to the model complexity.

Next, we will also consider two resampling methods used to assess the models. The first one is called bootstrap. In this method, a regression method is first applied to find the optimal parameters $\hat{\boldsymbol{\beta}}$ for a given data set. It should be mentioned that in regression methods, the data is divided into training and test data. The training data is used when performing the regression, and the test data is used to evaluate the model.

In the bootstrap method, random values are drawn from the original training data and replaced in a new data set. New parameters $\hat{\boldsymbol{\beta}}$ can then be found. This process is repeated B times. In each bootstrap operation, quantities like the MSE or the variance can be computed. In the end the average of these quantities can be used in the model assessment. Since one generates new data sets, and thus increases the amount of data, the bootstrap method is useful for small data sets.

The next resampling method is called k-fold cross-validation. The aim of this method is to avoid that only one part of the data is used for training and another for testing. The way this is done, is to first randomly reshuffle the data before dividing it into k parts. k is usually between 5-10. Then, each of the parts are in turn used as test data while the remaining parts are used for training. For each configuration, the MSE or variance can be found. In the end, as for the bootstrap method, these quantities are used in the evaluation of the model.

We have now presented a lot of theoretical aspects of linear regression. In the next section we describe how they were implemented numerically.

3 Method

In order to explore the various regression methods, we use the Franke function to make our own data set. This function takes the following form,

$$\begin{aligned} f(x,y) = &\frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) \\ &+ \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) \\ &+ \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) \\ &- \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right) \end{aligned} \tag{21}$$

In this project, we use $x, y \in [0,1]$. We choose to subtract the mean value to center the data around 0. The

reason for this is that we do not need to account for the intercept, meaning we will have one parameter less to fit.

We also explore the addition of some added stochastic noise, ϵ, that follows a normal distribution with mean 0 and variance 0.1 times the mean of the data. Alternatively we could have chosen to have a noise with variance 1 and scale the Franke function up instead, in which case we must make sure not to scale it too far where the noise becomes insignificant. Since the Franke function was already at a neat order, there was no real need to scale it. Therefore we choose to scale down the added noise to make sure that the Franke function is not completely "washed out" by the noise. Our own data set can then be written as

$$\mathbf{z} = f(x, y) + \boldsymbol{\epsilon}, \tag{22}$$

where $\mathbf{z}$ is a two dimensional matrix.

We split the data into a training set and a testing set, where 25% of the data goes to the testing set and the rest goes to the training set. One way to split the data is to take the first fourth of the data as the testing set and the last three fourths as the training set. However, in this case some features might only be present in one set which decreases the accuracy of the model. To make sure that all features in the data are present in both sets, we split the data such that for every four points, one point goes to the testing set and three points goes to the training set.

We perform the regression methods by creating a design matrix $\mathbf{X}$. In this project we will fit polynomials of different orders to data. Hence, the design matrix will consist of the different polynomial terms. Since we are working with two dimensional data, we also need to remember the cross terms. As an example, for a second order polynomial our design matrix will take the form

$$\mathbf{X} = \{1, x, y, xy, x^2, y^2\}, \tag{23}$$

where each element in X is a vector with length n. This means that, for example, x is a vector with n points from 0 to 1 (and the 1 in X is a vector of length n with only ones). Having made the design matrix, we can calculate the optimal values that fit our model to our data. In the case of OLS we find the optimal parameters using

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{z}, \tag{24}$$

where $\mathbf{z}$ is our $2D$ data at points x, y. In order to get the right dimensionality, this matrix is flattened. Since the matrix $\mathbf{X}^T\mathbf{X}$ might not be invertible, we chose to use the SVD to compute the inverse of this matrix.

When performing Ridge regression, the optimal parameters were found by

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{z}. \tag{25}$$

We tried different values of λ to find the one that gave the best fit to the data.

Since there are no simple analytical solutions for the optimal parameter $\hat{\boldsymbol{\beta}}$ in the Lasso regression method, we used the Lasso method from Scikit-Learn.

We are given data with 3601×1801 data points, which consists of the altitudes of southern Norway to perform our analysis on. The data is originally sourced from the United States Geological Survey agency, of type SRTM Arc-Second Global and Entity ID SRTM1N59E008V3, and provided through FYS-STK4155 course instructors. Due to the size of the data we select a small portion of the full data to perform our analysis on. The more features we include the higher the polynomial degree we require in order to describe all these features with our model. This is why, in order to keep the code from taking too long, we only pick a 30×30 section of the data, which only contains some of the features. The data has a resolution of 1 arc-second per pixel, which is about 30 meters.[1], the total area of our piece of the data is about 0.81 km^2.

[1] See https://www.usgs.gov/centers/eros/science/shuttle-radar-topography-mission-data-dictionaryresolution

