FYS-STK 4155 - Project 1

Sara Pernille Jensen, Daniel Johan Aarstein and Håkon Lindholm
(Dated: October 11, 2021)
(Github repository: `https://github.com/SaraPJensen/FYS-STK/tree/main/Project1`)

INTRODUCTION

Supervised machine learning is a tool for analysing data sets and modelling the behaviour of a system. The aim is to model the dependency of a set of dependent variables on a set of independent variables. The model is trained using a subset of the data containing information about the independent features of the system and a set of response variables. To evaluate the reliability of the model, the prediction generated by the model is tested against a never-before-seen subset of the data using statistical properties such as mean-squared-error, variance and bias. In this project, three different methods for linear regression were studied; Ordinary Least Squares (OLS), Ridge and Least Absolute Shrinkage and Selection Operator (LASSO). Better predictions can sometimes be obtained if the data is somehow pre-processed, such as by scaling it. Another tool for improving the analyses is by resampling the data-sets to generate several models and taking the average of these. In this project, two different data-sets were used to generate and compare different regression methods using the above-mentioned methods of scaling and resampling. The different methods will be discussed in more detail throughout the report. In addition, the risks related to under- and over-fitting of the model will be discussed in relation to the so-called bias-variance trade-off.

PRELIMINARIES

Model Fitting

The starting point for any machine learning analysis is a data-set consisting of independent variables, $\boldsymbol{x}$, and dependent response variables, $\boldsymbol{z}$, both of length n. The input variables contain the features believed to be relevant to the response variables. The aim is then to generate a model $\tilde{\mathbf{z}}$ which can be used to predict the dependent variables given a new set of input variables. In linear regression, one makes the assumption that the model can be written as:

$$\tilde{\mathbf{z}} = f(\mathbf{x}) + \epsilon,$$

where f is some linear function of the input variables and ϵ is some noise which is assumed to be normally distributed and have a mean of 0.

To generate the model, a design matrix is constructed from the input variables. Given p features and n datapoints, a design matrix of dimension $n \times p$ is constructed. This can be done in many ways, depending on the assumptions made about the dependency of the response variables on the features. In this project, the Vandermonde matrix was used. The two data-sets studied both contained two independent variables, $\boldsymbol{x}$ and $\boldsymbol{y}$, and one dependent variable, $\boldsymbol{z}$, so the design-matrix was given by the following expression:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & y_1 & {x_1}^2 & x_1y_1 & \dots & {x_1}^p & {x_1}^{p-1}y_1 & \dots & {y_1}^p \\ 1 & x_2 & y_2 & {x_2}^2 & x_2y_2 & \dots & {x_2}^p & {x_2}^{p-1}y_2 & \dots & {y_2}^p \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & y_n & {x_n}^2 & x_ny_n & \dots & {x_n}^p & {x_n}^{p-1}y_n & \dots & {y_n}^p \end{bmatrix}$$

It is then assumed that the function $f(x, y)$ is the matrix product of the design matrix with a column vector $\boldsymbol{\beta}$, of length p, the elements of which give the weights of the different elements of $\boldsymbol{X}$ in the model. These are called the *estimators* of the model. The function is then given by:

$$f(x, y) = \mathbf{X}\boldsymbol{\beta}$$

Since the noise is unknown, this becomes the model for $\tilde{\mathbf{z}}$. "Fitting the model" then refers to finding the optimal values of the estimator $\boldsymbol{\beta}$. This is done by minimising the so-called *cost function*, and the expression for this function depends on the regression model used. This will be elaborated on in later sections.

The outline above gives the method for generating the model, but does not include the crucial step of splitting the data into test- and training-data which is needed to test the accuracy of the model. It is common in machine learning to split the data into a test- and training set (also called out-of-sample and in-sample data), where the majority of the data is used for training the model and the remainder used to test it. The exact ratio differs, and in this project a ratio of 20/80 was used consistently.

The splitting is most easily done after the design matrix has been generated, giving the matrix $\mathbf{X}_{train}$ corresponding to the response variables $\mathbf{z}_{train}$ and the matrix $\mathbf{X}_{test}$ corresponding to the response variables $\mathbf{z}_{test}$. The model is then given by:

$$\tilde{\mathbf{z}} = \mathbf{X}_{train}\boldsymbol{\beta}$$

and is tested against $\mathbf{z}_{train}$, whereas the prediction is given by:

$$\mathbf{z}_{pred} = \mathbf{X}_{test}\boldsymbol{\beta},$$

and is tested against $\mathbf{z}_{test}$.

Model Evaluation

A common measure of the accuracy of the model is the mean squared error (MSE), which is defined as follows:

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

z in this case only represents the training-subset. Similarly, the accuracy of the prediction, which is what one is usually interested in, is given by the MSE for $\mathbf{z}_{pred}$ and $\mathbf{z}_{test}$. The optimal MSE-score is 0.

Another measure of the accuracy of the prediction is the R^2 score. For a perfect prediction, this takes the value 1. It is defined as follows:

$$R^2(\mathbf{z}, \tilde{z}) = 1 - \frac{\sum_{i=0}^{n-1}(z_i - \tilde{z}_i)^2}{\sum_{i=0}^{n-1}(z_i - \langle z\rangle)^2}$$

where the mean value of $\mathbf{z}$ is given by:

$$\langle z\rangle = \frac{1}{n}\sum_{i=0}^{n-1} z_i.$$

Again, this is either calculated between $\tilde{z}$ and $\mathbf{z}_{train}$ or between $\mathbf{z}_{pred}$ and $\mathbf{z}_{test}$.

The Franke Function

The first data-set studied in this project consist of two independent input variables x and y, both defined on the domain [0, 1]. The dependent variable z is given by:

$$z = F(x, y) + \alpha\epsilon,$$

where the function F(x, y), called the Franke function, is defined as follows:

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

The noise ϵ is normally distributed with a mean of zero, whereas α is a constant giving the strength of the noise. The effect of changing this will be studied.

Scaling

It can sometimes be useful to scale the data, especially for data-sets with high variance. Four different scaling methods were tested for the two data-sets analysed, all part of the Sklearn preprocessing library. These were standard scaler, minmax scaler, robust scaler and mean scaling. The standard scaler ensures that the mean value is zero and the variance one for each feature in the design matrix. The minmax scaler ensures that the values of the features in the design matrix are between 0 and 1. The robust scaler has a similar effect to the standard scaler, but ignores outlying data-points. The mean scaler subtracts the mean value from the features without dividing by the standard deviation.

PROBLEM 1: OLS ON THE FRANKE FUNCTION

Theory

The aim of the ordinary least squares method for linear regression is to minimise the sum of the squared differences between the training data and the model, i.e. the L_2-norm of the difference between z and $\boldsymbol{X}\boldsymbol{\beta}$. Formally, the *optimal* estimators

$\hat{\boldsymbol{\beta}}$ (denoted with a hat) are those that minimise the cost function, i.e.:

$$\hat{\boldsymbol{\beta}}_{OLS} = \text{argmin}||\mathbf{z}_{train} - \mathbf{X}_{train}\boldsymbol{\beta}||_2^2$$

By taking the derivative with respect to $\boldsymbol{\beta}$, the following analytical expression of $\hat{\boldsymbol{\beta}}_{OLS}$ is obtained:

$$\hat{\boldsymbol{\beta}}_{OLS} = (\boldsymbol{X}^{\boldsymbol{T}}_{train}\mathbf{X}_{train})^{-1}\boldsymbol{X}^{\boldsymbol{T}}_{train}\mathbf{z}_{train}$$

The model and prediction is then calculated following the procedure given in the preceding section, and the results were evaluated.