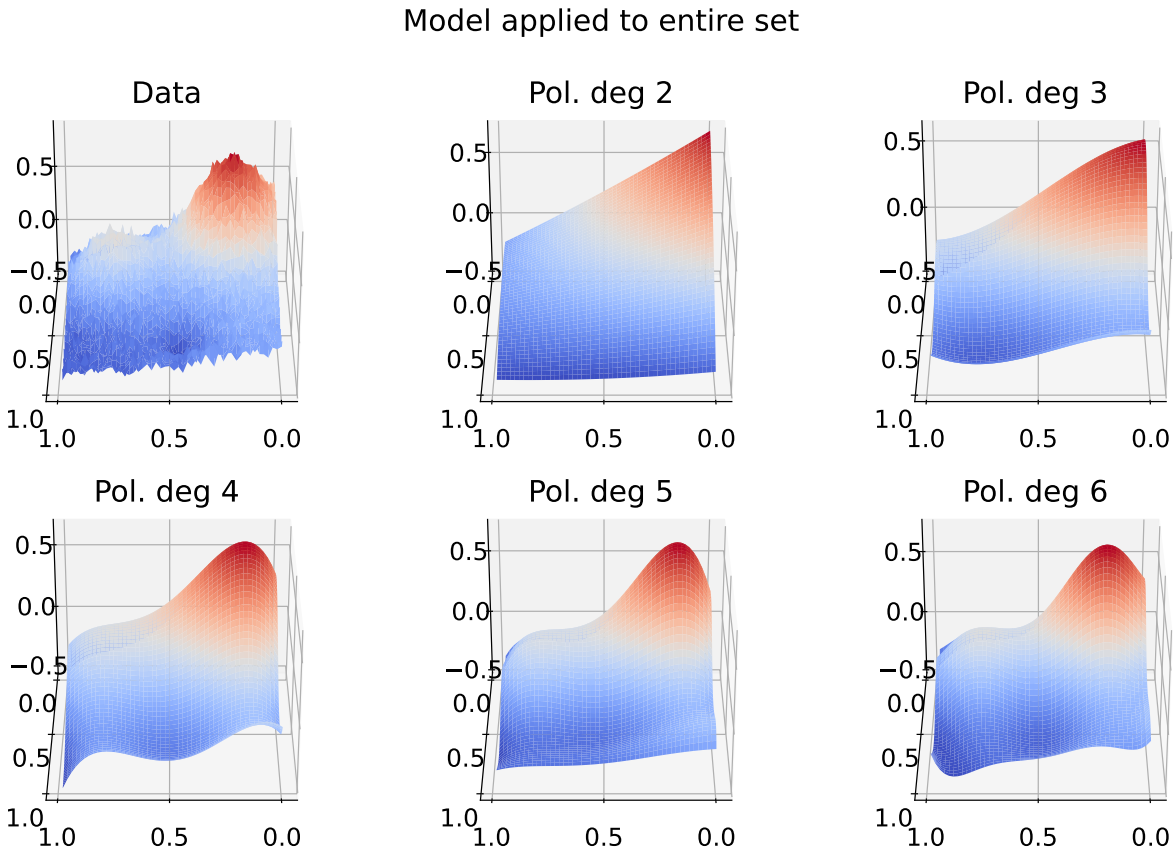

Figure 1: Surface plots of the Franke function and OLS regression fits in terms of polynomial degree, p.

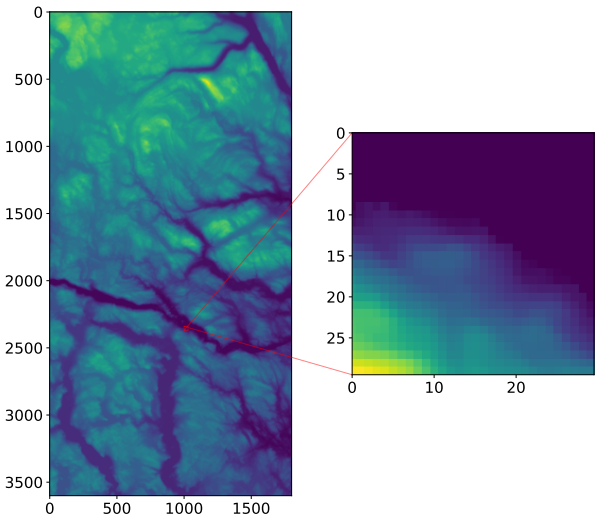

Figure 2: Image view of the cartographic data. We use only a small 30×30 box (right) from the full data (left).

4 Results & Discussion

We plot the $R2$ score and MSE as a function of polynomial degree using 21 as our data to test our methods. We plot the MSE in figure 3 and the $R2$ score in figure 4.

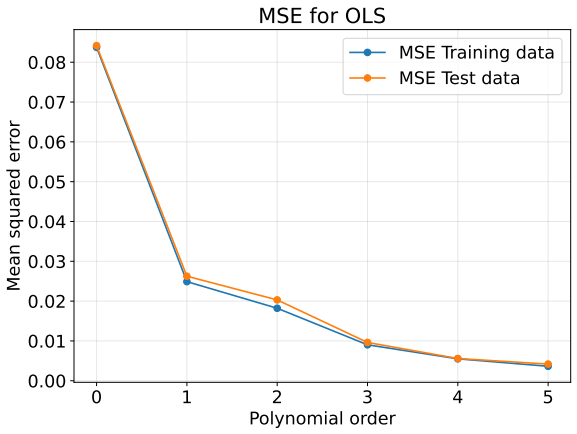

Figure 3: From OLS we plot MSE as a a function of polynomial degree. Where the orange line represent the test data and the blue line represent the training data.

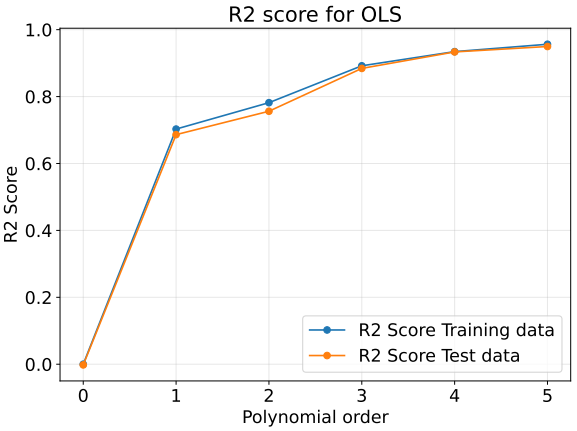

Figure 4: From OLS we plot R^2 as a a function of polynomial degree. The orange line represent the test data and the blue line represent the training data.

We can see from figure 3 that as we increase the polynomial degree the MSE decreases, meaning we are getting a better fit for the model for every increase in polynomial degree. We are here plotting both the training data and the test data. The training data is the data we use to make a fit, while the test data is what we use to compare our model to. Since these lines are very close to overlapping, it means that the model we have fitted is sufficient in describing our data. If these lines would start to grow apart from each other then this would be a sign that we are over-fitting our model to the training data. This means that our data is so well fitted to the training data, taking every piece of noise into account to minimize the variance, that when we compare it to the test data we see a high variance as the model over compensates for noise and thus missed the actual "signal" (remember that with our model we are trying to find f in equation 1).