Results

Using a seed to ensure that the random values generated were the same for each run, the errors on the models and predictions obtained from OLS for polynomial degree up to 5 were calculated. The noise-coefficient α was initially set to 0.05, and the number of data-points to 400. The results are presented in Table I. As can be seen, the out-of-sample errors are consistently higher than the in-sample errors, as can be expected, since the model is trained using only the training-data. However, this discrepancy is reduced as the complexity of the model increases, and the errors on the test- and training-data converges. This tendency can not, however, be extrapolated to any degree of complexity, as will be discussed in the next section. It was found that as the number of data-points is increased, the out-of-sample error is reduced, whereas the in-sample error increases. Furthermore, the out-of-sample and in-sample errors converges even for lower polynomial degrees. Assuming the true function is sufficiently complicated that it cannot be known perfectly (which is what the random noise ensures), this is as expected. As the number of data-points in the training set increases, it becomes impossible for the model to account for all the random fluctuations, so the in-sample error increases. On the other hand, a greater data-set narrows the distance between the data-points in the test- and training-set, making it more likely that points nearby the test-data have been included in the model, making the prediction more accurate. In fact, in the limit of an infinite data-set, the two errors converges to what is called the true *bias* of the model [1]. This is the best possible error which our model could obtain, given the sampling noise. Our findings correspond well with this.

Table I. MSE and R^2 scores for test- and train-data for polynomial degrees up to 5.

Complexity	MSE train	MSE test	R^2 train	R^2 test
1	0.0275	0.0286	0.6744	0.6456
2	0.0200	0.0209	0.7637	0.7403
3	0.0102	0.0108	0.8799	0.8664
4	0.0067	0.0070	0.9214	0.9135
5	0.0046	0.0046	0.9455	0.9425

If the noise is increased, both the errors and the discrepancy between the out-of-sample and in-sample errors increase with increasing levels of noise, since this makes the data harder to model and thus both the model and the prediction less accurate.

Confidence Intervals

Seeing as the data is normally distributed about its true value and is unbiased, it is known [4] that we may calculate the $100(1-a)\%$ confidence interval for the ith beta parameter by

$$\hat{\beta}_i \pm t_{a/2,n-(k+1)} \cdot s_{\hat{\beta}_i}$$

where $\hat{\beta}_i$ is the unbiased expected value for β_i, t is the t-distribution and $s_{\hat{\beta}_i}$ is the standard error in our sample of $\hat{\beta}_i$'s. This in turn is simply given by $\sqrt{\text{var}(\boldsymbol{\beta_i})} = \sqrt{\alpha^2\sigma^2(\boldsymbol{X^T X})^{-1}_{ii}} = \alpha\sqrt{(\boldsymbol{X^T X})^{-1}_{ii}}$.

By a simple OLS regression analysis we obtain the confidence intervals for the different parameters as shown in Figure 1.

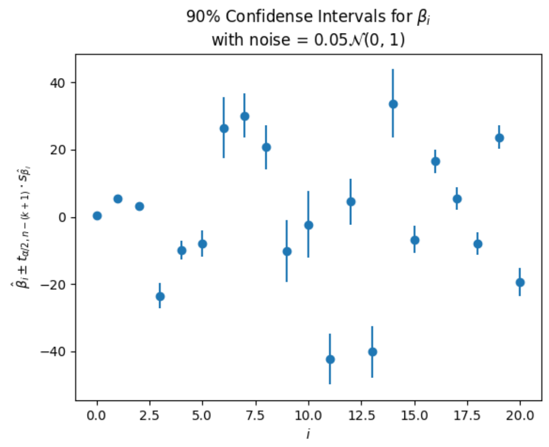

Figure 1. The expectation values as well as the confidence intervals for the 21 parameters which are fitted for a fifth-degree polynomial

Comparison with SciKit-Learn

In order to ensure the validity of our results, our manually implemented functions were compared with the equivalent ones provided by the python library SciKit-Learn. The results are presented in Table II.

Table II. MSE and R^2 scores for the manual OLS regression and SciKit-learn's OLS regression for polynomial degrees up to 10, noise = 0.15

Complexity	Manual MSE	SK-learn MSE	Manual R^2	SK-learn R^2
1	0.046404	0.046404	0.56598	0.56598
2	0.036826	0.036826	0.65556	0.65556
3	0.027741	0.027741	0.74054	0.74054
4	0.028201	0.028201	0.73624	0.73624
5	0.024655	0.024655	0.7694	0.7694
6	0.024399	0.024399	0.77179	0.77179
7	0.022447	0.022447	0.79005	0.79005
8	0.023179	0.02318	0.7832	0.7832
9	0.023655	0.023655	0.77875	0.77876
10	0.025921	0.025642	0.75756	0.76017

Effects of Scaling

The four scaling methods discussed above, along with no scaling, were tested for each data-set. These were the standard scaler, minmax scaler, robust scaler, mean scaling and no scaling. Table III shows the in-sample and out-of-sample MSE and R^2 using OLS with a polynomial of 5th degree for all the different scalers.

Table III. In-sample and out-of-sample MSE and R^2 scores using different scalers, noise = 0.05

Scaler	MSE train	MSE test	R^2 train	R^2 test
Standard Scaler	0.0545	0.0548	0.9454	0.9425
MinMax Scaler	0.0041	0.0051	0.9146	0.8894
Mean	0.0046	0.0046	0.9455	0.9425
Robust Scaler	0.0459	0.0532	0.9283	0.9127
No scaling	0.0046	0.0046	0.9455	0.9425

These results show that the mean scaling yields the same result as not scaling the data, which is expected in this case, given the distribution of the data-set. The standard, robust and minmax scalers all yield the same or worse results for both out-of-sample MSE and R^2 compared to no scaling. We therefore conclude that scaling is not necessary for the Franke function, and the data will not be scaled in Problems 2-5.

PROBLEM 2: BIAS-VARIANCE TRADE-OFF AND BOOTSTRAPPING

Theory

An important variable needed in generating the model is the *complexity* of the model. What feature of the model this complexity refers to differs from model to model, but in the case of linear regression, the degree of the polynomial used to fit the model is a good measure of its complexity. In general, the higher the complexity, the better the fit of the model with the training data will be. If the complexity is too low, there is a risk of *under-fitting*, meaning that the model is too simple to account for much of the data. However, if the complexity is too high, there is a risk of *over-fitting*. This happens when the model attempts to go through all the data-points by force instead of capturing the overarching pattern of the data, making it unable to predict new data-points. To find the optimal level of complexity, one must look at the error from the test data. This is expected to decrease up to a certain level of complexity, after which it increases as a result of over-fitting. This trade-off between the in-sample and out-of-sample error as a function of complexity was studied for OLS regression for the Franke function.

The cause of the increase in the out-of-sample error can be studied further by decomposing the MSE into the bias and the variance of the model. The increasing error can then be studied in terms of the *bias-variance trade-off* of the model. This trade-off is essential to understanding the difficulties with machine learning. The bias of the model is a measure of the deviation of the expectation value of the estimators β_i (i.e. their asymptotic value in the limit of an infinite data-set) from the true value. The variance is a measure of the fluctuations in the estimator $\hat{\beta}$ of the model resulting from the finiteness of the sample [1]. In general, the bias decreases with the complexity of the model, whereas the variance increases. See the Appendix for a derivation of the bias-variance decomposition of the MSE. Since data is often limited, it can therefore be better to use a less complex model with higher bias, in order to avoid high variance.

To study the bias-variance trade-off, and to get more reliable information about the reliability of the model, it is useful to use *resampling* techniques. These work by taking an average of the results obtained from fitting models to the same data-set, but with slightly different training-data. The two methods which will be covered in this project are called *bootstrapping* and *k-fold cross-validation*.

In this section we shall look at bootstrapping. In bootstrapping, the data-set is divided into test- and training-data as before. However, each model generated is trained on a slightly different subset taken from the training-data. For each run, a sample data-set is randomly drawn, with replacement, from the training-data. Each sample is of the same size as the original training data-set, and a new model, giving a new set of predictions, is generated for each run. This is done B times. Importantly, the same train- and test-data is used for each run and for each polynomial degree to ensure a reliable comparison between degrees. This was ensured by initially calculating the design matrix for the highest polynomial degree and splitting the test- and training-data before looping over the different degrees of model complexity. The required subset of the design matrix is then selected for each run. Each prediction is compared with the out-of-sample-data, and an estimate of the bias and variance is calculated from the average values.

Results

Keeping the noise coefficient α at 0.05 and the number of data-points at 400, the effect of increasing complexity on in-sample and out-of-sample MSE was studied further. As mentioned in the problem above, the in-sample error is expected to keep decreasing with increasing complexity, since this error is a measure of the model's ability to pass through all the data-points in the training-data. The out-of-sample error, however, is a measure of the model's ability to predict unseen data-points, and this is only expected to decrease up to a certain level of complexity, after which the model becomes over-fitted and the out-of-sample error begins to increase. To study this effect, a plot of the in-sample and out-of-sample MSE for increasing complexity was plotted for OLS regression, as shown in Figure 2. As can be seen, the out-of-sample and in-sample MSE converges for low levels of complexity, before they start diverging and the out-of-sample MSE begins to fluctuate, meaning that the models at higher complexity are less reliable and are on average less able to predict unseen data. The minimum out-of-sample MSE was obtained using a polynomial of degree 8, giving an error of 0.00314.

A problem with the above results and discussion

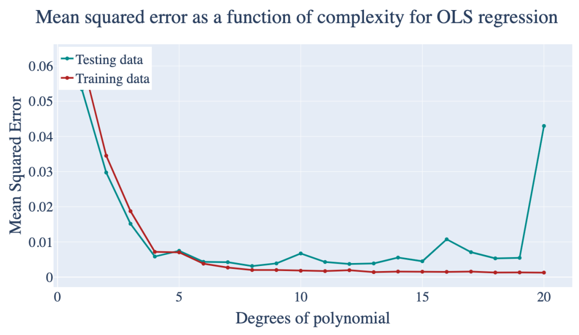

Figure 2. MSE of test and training data as a function of model complexity for a single run of OLS regression with a data-set of 400 data-points.

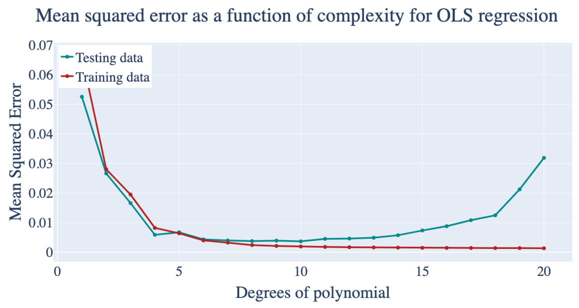

Figure 3. MSE of test and training data as a function of model complexity for OLS regression using bootstrapping with 100 runs.

is that they are based on a single set of models, where only one model is calculated for each polynomial degree. This makes the conclusions drawn hard to generalise. In order to obtain more reliable measures, it is necessary to take the averages over several models fitted on the same data-set. This was done using the bootstrap resampling technique with $B = 100$.

Figure 3 shows the in-sample and out-of-sample MSE for a data-set of 400 points for OLS using bootstrap. The same tendency was found here as without bootstrapping, suggesting that the above model and errors were representative of what could be obtained given the data-set. The minimum out-of-sample MSE is now found at polynomial degree 10 with an error of 0.00366.

It is important to note that the point at which of over-fitting occurs is highly dependent on the number of data-points, not just on the true source function. To see this, the same calculation as above was done, but this time for a data-set containing 900 data-points. The plot showing the in-sample and out-of-sample MSE as a function of complexity is shown in Figure 4. As expected from the discus-

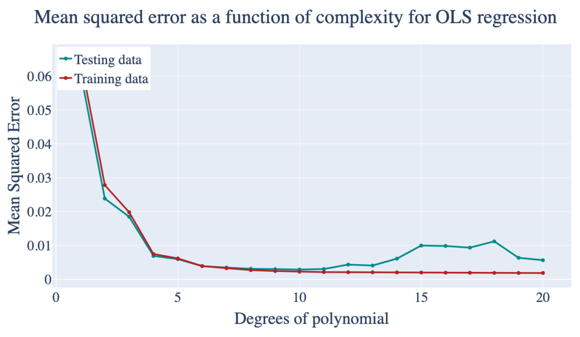

Figure 4. MSE of test and training data as a function of model complexity for OLS regression with a data-set of 900 data-points and using bootstrapping with 100 runs.

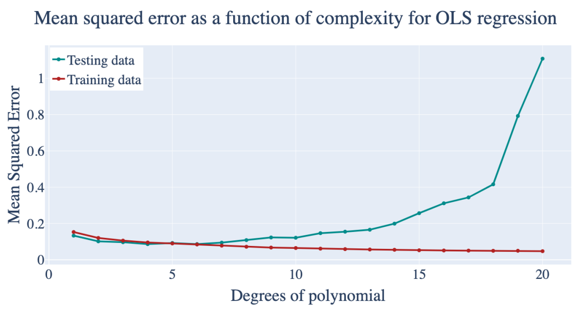

Figure 5. MSE of test and training data as a function of model complexity for OLS regression with the noise-coefficient set to 0.3 using bootstrapping with 100 runs.

sion above, the out-of-sample MSE is lower when the data-set is larger, especially for higher degrees of complexity. This shows that the greater data-set makes the model less prone to over-fitting for the same level of complexity. This is because the larger data-set makes it less likely that outliers contribute significantly to the model, which is what leads to the over-fitting, as well as the test-data being closer in distance to the training-data. Furthermore, although this is hard to see just from looking at the graphs, the in-sample MSE is higher for the greater data-set, as expected from the discussion above. Note, however, that the best out-of-sample MSE is still found at polynomial degree 10, now with an MSE of 0.00289.

Another factor contributing to the over-fitting and what the optimal level of complexity of the model is will be the amount of noise in the data-set. This was investigated by changing the noise-coefficient to $\alpha = 0.3$. Again using 400 data-points, the in-sample and out-of-sample MSE was calculated for OLS using bootstrapping as before. The resulting errors are shown in Figure 5. Here the absolute value of both errors are consistently higher.

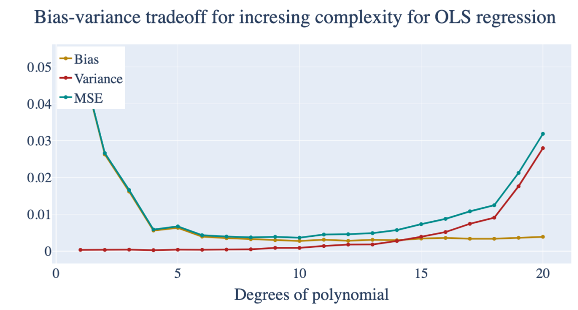

Figure 6. MSE, bias and variance as a function of model complexity for OLS regression using bootstrapping with 100 runs. This shows the trade-off between bias and variance with increasing complexity.

More interestingly, the over-fitting becomes significant at lower levels of complexity than before, and the lowest out-of-sample MSE is now found at polynomial degree 4, with an MSE of 0.08619.

As discussed above, the out-of-sample MSE can be decomposed into the bias and variance of the prediction. The bias-variance trade-off was calculated and plotted for the same data-set, again using bootstrapping with 100 repetitions. The resulting plot is shown in Figure 6. As expected, the bias decreases with increasing complexity, whereas the variance increases, showing that there is a trade-off between the two as the complexity of the model increases.