In figure 4 we plot the $R2$-score for our OLS fit to the Franke function. We get a similar conclusion to the discussion above by looking at this figure; we see that as the polynomial degree increases, the accuracy of our model increases. This is apparent since the $R2$-Score approaches 1 as we increase the polynomial order.

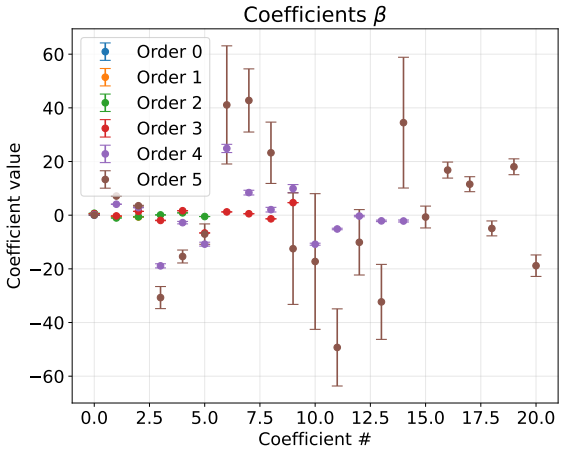

Figure 5: The coefficients β with variance for different polynomial orders.

In figure 5 we plot the coefficients used to construct our model for a given polynomial order (recall equation 2 to see exactly how these coefficients β makes up our model). We can see that for all orders except for with polynomial order 5 the variance of the coefficients are relatively low. For the fit with polynomial order 5 we get low variance for the first and last set of coefficients, while the ones in the middle seem to yield a higher variance. We interpret this as the model already being well fitted with the first and

last set of parameters, while the ones in the middle add up to a small or no contribution to the data. Since they add up to a small or no contribution there is a range of ways they can be combined to do so, which results in a higher variance.

We divide the MSE of our model using the Bootstrap re-sampling method into three parts; the variance, the bias and the error. This is done in figure 6. We can see that at around polynomial order 5 we get a bias-variance trade-off where the variance crosses over the bias. At this stage the error goes over from being dominated by the bias to being dominated by the variance. In other words, we go from under-fitting to over-fitting our data. We can see that the best fit (least MSE) is found at polynomial order 5 right before the trade off happens. In figure 7 we again plot the variance, bias and error but we increase the size of our data set from 10×10 points to 20×20 points, in addition to increasing the max polynomial order present in the calculations from 9 to 19. Also in this figure we find that the best fit is found just before the bias-variance trade-off, at polynomial degree 6 while the trade-off happens at degree 10. We can see that increasing the size of our data moves the point where the trade-off happens. After the trade-off, the error, bias and variance increases further with increasing complexity. This is a sign of over-fitting.

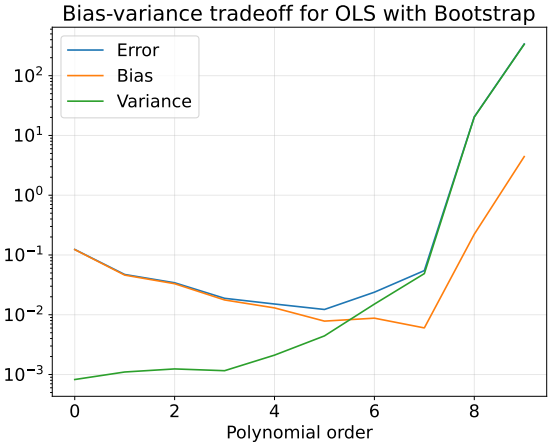

Figure 6: Plot of the bias, variance and MSE for test data as a function of polynomial degree between 0 and 9, using data with 10×10 points. We use $B = 100$ iterations.

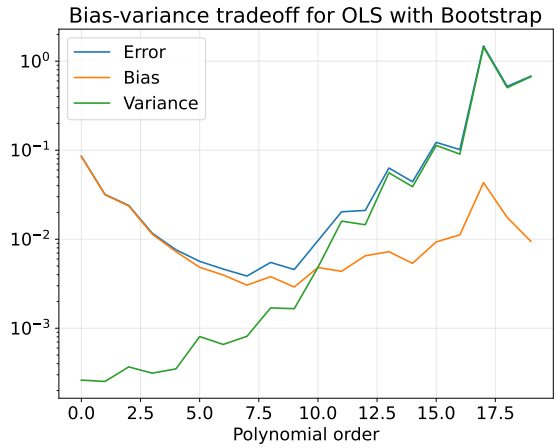

Figure 7: Plot of the bias, variance and MSE for test data as a function of polynomial degree between 0 and 19 using data with 20×20 points. We use $B = 100$ iterations.

The result of performing the cross-validation re-sampling technique on our data is plotted in figure 8. In this case we use $k = 10$ k-folds. We can see that the test data starts to stray away from the training data after polynomial order around 4. This is a sign that we are over-fitting the data.

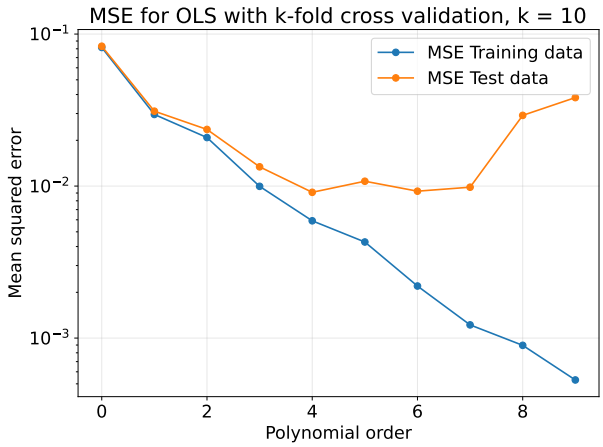

Figure 8: A plot of for MSE as a function of polynomial degree for OLS by using cross-validation resampling method.

We perform a Ridge regression on our model with λ ranging from 0.1 to 10^{-12} and between polynomial orders 0 and 19 alongside a Bootstrap re-sampling. The result is shown in figure 9.

The parameter λ affects our model by restricting which polynomial orders dominate our model. Since the model can be better described by polynomials of orders around 8 than those close to 0 it makes sense to let the model be more dependent on the coefficients of the polynomials of orders close to 8. This is reflected in the figure where the MSE for polynomial order 0 to about 4 is higher than for the rest of the polynomial orders. We find that the best fit is found at $p = 14$ and $\lambda \approx 1.6 \times 10^{-7}$, where the mean squared error is of the order $MSE = 10^{-3.12}$.

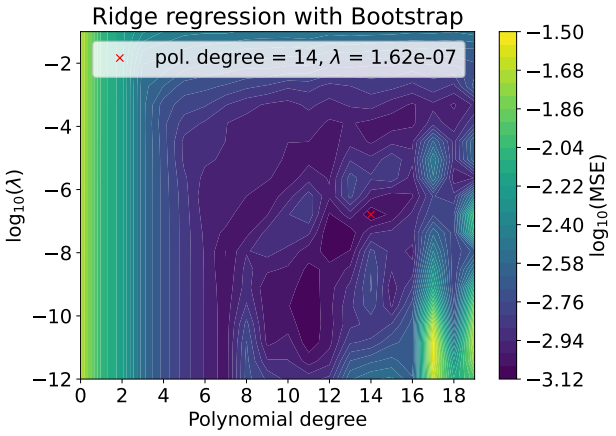

Figure 9: Performing a Ridge regression using bootstrap method to plot the MSE as a function of polynomial degree.

We perform the same analysis as above, but now take use of cross-validation to re-sample our data. In figure 10, we can see that we get similar results to what we found previously. Compared to the Bootstrap case, there is a much larger region where the MSE reaches below 10^{-2}. We also see that for the smallest λ there is an increase in the MSE for higher polynomial degree. This is an indication that we only get bias-variance trade-off for very small λ. Since bias-variance trade-off can only happen for higher polynomial orders (remember that bias-variance trade-off happens when we are beginning to over-fit and include the noise in our model, which can only be done with a high polynomial order), it is also an indication that larger λ suppresses higher polynomial orders, while smaller λ suppresses lower polynomial orders. The best fit is found at $\lambda \approx 3 \times 10^{-9}$ and $p = 11$.

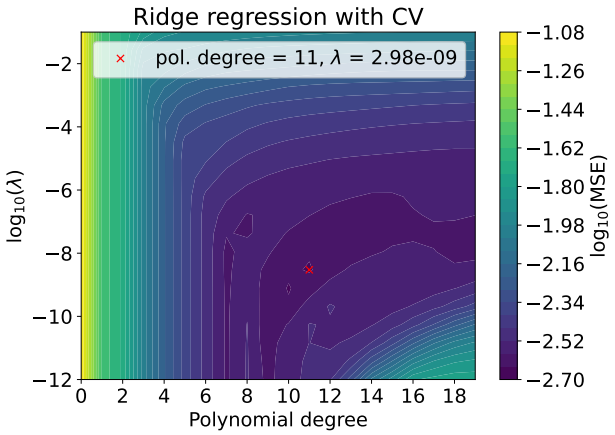

Figure 10: Performing a Ridge regression using cross validation to plot the MSE as a function of polynomial degree.

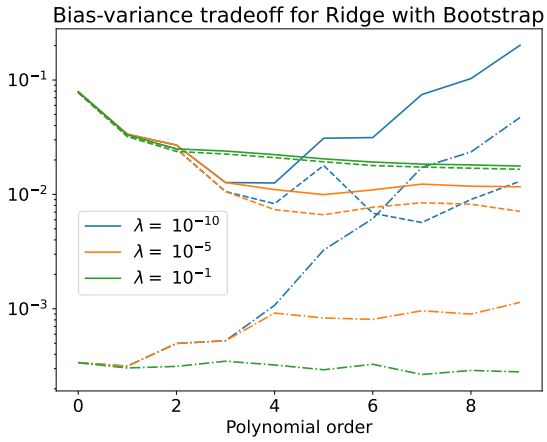

Figure 11: MSE (solid), bias (dashed) and variance (dash dot) using different λ as a function of polynomial degree.

We plot the MSE, bias and variance for three values of λ using Ridge regression with Bootstrap. We can see in figure 11 that we get bias-variance trade-off

for the case with $\lambda = 10^{-10}$, and not for the other two cases. Again this is because λ is too large to allow this trade-off to occur.

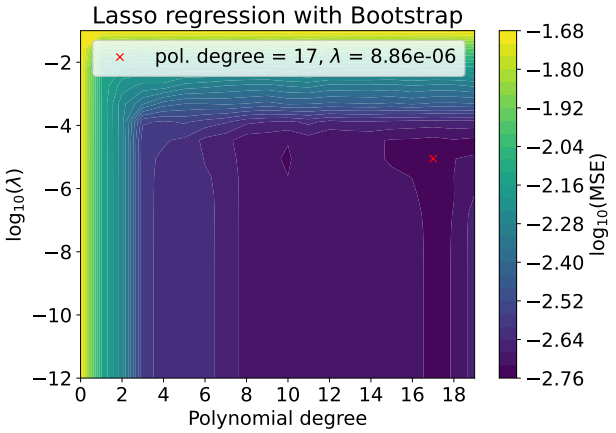

Figure 12: Performing a Lasso regression using bootstrap method to plot the MSE as a function of polynomial degree.

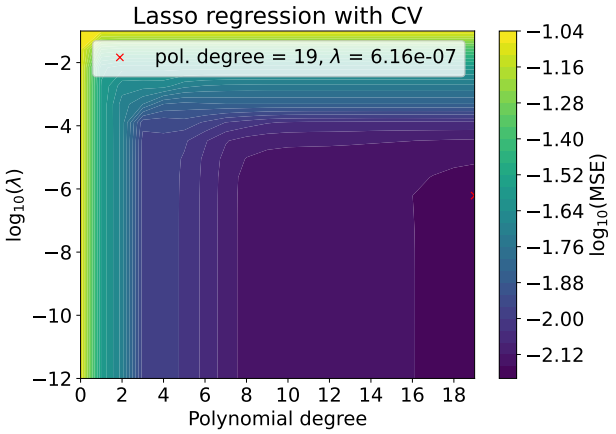

Figure 13: Performing a Lasso regression using cross validation to plot the MSE as a function of polynomial degree.