PROBLEM 3: K-FOLD CROSS-VALIDATION

Theory

k-fold cross validation is another commonly used resampling technique in machine learning. Here the data is divided into k partitions of equal size. $k-1$ of the partitions are then used as training data to fit the model while the last part is reserved in order to test the model, resulting in an out-of-sample MSE score. This process is repeated k times, such that every partition is used for testing once, so the model generates k different MSE values. The total cross-validation estimate for the MSE is given by the mean of all these values, i.e.

$$MSE_{CV} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

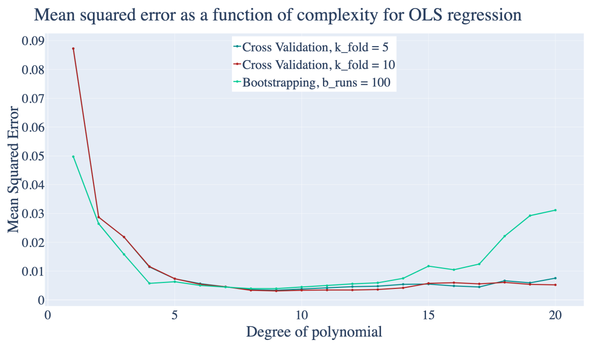

Figure 7. MSE as a function of model complexity for OLS regression using cross validation and bootstrapping with the noise coefficient set to 0.05.

Results

The results in Figure 7 were obtained by using a noise coefficient of 0.05. Plots for cross validation with 5 and 10 folds are compared to a run with 400 bootstrap cycles. We first note that the errors for a fourth degree polynomial is lower for bootstrapping than for either of the cross validation errors. We also see that for higher degree polynomials, the bootstrap resampling is over-fitting, while the cross validation curves are more stable. When comparing the two methods, it is important to note that although they use the same data-set, they use a different split between the train and the test data. This is a potential source of error in the comparison.

Figure 8 is the same analysis done in Figure 7, with a noise coefficient of 0.5. The minimum error is achieved with a 6th degree polynomial for cross validation with 10 folds. We also see that both resampling methods are over-fitted as opposed to when the noise coefficient was 0.05. For either noise coefficient, the lowest error estimate was achieved by the 10-fold cross validation. This is likely the result of 10-fold using 90% of the data for training, rather than 80%. This likely gives a better model, but at the risk of not having enough out-of-sample data to test it properly.

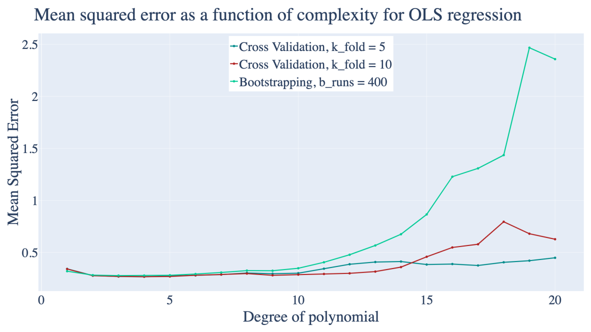

Figure 8. MSE as a function of model complexity for OLS regression using cross validation and bootstrapping with the noise coefficient set to 0.5.

PROBLEM 4: RIDGE REGRESSION

Theory

In Ridge regression, a regularisation parameter λ is added to the expression to be minimised in OLS, acting as a penalty on the L_2-norm of the estimator-vector $\hat{\boldsymbol{\beta}}$. The optimal estimators are then given by the minimisation problem:

$$\hat{\boldsymbol{\beta}}_{Ridge} = \text{argmin}(||\mathbf{z}_{train} - \mathbf{X}_{train}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2)$$

giving the following analytical expression for $\hat{\boldsymbol{\beta}}_{Ridge}$:

$$\hat{\boldsymbol{\beta}}_{Ridge} = (\boldsymbol{X}_{train}^{\boldsymbol{T}}\mathbf{X}_{train} + \lambda\mathbb{1})^{-1}\boldsymbol{X}_{train}^{\boldsymbol{T}}\mathbf{z}_{train}$$

$\mathbb{1}$ is the identity matrix with dimensions $p \times p$. λ is the parameter to be tuned to obtain the optimal MSE. By penalizing the larger values in the estimator $\hat{\boldsymbol{\beta}}_{Ridge}$, the model will become more stable. After some threshold, the introduced complexity is penalized to the point where it essentially no longer contributes, and this ensures the stability of the model even at high levels of complexity [3]. The main consequence of this is that Ridge regression is far less prone to over-fitting than e.g. OLS regression.

Results

To study the effect of using different values of λ in Ridge regression, a plot of the out-of-sample error for different values of λ ranging from 0 to 10 was generated. Bootstrapping with 100 repetitions and no scaling was used. The resulting plot is shown in Figure 9. As can be seen, the out-of-sample error is much more resistant to over-fitting when Ridge regression is used, and the out-of-sample quickly flattens out and stays constant. This is to be expected, since part of the function of the regularisation parameter is to prevent over-fitting. Furthermore, the lowest error is obtained for the lower values of λ, the lowest where $\lambda = 0$, which is simply OLS regression. As before, the best fit was obtained for polynomial degree 10.

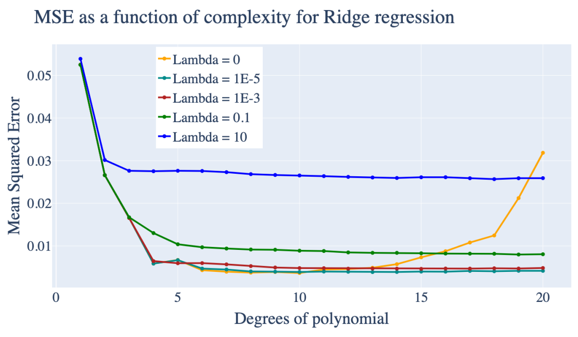

Figure 9. A plot of out-of-sample error as a function of model complexity for Ridge regression for different values of the regularisation parameter λ using bootstrap, including $\lambda = 0$, corresponding to OLS regression. The noise coefficient was set to 0.05.

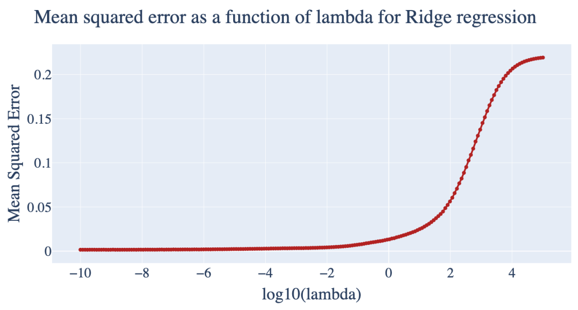

Figure 10. Out-of-sample error for Ridge regression as a function of the regularisation parameter λ for polynomial degree 10 using bootstrap, with λ ranging from 10^{-10} to 10^{5} and the noise coefficient set to 0.05.

To study the dependence on λ in more detail, a plot was then made of out-of-sample MSE as a function of λ for a constant level of complexity (polynomial degree 10) and for λ ranging from 10^{-10} to 10^{5}. This is shown in Figure 10. This confirms the earlier finding that for such a level of model complexity, where OLS does not over-fit, the lowest error is obtained in the limit where λ goes to zero, i.e. for OLS regression.

A similar analysis was done using 10-fold cross validation as resampling method. Since cross validation is faster to run than bootstrap, a contour plot with different polynomial degrees and lambdas was made. The resulting plot is shown in Figure 11. The lowest MSE is now found at polynomial degree 18 for $\lambda = 0.00002$. One potential reason for the difference in results is that cross validation can go to far higher complexity without over-fitting. However, considering how close this point is to the lowest λ in the range, this might well be the result of a random fluctuation.

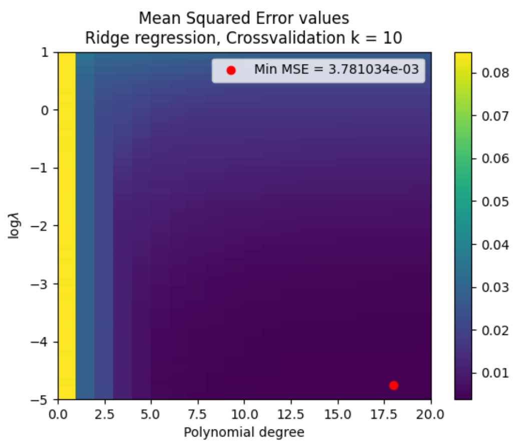

Figure 11. Out-of-sample error for Ridge regression as a function of the regularisation parameter λ and complexity using 10-fold cross validation with the noise coefficient set to 0.05.

The conclusion was then that OLS regression is better than Ridge regression for the Franke function with a noise coefficient of 0.05. The above analysis was then repeated for higher levels of noise, and when α was increased to 0.5, it was found that Ridge begins to outperform OLS. A similar plot as shown in Figure 9 was done for the higher level of noise, though only for polynomial degrees up to 10, as the OLS regression quickly starts to over-fit. The resulting plot is shown in Figure 12. The lowest error is now obtained at polynomial degree 7, with $\lambda = 0.1$. The error was then 0.23198. To study the dependency on λ further, a plot of the error as a function of λ for polynomial degree 7 with λ in the same range, but with far more points, as above was plotted. This is shown in Figure 13. As can be seen, the dependency on λ has changed with the increased level of noise, and the minimum MSE of 0.23132 is now obtained for $\lambda = 0.15703$. This suggests that Ridge might be more useful when fitting models to data-sets with more noise. This is likely because OLS is more prone to over-fitting when there is more noise in the data, which is what Ridge regression prevents.

To take a final look at the stability of the Ridge regression solution and its resistance to over-fitting, the bias-variance trade-off was plotted for the same noise using the optimal $\lambda = 0.15703$. The resulting plot is shown in Figure 14. This is clearly different from the similar plot obtained from OLS regression, as shown in Figure 6. Whereas for OLS the bias is reduced and the variance increased for increasing levels of complexity, eventually leading to a significant increase in the out-of-sample error, the bias

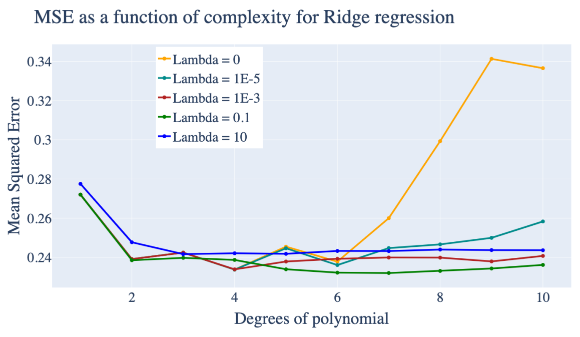

Figure 12. A plot of out-of-sample error as a function of model complexity for Ridge regression for different values of the regularisation parameter λ, including $\lambda = 0$, corresponding to OLS regression. The noise coefficient was set to 0.5.

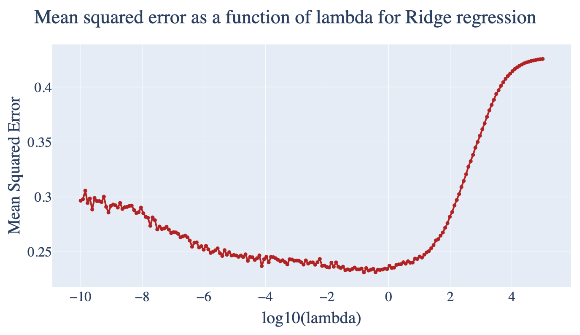

Figure 13. Out-of-sample error for Ridge regression as a function of the regularisation parameter λ for polynomial degree 7, with λ ranging from 10^{-10} to 10^{5} and the noise coefficient set to 0.5.

and variance are near constant for Ridge. This explains Ridge's resistance to over-fitting, and shows that it comes at the cost of not reducing the bias even for higher levels of complexity. It should be noted that the variance does increase slightly, along with the MSE, whereas the bias stays near constant.

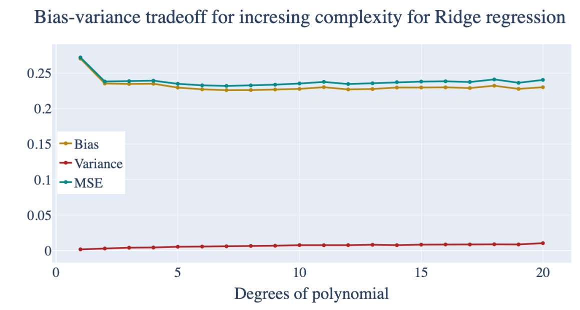

Figure 14. Bias-variance trade-off plotted for Ridge using $\lambda = 0.15703$ with the noise coefficient set to 0.5.

Again, the same analysis, for the same level of noise, was done using cross validation. The resulting plot is shown in Figure 15. The lowest MSE is now found at for $\lambda = 0.00091$ at polynomial degree 7. The optimal polynomial is the same as was found using bootstrap, whereas the value of λ is different. This discrepancy could be a result both of the different split of train/test data used in the different techniques or of the different split size, where bootstrap uses 80% to train whereas 10-fold cross validation uses 90% to train. Ridge was nevertheless found to be better than OLS with both resampling methods.

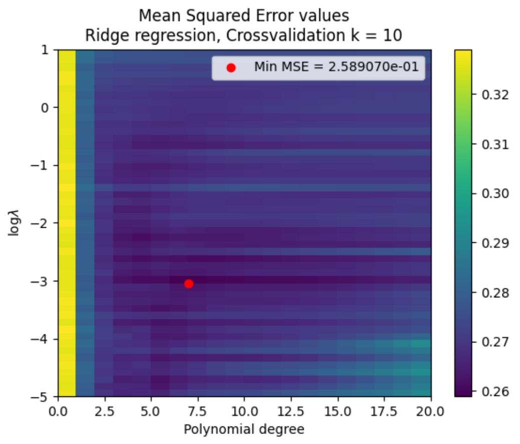

Figure 15. Out-of-sample error for Ridge regression as a function of the regularisation parameter λ and complexity using 10-fold cross validation with the noise coefficient set to 0.5.

Finally a comparison between the manually implemented Ridge regression and the equivalent function provided by SciKit-Learn was conducted in order to ensure the validity of our results. This was done by calculating the MSE for every combination of polynomial degree and λ-value for each function, and then comparing the two results. By letting the polynomial degree go up to 20 and testing for 500 values for λ ranging from 10^{-4} to 10 for each degree, the largest difference between our implemented method and SciKit-Learns was found at polynomial degree 20 and $\lambda = 1.12 \cdot 10^{-4}$. The largest difference in MSE was measured to be $3.899 \cdot 10^{-12}$.

PROBLEM 5: LASSO REGRESSION

Theory

Least Absolute Shrinkage and Selection Operator (LASSO) regression works similarly to Ridge regression by introducing a regularization parameter

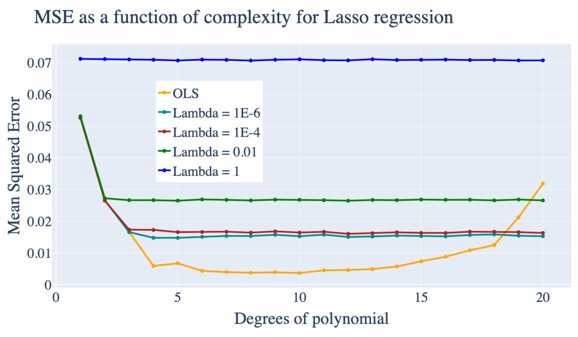

Figure 16. A plot of out-of-sample error as a function of model complexity for LASSO regression for different values of the regularisation parameter λ, along with OLS regression. The noise coefficient was set to 0.05.