We use the Lasso regression method alongside Bootstrap (figure 12) and cross-validation (figure 13). Again we see that there is much more variation in the Bootstrap case, compared to when we use cross-validation. Since the cross-validation method resamples the training data itself and not just the points within it, it makes sense that the cross-validation resampling plot is much smoother than for Bootstrap. We are essentially sampling more of the data, which naturally gives rise to a smoother result. The result of using LASSO is that the plot varies much less. Since LASSO is much more restrictive when it comes to what polynomial degrees dominates for a given λ, there is a large difference in MSE depending on what value of λ we use. We can see that for smaller λ, the MSE is much higher than it is for larger λ where it is generally much smaller. We find that the best fit using Bootstrap has $MSE \approx 10^{-2.76}$, $p = 17$ and $\lambda \approx 8.86 \cdot 10^{-6}$

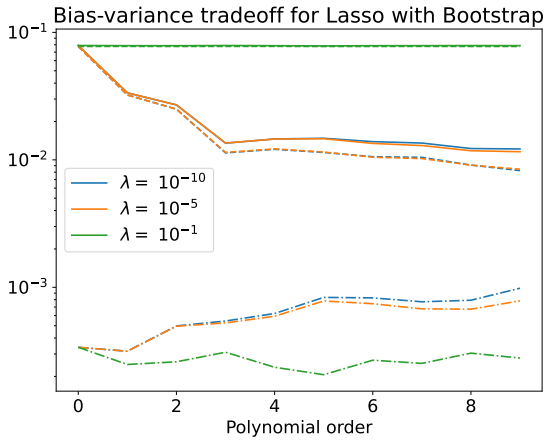

Figure 14: MSE (solid), bias (dashed) and variance (dash dot) using different λ as a function of polynomial degree.

We plot the MSE, bias and variance for three different values of λ as a function of polynomial order in figure 14. We can see that there is no trade-off for any of the cases. This could be because LASSO requires even higher polynomial orders in order to achieve trade-off, or it could be because λ is too large.

4.1 Analysis of Real Data

We test our regression models on data of the altitudes of southern Norway to see how these models fares against real data. In figure15 we have plotted the result of using the OLS regression model on the data. We can see that once we hit polynomial order 2 there is very little change in the MSE. We see similar results

in the R^2-score for the same regression model in figure 16, where the R^2-score is very close to 1 at polynomial degree 2 as well as starting to approach 1 much slower at this degree.

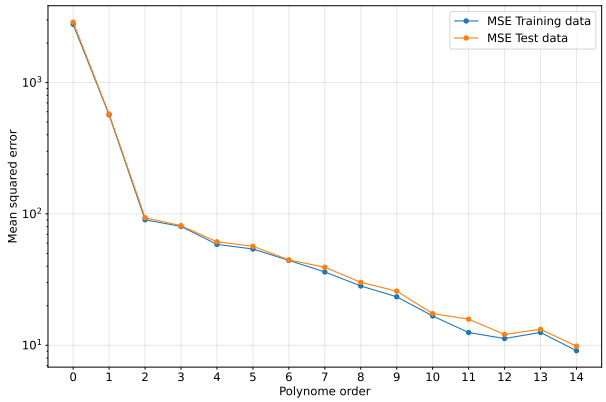

Figure 15: MSE of the data with 25% of the data assigned as test data. We calculate between orders 0 and 14.

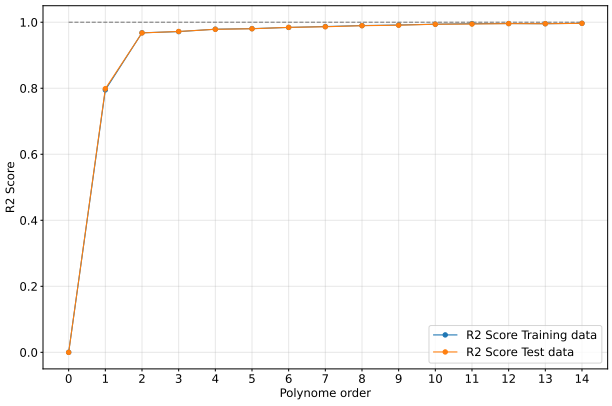

Figure 16: R^2 Score for our data.

We perform the Bootstrap re-sampling technique on the data, and the result is plotted in figure 17. We can see that we get bias-variance trade off around polynomial degree 11, while the point with least error is found a little before this at polynomial degree 10. This is similar to the result found when applying this method to the Franke function. In this case we find that OLS performs better than Bootstrap in finding a model that fits the data as OLS gives smaller MSE.

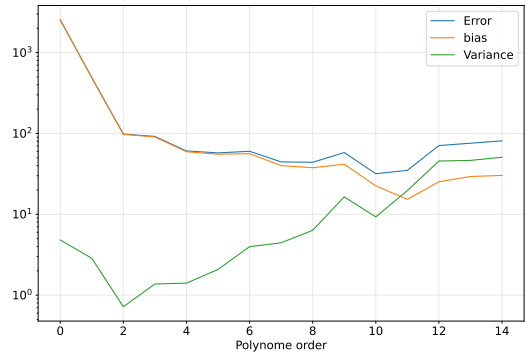

Figure 17: Bootstrap applied to the data with $n = 90$ iterations, up to polynomial degree 14.

In figure 18 we show the result of calculating the MSE using cross validation with $k = 50$ k-folds. Again we find that OLS finds a better fit to the data than cross validation.