λ in the minimisation problem to find the optimal estimator-vector $\hat{\boldsymbol{\beta}}$. In the case of LASSO regression, this parameter works as a penalty on the L_1-norm of the estimator-vector.

$$\hat{\boldsymbol{\beta}}_{LASSO} = \text{argmin}(||\mathbf{z}_{train} - \mathbf{X}_{train}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1)$$

Unfortunately, this problem cannot be solved analytically, so it must be solved numerically. In this case, scikit-learn's built in functionality for LASSO-regression was used.

Similarly to Ridge regression, Lasso regression becomes stable after a threshold in the complexity of the model. As mentioned, this is because of the regularization parameter. A key difference between the Ridge and Lasso is that Lasso will penalize some estimators to zero, whereas the Ridge regression only will make them arbitrarily close to zero [3].

Results

The same analysis as was done for Ridge regression was then done for LASSO regression. The noise coefficient was first set to 0.05, using bootstrap with 100 repetitions, a plot of the out-of-sample MSE for different values of λ was plotted. Similar to Ridge, LASSO is less prone to over-fitting at higher levels of complexity, but is outperformed by OLS regression for lower levels. This is also where the error is lowest, so OLS is again found to be the preferred method of regression. As for Ridge, the lack of over-fitting is a result of the regularization parameter λ, which keeps the variance low at the cost of a higher bias, meaning that there is no clear bias-variance tradeoff. The out-of-sample

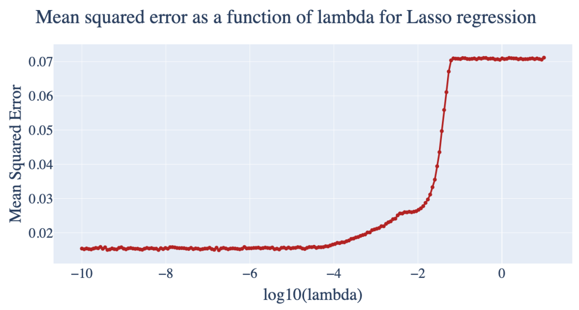

Figure 17. Out-of-sample error for Ridge regression as a function of the regularisation parameter λ for polynomial degree 10, with λ ranging from 10^{-10} to 1 and the noise coefficient set to 0.05.

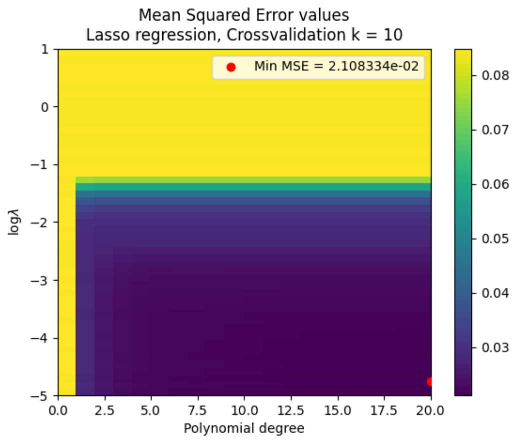

Figure 18. Out-of-sample error for LASSO regression as a function of the regularisation parameter λ and model complexity, with λ ranging from 10^{-5} to 1 and the noise coefficient set to 0.05.

error as a function of λ for polynomial degree 10 and λ in the range 10^{-10} to 10 was also plotted, as shown in Figure 17. This supports the assumption that the lowest error is found in the limit where λ approaches 0, corresponding to OLS regression.

The results obtained using bootstrap was then compared with 10-fold cross validation, with the resulting plot shown in Figure 18. Again, the optimal complexity is higher for cross validation than for bootstrap, being at polynomial degree 20. The optimal λ is now found to be 0.00002, but given how close this is to the minimum, it is uncertain whether this is simply a fluctuation or a sign that LASSO is preferred, especially since OLS regression is not included in this plot.

The noise coefficient was then increased to 0.5 as in the comparison with Ridge. A plot of the out-of-sample error as a function of complexity for up to

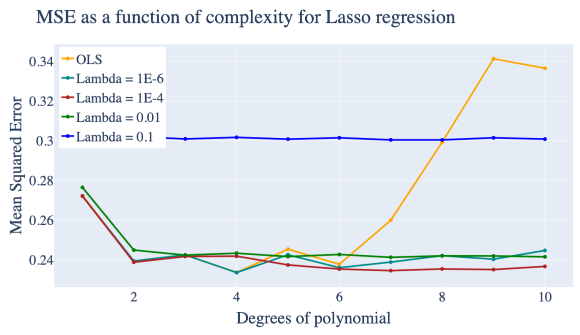

Figure 19. A plot of out-of-sample error as a function of model complexity for LASSO regression for different values of the regularisation parameter λ, along with OLS regression. The noise coefficient was set to 0.5.

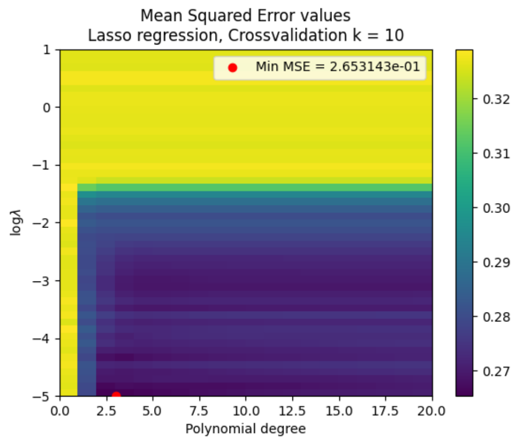

Figure 20. Out-of-sample error for LASSO regression as a function of the regularisation parameter λ and model complexity, with λ ranging from 10^{-5} to 1 and the noise coefficient set to 0.5.

polynomial degree 10 for different values of λ was plotted, as shown in Figure 19. At its best, LASSO performs equally to OLS for the lowest value of λ, which is worse than what was achieved with Ridge, so the dependence on λ was not investigated further using bootstrap. Conducting the same analysis using 10-fold cross validation as for the lower noise, similar results were found, as shown in Figure 20. Here the lowest MSE is found for the lowest λ and at polynomial degree 3, confirming the finding that OLS regression gives a better fit than LASSO for this data-set.

Comparison

The above analysis of the results obtained when fitting models to the Franke function with different levels of noise using three different methods of linear regression has shown that the optimal method is dependent on the amount of noise in the data-set. When the noise-coefficient was set to 0.05, the best fit on a data-set of 400 points was obtained using OLS regression with polynomial degree 10. Although both Ridge and LASSO were less prone to over-fitting for higher levels of complexity, the absolute error was lower for OLS at polynomial degree 10, so this was found to be the best linear regression model for such a data-set.

When the noise-coefficient was increased to 0.5, however, a different picture emerged. Ridge then gave better results than both OLS and LASSO, and was still resistant to over-fitting. The conclusion is that for a smooth function with little noise such as the Franke Function, the regularization parameter from Ridge or LASSO is unnecessary, and OLS was found to be the preferred method. As the noise is increased, however, OLS becomes more prone to over-fitting even at lower levels of model complexity, and Ridge was found to perform the best.

PROBLEM 6: ANALYSIS OF TOPOLOGICAL DATA

The second data-set analysed consisted of real terrain data from Norway[5]. Due to limited time and computing power, only a subset of 100 × 100 points was analysed. A plot of the terrain in this range is shown in Figure 21.

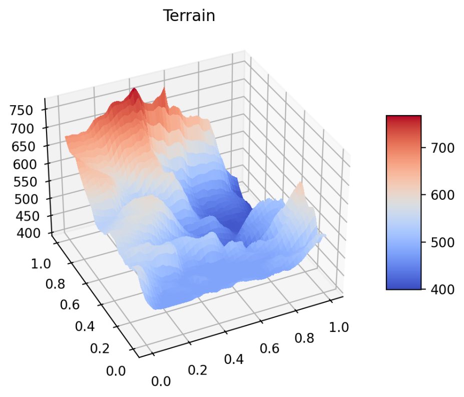

Figure 21. Visualization of the terrain-data.

Since the data in the new data-set is distributed differently to that of the Franke function, and the range in values is far greater, the analysis of the data was initialised with a test of the different scaling functions for a single run of OLS regression with polynomial degree 15. Polynomials of degrees 5, 7 and 10 were also tested, showing the same trend. The out-of sample MSE and R^2 scores for the different scalers tested are shown in Table IV. Whereas the R^2 scores are fairly similar for the different scaling methods, the MSE differs considerably between the different methods. The MinMax scaler gave the best results, so it was assumed that this would hold for the other regression methods and polynomial degrees as well, and this was used for the rest of the analysis.

Table IV. Out-of-sample MSE and R^2 scores for a single run of OLS regression of polynomial degree 15 on terrain data for different scalers.

Scaler	MSE	R^2
Standard Scaler	0.03866	0.96059
MinMax Scaler	0.00159	0.95373
Mean	201.25	0.95782
Robust Scaler	0.03201	0.95707
No scaling	207.15	0.95659

To study the bias-variance trade-off and thereby find the optimal polynomial degree for OLS regression, bootstrap with 100 resamplings was used. Polynomial degrees between 5 and 25 were plotted to avoid plotting the high error at the low end of the scale. The resulting plot is shown in Figure 22. As can be seen, for polynomial degrees up to 25, the error is mostly decreasing, meaning that no over-fitting takes place in this range. The reason is likely to be the size of the data-set, which is far greater than the one analysed for the Franke function above. The minimum MSE is found to be 0.00109 at polynomial degree 22.

Next, the effect of using Ridge regression compared to OLS regression was investigated. A plot was made for polynomial degrees between 5 and

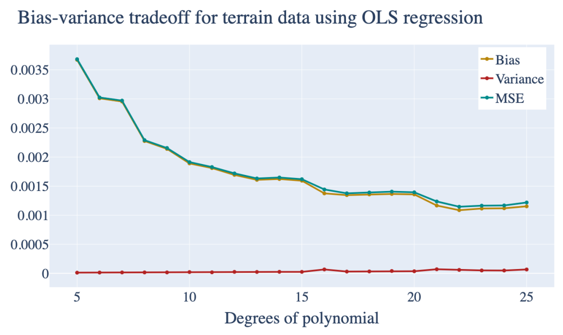

Figure 22. Bias-variance tradeoff for increasing levels of complexity using OLS regression and bootstrap on the terrain data.

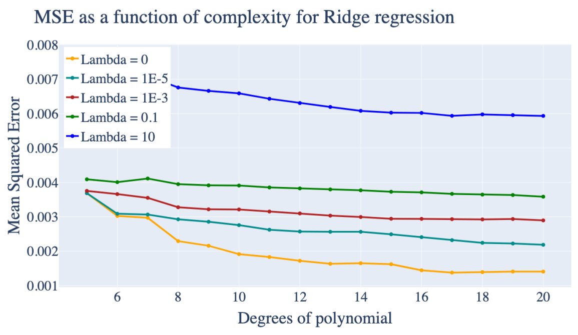

Figure 23. A plot of the out-of-sample MSE for different values of λ with Ridge regression using bootstrap to resample.

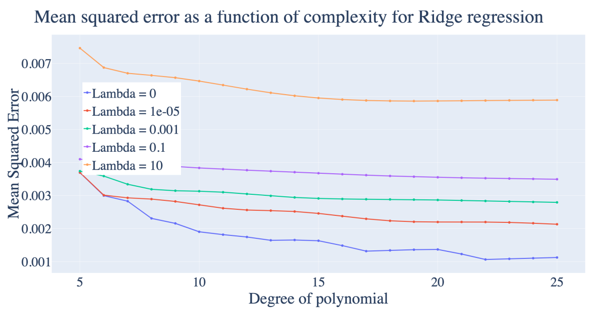

Figure 24. A plot of the out-of-sample MSE for different values of λ with Ridge regression using cross validation to resample.

20 for lambdas ranging from 10^{-6} to 10 using bootstrap with 100 resamplings. The resulting plot is shown in Figure 23. This shows that for this degree of complexity, OLS is clearly best. Due to limited computing power, higher levels of complexity were not tested, but from the results obtained from the analysis of the Franke function, there is little reason to believe that Ridge will outperform OLS at higher polynomial degrees unless this is a result of OLS's over-fitting. The same analysis was done using cross validation with 10 folds, and the results are shown in Figure 24. Because of the greater size of the data-set, a contour plot was not made in this part. As with the Franke function, the absolute values of the MSEs are lower, but the relative difference between the errors for different λs is the same as for bootstrapping, confirming the finding that OLS gives a better model of the terrain than Ridge.

LASSO regression was tested in a similar fashion, with the plot resulting from bootstrapping shown in Figure 25. Again, the same analysis was done using cross validation, shown in Figure 26. Both plots show that OLS outperforms LASSO significantly for the given range of polynomials and lamb-

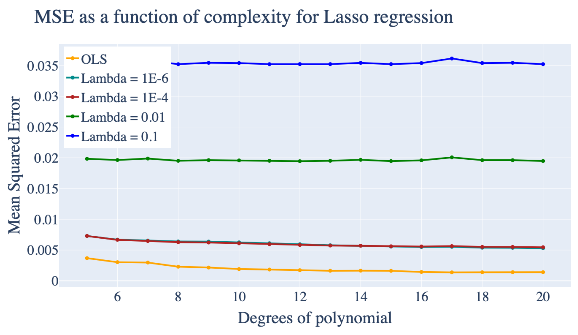

Figure 25. A plot of the out-of-sample MSE for different values of λ using LASSO regression, along with OLS, using bootstrap to resample.

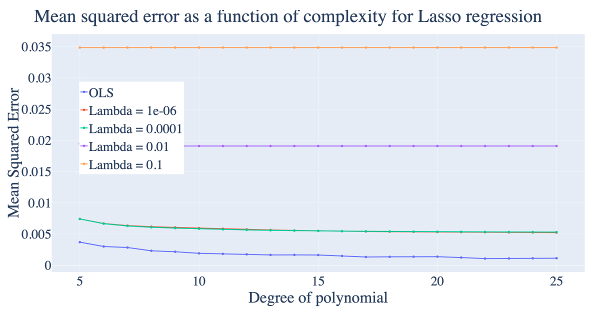

Figure 26. A plot of the out-of-sample MSE for different values of λ using LASSO regression, along with OLS, using cross validation to resample.

das, and it seems reasonable to assume that this will hold for higher polynomial degrees as well. It was therefore concluded that OLS was a better method of regression than both Ridge and LASSO when modelling the given terrain data.

Conclusion

Three methods of linear regression, OLS, Ridge and LASSO, have been tested on two different data-sets, the Franke function with different levels of noise and real terrain data. Which model performs best was found to be dependent on the data-set. For smooth distributions, such as the Franke function with low noise or the terrain data, OLS was found to be better or equal to Ridge and LASSO with low values of λ. When more noise was included in the dependent variable of the Franke function, the tendency of OLS regression to over-fit the model became significant even for lower levels of complexity. Ridge then gave better results than OLS, and LASSO gave equal results for higher values of λ than before. This shows the usefulness of the regularisation parameter in preventing over-fitting. Although testing on more data-sets would be needed to draw any firm conclusions from this, the tendency seems to be that Ridge and LASSO are more useful for data-sets with higher levels of noise, where the risk of over-fitting with OLS regression becomes significant.