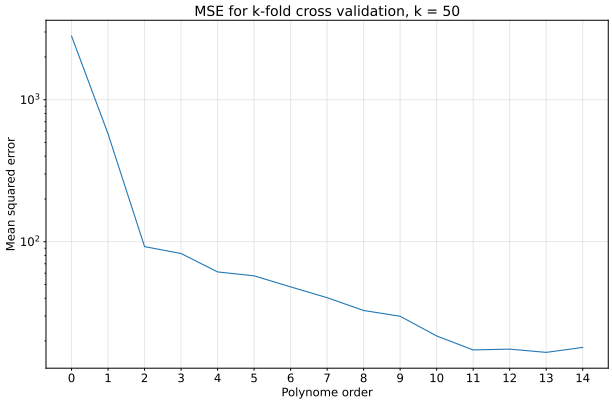

Figure 18: MSE for cross validation with $k = 50$ k-folds.

The result of performing a Ridge regression on the data is plotted in figure 19. We can see that the MSE varies a lot depending on which λ we use. The best fit is found at polynomial degree 17 and $\lambda = 4.83 \cdot 10^{-12}$.

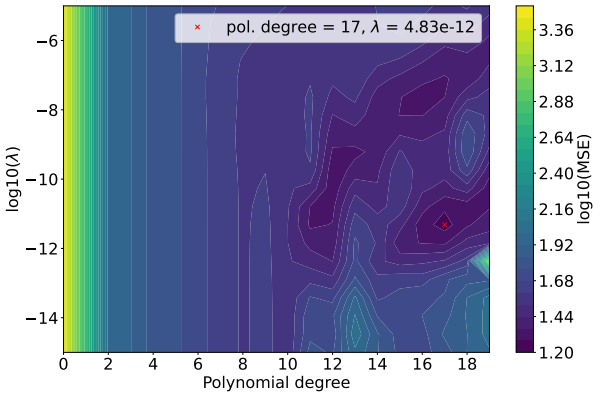

Figure 19: MSE for Ridge regression with varying λ and polynomial order. We can see that the best fit (smallest MSE) is found with polynomial degree 17 and $\lambda \approx 4.83 \cdot 10^{-12}$.

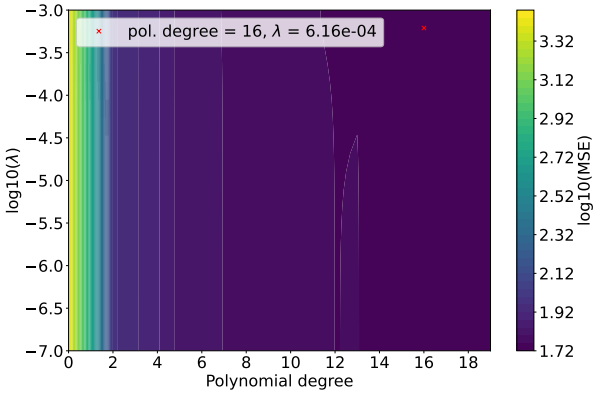

Figure 20: MSE with Lasso regression for varying polynomial degree and λ. The best fit in this case if found at polynomial degree 16 and $\lambda \approx 6.16 \cdot 10^{-4}$

Lastly we plot the result of performing a Lasso regression on the data in figure 20. We find that the best fit to the data is found at polynomial degree 16 and $\lambda \approx 6.16 \cdot 10^{-4}$. Since LASSO is a more aggressive model in terms of varying λ we can see that there are not as many contours as in the case of Ridge. Notice how the MSE varies very little when compared to the Ridge case. In addition, we find that we require a much larger λ to get a sufficient fit to the data. This is also a result of the LASSO models dependence on λ. It should be mentioned that LASSO is an iterative method for calculating the MSE, which means it takes considerably longer to calculate.

5 Conclusion

In conclusion we found that while all our regression models give serviceable fits to our mock data, tweaking the parameter λ can give better results by suppressing less important factors. The mean squared error for all models reached scales of 10^{-2} and below, which for our means we fit the data well. Our best result for the Franke function came with Ridge regression with added Bootstrapping, where the MSE reached $10^{-3.12}$ for a polynomial degree of 14 and $\lambda = 1.62 \cdot 10^{-7}$. LASSO was close behind at $10^{-2.76}$, leaving both it and Ridge as great improvements over OLS where extra computational power is available, though as an iterative method LASSO also craves even more than Ridge.

The Bootstrap resampling method lead to our best results, and while the k-fold validation didn't have a large impact for us, it is still worth keeping to avoid potential train/test splitting pitfalls. Resampling lead to better fits on average for all our models. K-fold cross validation did not have much of an impact, but is important to keep in mind to avoid potential bias in selection of testing data.

Applying our findings to the real data set, we found the same results as for the generated data. All our models performed well, reaching mean squared errors of near 10. Our best results this time came from the OLS model, but they all within an order of magnitude as opposed to the initial Franke test, showing that it is important to try out different approaches to each data set.

References

[1] Hjorth-Jensen, M. (2022). Project1: *Machine Learning*`https://github.com/CompPhysics/MachineLearning/tree/master/doc/Projects/2022/Project1`

[2] Hjorth-Jensen, M. (2021). 5. Resampling Methods: *Applied Data Analysis and Machine Learning Resampling Methods*: `https://compphysics.github.io/MachineLearning/doc/LectureNotes/_build/html/chapter3.html`

[3] Hastie, T., Tibshirani, R. , & Friedman, J. (2017). *The Elements of Statistical Learning*. (pp 33-47). Stanford CA, USA: Springer.