[1] Mehta, Pankaj, Marin Bukov, Ching-Hao Wang, Alexandre G.R Day, Clint Richardson, Charles K Fisher, and David J Schwab. "A High-bias, Low-variance Introduction to Machine Learning for Physicists." *Physics Reports* 810 (2019): 1-124.

[2] Bishop, Christopher M. *Pattern Recognition and Machine Learning. Information Science and Statistics.* New York: Springer, 2006.

[3] Hastie, Trevor., Robert. Tibshirani, and Jerome. Friedman. *The Elements of Statistical Learning : Data Mining, Inference, and Prediction, Second Edition.* 2nd Ed. 2009. ed. Springer Series in Statistics. New York, NY: Springer New York : Imprint: Springer, 2009.

[4] Jay L. Devore and Kenneth N. Berk. *Modern Mathematical Statistiscs with Applications, Second Edition.*Springer, 2012.

[5] Terrain data `https://github.com/CompPhysics/MachineLearning/blob/master/doc/Projects/2021/Project1/DataFiles/SRTM_data_Norway_1.tif`

APPENDIX
BIAS VARIANCE TRADEOFF FOR THE MEAN SQUARED ERROR

The cost-function used in order to derive the mean squared error is the error squared and is defined as follows:

$$\begin{aligned}\mathcal{C}(\boldsymbol{z}, \tilde{\boldsymbol{z}}) &= \frac{1}{n}(\boldsymbol{z} - \tilde{\boldsymbol{z}})^T(\boldsymbol{z} - \tilde{\boldsymbol{z}}) \\ &= \frac{1}{n}\|\boldsymbol{z} - \tilde{\boldsymbol{z}}\|_2^2 \\ &= \frac{1}{n}\sum_i \left(z_i - \hat{f}(x_i)\right)^2\end{aligned}$$

where $\tilde{\boldsymbol{z}} = \hat{f}(\boldsymbol{x})$ is our approximation and $\boldsymbol{z}$ is the dependent variable in the data provided.
The mean squared error thus has the expression

$$\mathbb{E}\left[(\boldsymbol{z} - \tilde{\boldsymbol{z}})^2\right]$$

First some assumptions: Let $z = f(\boldsymbol{x}) + \epsilon$ be the true expression for z which we wish to approximate. Furthermore let $f(\boldsymbol{x})$ be a deterministic function, and let $\epsilon \sim \mathcal{N}(0, \sigma^2)$, that is ϵ is normally distributed about 0 with variance σ^2. From these assumptions it follows that $\mathbb{E}(f(\boldsymbol{x})) = f(\boldsymbol{x})$ and $\text{var}(f(\boldsymbol{x})) = 0$, as well as $\mathbb{E}(\epsilon) = 0$ and $\text{var}(\epsilon) = \sigma^2$.
Now using the vector-notation $f(\boldsymbol{x}) = \boldsymbol{f}$ and $\tilde{\boldsymbol{z}} = \hat{f}(\boldsymbol{x}) = \hat{\boldsymbol{f}}$ the mean squared error may thus be rewritten as follows:

$$\begin{aligned}\mathbb{E}\left[(\boldsymbol{z} - \hat{\boldsymbol{f}})^2\right] &= \mathbb{E}\left[(\boldsymbol{f} + \epsilon - \hat{\boldsymbol{f}})^2\right] \\ &= \mathbb{E}\left[(\boldsymbol{f} + \epsilon - \hat{\boldsymbol{f}} + \mathbb{E}[\hat{\boldsymbol{f}}] - \mathbb{E}[\hat{\boldsymbol{f}}])^2\right] \\ &= \mathbb{E}\left[(\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}])^2\right] + \mathbb{E}\left[\epsilon^2\right] \\ &\quad + \mathbb{E}\left[(\mathbb{E}[\hat{\boldsymbol{f}}] - \hat{\boldsymbol{f}})^2\right] \\ &\quad + 2\mathbb{E}\left[(\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}])\epsilon\right] \\ &\quad + 2\mathbb{E}\left[\epsilon(\mathbb{E}[\hat{\boldsymbol{f}}] - \hat{\boldsymbol{f}})\right] \\ &\quad + 2\mathbb{E}\left[(\mathbb{E}[\hat{\boldsymbol{f}}] - \hat{\boldsymbol{f}})(\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}])\right]\end{aligned}$$

Using the fact that $\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}]$ is deterministic, and the fact that $\mathbb{E}[\epsilon] = 0$ we have that

$$\begin{aligned}2\mathbb{E}\left[(\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}])\epsilon\right] &= 2(\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}])\mathbb{E}[\epsilon] \\ &= 0\end{aligned}$$

By the same argument

$$2\mathbb{E}\left[\epsilon(\mathbb{E}[\hat{\boldsymbol{f}}] - \hat{\boldsymbol{f}})\right] = 0$$

and

$$\begin{aligned}2\mathbb{E}\left[(\mathbb{E}[\hat{\boldsymbol{f}}] - \hat{\boldsymbol{f}})(\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}])\right] &= 2(\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}])\mathbb{E}\left[(\mathbb{E}[\hat{\boldsymbol{f}}] - \hat{\boldsymbol{f}})\right] \\ &= (\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}])\left[\mathbb{E}[\hat{\boldsymbol{f}}] - \mathbb{E}[\hat{\boldsymbol{f}}]\right] \\ &= 0\end{aligned}$$

Using the distributive properties of the expected value.
Since $\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}]$ is deterministic, so is $(\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}])^2$. Hence, $\mathbb{E}\left[(\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}])^2\right] = (\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}])^2$. This quantity is the bias squared, that is, the difference between the expected value of the unknown true distribution and our estimator of the same distribution, all squared.
Now using the following well known result from statistics

$$\begin{aligned}\text{var}[x] &= \mathbb{E}[x^2] - \mathbb{E}[x]^2 \\ &\iff \\ \mathbb{E}[x^2] &= \text{var}[x] + \mathbb{E}[x]^2\end{aligned}$$

we have that

$$\begin{aligned}\mathbb{E}\left[(\mathbb{E}[\hat{\boldsymbol{f}}] - \hat{\boldsymbol{f}})^2\right] &= \text{var}[\mathbb{E}[\hat{\boldsymbol{f}}] - \hat{\boldsymbol{f}}] + \mathbb{E}[\mathbb{E}[\hat{\boldsymbol{f}}] - \hat{\boldsymbol{f}}]^2 \\ &= \text{var}[\mathbb{E}[\hat{\boldsymbol{f}}] - \hat{\boldsymbol{f}}] \\ &= \text{var}[\hat{\boldsymbol{f}}]\end{aligned}$$

and

$$\begin{aligned}\mathbb{E}\left[\epsilon^2\right] &= \text{var}[\epsilon] + \mathbb{E}[\epsilon]^2 \\ &= \text{var}[\epsilon] \\ &= \sigma^2\end{aligned}$$

Thus all the terms in $\mathbb{E}\left[(\boldsymbol{z} - \hat{\boldsymbol{z}})^2\right]$ are known and we are left with

$$\begin{aligned}\mathbb{E}\left[(\boldsymbol{z} - \hat{\boldsymbol{z}})^2\right] &= (\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}])^2 + \text{var}[\hat{\boldsymbol{f}}] + \sigma^2 \\ \text{MSE} &= \text{bias}^2 + \text{variance} + \sigma^2\end{aligned}$$