A Ordinary Least Squares model

In the ordinary least squares regression method we assume that our data $\boldsymbol{y}$ can be described by a continuous function $\boldsymbol{f}$ and some normally distributed noise $\boldsymbol{\epsilon}$ with mean 0 and variance σ^2,

$$\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\epsilon}. \tag{26}$$

We then approximate $f(\boldsymbol{x})$ by a model

$$\tilde{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta}, \tag{27}$$

where $\mathbf{X}$ is the design matrix and $\boldsymbol{\beta}$ are the unknown parameters. The expectation value of an element i in the data can then be calculated using

$$\mathbb{E}[y_i] = \mathbb{E}[\tilde{y}_i] + \mathbb{E}[\epsilon_i] = \mathbb{E}\left[\sum_j x_{ij}\beta_j\right] = \sum_j x_{ij}\beta_j = \mathbf{X}_{i,*}\boldsymbol{\beta}, \tag{28}$$

since the expectation value of ϵ is zero. $\mathbf{X}_{i,*}\boldsymbol{\beta}$ means summing over all elements in row i. The reason $\mathbb{E}[\sum_j x_{ij}\beta_j] = \sum_j x_{ij}\beta_j$ is that this is the model we provide. The elements of the model do not follow any distribution but are simply values from the function we choose. The expectation values are therefore just the values.

The variance of an element i of the data can then be found as follows,

$$\begin{aligned}
\mathrm{Var}[y_i] &= \mathbb{E}[y_i^2] - \mathbb{E}[y_i]^2 && (29)\\
&= \mathbb{E}[(\mathbf{X}_{i,*}\boldsymbol{\beta} + \epsilon_i)^2] - (\mathbb{E}[\mathbf{X}_{i,*}\boldsymbol{\beta} + \epsilon_i])^2 && (30)\\
&= \mathbb{E}[(\mathbf{X}_{i,*}\boldsymbol{\beta})^2 + 2(\mathbf{X}_{i,*}\boldsymbol{\beta})\epsilon_i + \epsilon_i^2] - (\mathbb{E}[(\mathbf{X}_{i,*}\boldsymbol{\beta})] - \mathbb{E}[\epsilon_i])^2 && (31)\\
&= \mathbb{E}[(\mathbf{X}_{i,*}\boldsymbol{\beta})^2] + \mathbb{E}[2(\mathbf{X}_{i,*}\boldsymbol{\beta})\epsilon_i] + \mathbb{E}[\epsilon_i^2] - \mathbb{E}[(\mathbf{X}_{i,*}\boldsymbol{\beta})]^2 && (32)\\
&= \mathbb{E}[2(\mathbf{X}_{i,*}\boldsymbol{\beta})\epsilon_i] + \mathbb{E}[\epsilon_i^2]. && (33)
\end{aligned}$$

y_i is independent of the noise ϵ_i, so we can take the expectation value of each component in the first term above. That gives zero since $\mathbb{E}[\epsilon_i] = 0$. Since the expectation value of ϵ is zero, we also have $\mathbb{E}[\epsilon_i^2] = \mathrm{Var}[\epsilon_i] = \sigma^2$. Thus, the variance of element i of the data is

$$\mathrm{Var}[y_i] = \sigma^2. \tag{34}$$

We have then found that the data $\boldsymbol{y}$ follows a normal distribution with mean $\boldsymbol{X}_{i,*}\beta$ and variance σ^2.

Next, we consider the expectation value and variance of the optimal parameter $\hat{\beta}$ which is given by

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\boldsymbol{y}. \tag{35}$$

The expectation value of $\hat{\boldsymbol{\beta}}$ is then

$$\mathbb{E}[\hat{\boldsymbol{\beta}}] = \mathbb{E}\left[\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\boldsymbol{y}\right] = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbb{E}[\boldsymbol{y}], \tag{36}$$

because the expectation value of $\mathbf{X}$ and its transpose are just $\mathbf{X}$ and $\mathbf{X}^T$. Inserting the expectation value of $\boldsymbol{y}$ found previously we get

$$\mathbb{E}[\hat{\boldsymbol{\beta}}] = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} = \boldsymbol{\beta}, \tag{37}$$

since $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{X} = \mathbf{I}$ where $\mathbf{I}$ is the identity matrix. The variance of $\hat{\boldsymbol{\beta}}$ is found from the following expression,

$$\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \mathbb{E}\left[(\hat{\boldsymbol{\beta}} - \mathbb{E}[\hat{\boldsymbol{\beta}}])(\hat{\boldsymbol{\beta}} - \mathbb{E}[\hat{\boldsymbol{\beta}}])^T\right]. \tag{38}$$

Inserting for $\hat{\boldsymbol{\beta}}$ and $\mathbb{E}[\hat{\boldsymbol{\beta}}]$ gives

$$\begin{aligned}
\mathrm{Var}(\hat{\boldsymbol{\beta}}) &= \mathbb{E}\left[\left((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} - \boldsymbol{\beta}\right)\left((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} - \boldsymbol{\beta}\right)^T\right] &(39)\\
&= \mathbb{E}\left[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}\mathbf{y}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} - (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}\boldsymbol{\beta}^T - \boldsymbol{\beta}\mathbf{y}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} + \boldsymbol{\beta}\boldsymbol{\beta}^T\right] &(40)\\
&= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbb{E}[\mathbf{y}\mathbf{y}^T]\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} - (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbb{E}[\mathbf{y}]\boldsymbol{\beta}^T - \boldsymbol{\beta}\mathbb{E}[\mathbf{y}^T]\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} + \boldsymbol{\beta}\boldsymbol{\beta}^T. &(41)
\end{aligned}$$

We can write $\mathbb{E}[\mathbf{y}\mathbf{y}^\mathbf{T}] = \mathbf{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\mathbf{X}^\mathbf{T} + \sigma^2\mathbf{I}$. Inserting this, and the expectation value for $\boldsymbol{y}$ found earlier, we get

$$\begin{aligned}
\mathrm{Var}(\hat{\boldsymbol{\beta}}) &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\left(\mathbf{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\mathbf{X}^\mathbf{T} + \sigma^2\mathbf{I}\right)\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} - (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}\boldsymbol{\beta}^T - \boldsymbol{\beta}\boldsymbol{\beta}^T\mathbf{X}^\mathbf{T}\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} + \boldsymbol{\beta}\boldsymbol{\beta}^T &(42)\\
&= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\mathbf{X}^\mathbf{T}\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} + \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T - \boldsymbol{\beta}\boldsymbol{\beta}^T + \boldsymbol{\beta}\boldsymbol{\beta}^T &(43)\\
&= \boldsymbol{\beta}\boldsymbol{\beta}^T + \sigma^2(\mathbf{X}^T\mathbf{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T &(44)\\
&= \sigma^2(\mathbf{X}^T\mathbf{X})^{-1} &(45)
\end{aligned}$$

This expression can be used to determine the standard deviation of the optimal parameters $\boldsymbol{\beta}$ that we find.

B Bias-Variance Trade-off

The cost function can be written in the following way as the expectation value of the difference between the data $\boldsymbol{y}$ and the model $\tilde{\boldsymbol{y}}$,

$$C(\boldsymbol{X}, \boldsymbol{\beta}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = \mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right]. \tag{46}$$

To investigate the relation between the bias and variance of the model we want to write the above expectation value in terms of these quantities. First we use the assumption that we can describe the data by a continuous function f and normally distributed noise ϵ with mean 0 and variance σ^2,

$$\boldsymbol{y} = \boldsymbol{f} + \boldsymbol{\epsilon}. \tag{47}$$

Inserting for $\boldsymbol{y}$ then gives,

$$\mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right] = \mathbb{E}\left[(\boldsymbol{f} + \boldsymbol{\epsilon} - \tilde{\boldsymbol{y}})^2\right]. \tag{48}$$

Then, to get the expression on the desired form we add and subtract the expectation value of the model $\mathbb{E}\left[\tilde{\boldsymbol{y}}\right]$,

$$\begin{aligned}
\mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right] &= \mathbb{E}\left[(\boldsymbol{f} + \boldsymbol{\epsilon} - \tilde{\boldsymbol{y}} + \mathbb{E}\left[\tilde{\boldsymbol{y}}\right] - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2\right] &(49)\\
&= \mathbb{E}\left[\left((\boldsymbol{f} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right]) - (\tilde{\boldsymbol{y}} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right]) + \boldsymbol{\epsilon}\right)^2\right] &(50)\\
&= \mathbb{E}\left[(\boldsymbol{f} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2\right] + \mathbb{E}\left[(\tilde{\boldsymbol{y}} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2\right] + \mathbb{E}\left[\boldsymbol{\epsilon}^2\right]\\
&\quad + \mathbb{E}\left[2(\boldsymbol{f} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])\boldsymbol{\epsilon} - 2(\tilde{\boldsymbol{y}} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])\boldsymbol{\epsilon} - 2(\boldsymbol{f} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])(\tilde{\boldsymbol{y}} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])\right]. &(51)
\end{aligned}$$

We now introduce the bias and variance of the model as

$$(\text{Bias}[\tilde{\boldsymbol{y}}])^2 = \mathbb{E}\left[(\boldsymbol{f} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2\right] = \frac{1}{n}\sum_i (f_i - \mathbb{E}[\tilde{\boldsymbol{y}}])^2 \tag{52}$$

$$\text{Var}[\tilde{\boldsymbol{y}}] = \mathbb{E}\left[(\tilde{\boldsymbol{y}} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2\right] = \frac{1}{n}\sum_i (\tilde{y}_i - \mathbb{E}[\tilde{\boldsymbol{y}}])^2. \tag{53}$$

The expectation value of the squared of the noise can be found by considering the expression for the variance of the noise which is σ^2,

$$\text{Var}[\boldsymbol{\epsilon}] = \mathbb{E}[\boldsymbol{\epsilon}^2] - \mathbb{E}[\boldsymbol{\epsilon}]^2. \tag{54}$$

The expectation value of the noise is zero, which gives

$$\mathbb{E}[\boldsymbol{\epsilon}^2] = \text{Var}[\boldsymbol{\epsilon}] = \sigma^2. \tag{55}$$

Since the expectation value of ϵ is zero, and ϵ is independent of the other quantities in Equation 51, the terms in Equation 51 containing only ϵ becomes zero. We are then left with

$$\mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right] = (\text{Bias}[\tilde{\boldsymbol{y}}])^2 + \text{Var}[\tilde{\boldsymbol{y}}] + \sigma^2 + \mathbb{E}\left[2(\boldsymbol{f} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])(\tilde{\boldsymbol{y}} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])\right] \tag{56}$$

$$= (\text{Bias}[\tilde{\boldsymbol{y}}])^2 + \text{Var}[\tilde{\boldsymbol{y}}] + \sigma^2 + 2\mathbb{E}\left[\boldsymbol{f}\tilde{\boldsymbol{y}} - \boldsymbol{f}\mathbb{E}\left[\tilde{\boldsymbol{y}}\right] - \tilde{\boldsymbol{y}}\mathbb{E}\left[\tilde{\boldsymbol{y}}\right] + \mathbb{E}\left[\tilde{\boldsymbol{y}}\right]^2\right] \tag{57}$$

$$= (\text{Bias}[\tilde{\boldsymbol{y}}])^2 + \text{Var}[\tilde{\boldsymbol{y}}] + \sigma^2 + 2\mathbb{E}[f]\mathbb{E}[\tilde{y}] - 2\mathbb{E}[f]\mathbb{E}[\mathbb{E}[\tilde{y}]] - 2\mathbb{E}[\tilde{y}]\mathbb{E}[\mathbb{E}[\tilde{y}]] + 2\mathbb{E}[\mathbb{E}[\tilde{y}]^2]. \tag{58}$$

$\boldsymbol{f}$ and $\tilde{\boldsymbol{y}}$ are independent and the expectation value of an expectation value is just the expectation value, meaning $\mathbb{E}[\mathbb{E}[\tilde{y}]] = \mathbb{E}[\tilde{y}]$ and $\mathbb{E}[\mathbb{E}[\tilde{y}]^2] = \mathbb{E}[\tilde{y}]^2$. We then see that the last terms cancel and we end up with the following expression for the cost function,

$$\mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right] = (\text{Bias}[\tilde{\boldsymbol{y}}])^2 + \text{Var}[\tilde{\boldsymbol{y}}] + \sigma^2. \tag{59}$$