FYS-STK4155 Project 1: Regression analysis and resampling methods

Sushma Sharma Adhikari, Gert Kluge and H. Alida F. Hardersen
Fysisk Institutt, Universitetet i Oslo
(Dated: October 10, 2020)

In this project we use self-implemented methods for two common Linear regression models, Ordinary Least Squares and Ridge-regression, and the scikitlearn method for Lasso-regression on a sample of real Norwegian terrain. The implementations are tested using data generated by the Franke Function before the analysis is repeated on the data. The analysis on the Franke Function showed some discrepancies when using the bootstrap resampling method, and when applying the analysis on real data it became clear that the implementation needs more work.

I. INTRODUCTION

One of the most important tasks in statistics is to find correlations between statistically random variables. In particular, one wishes to be able to predict some statistical variable (e.g. the spread of a virus amongst the members of a population) on the basis of some measurements of another statistical variable (e.g. the average age of that population). In general, such correlations are found by means of regression. In order to make effective regressions, it is important to make a reasonable model, as well as to define a meaningful criterion for the "goodness" of the model's fit to the measured data. This criterion must also be suitable in the sense that the model can be effectively "adapted" using mathematical techniques, to provide an optimal fit. The mathematical structure of the model, the criterion, and the techniques for finding the optimal fit to the criterion define the regression method.

In this project, we have looked at one particular model class, that is linear models. We have hence tried to implement three of the most common linear regression techniques: ordinary least squares (OLS), ridge and lasso regression. To focus our work, we have chosen a 2-dimensional polynomial as our model, and used data — simulations and real life measurements — that can effectively be modelled in this manner. We have compared the three methods, looking in particular at their bias-variance trade-offs, and we discuss their advantages and disadvantages in what follows.

Following this introduction, we introduce the theoretical background behind each regression technique. We then describe the methods that we have used to analyse the techniques, in particular the resampling procedures. We then present our results in parallel with discussions and interpretations, before we conclude.

II. LINEAR REGRESSION

Regression refers to a statistical method that is used to estimate the relationship between a set of *independent variables* and an *independent variable*. We will also refer to the independent variables as *features* or *predictors*. The independent variable is also commonly referred to as the *response*. It is implicitly assumed that the values of the independent variables have some effect on the dependent variable, meaning that the statistical properties of the features can be used to predict those of the response.Regression is often used in forecasting or to find the cause-and-effect relationship between two (or more) phenomena.

One widely used type of regression is *linear regression*, in which the "true" relationship between the variables is assumed to be linear. In other words, the response is expected to be a linear combination of the features, if one corrects for the statistical noise. Let the true (or measured) response be represented by a vector $\boldsymbol{z} \in \mathbb{R}^n$, where n corresponds to the number of times that the response is measured (i.e. the number of data points). The linear relationship can be expressed as

$$\boldsymbol{z} = \boldsymbol{z'} + \boldsymbol{\epsilon} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{1}$$

where

$$\boldsymbol{X} = \begin{bmatrix} X_{0,0} & X_{0,1} & \cdots & X_{0,p-1} \\ X_{1,0} & X_{1,1} & \cdots & X_{1,p-1} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n-1,0} & X_{n-1,1} & \cdots & X_{n-1,p-1} \end{bmatrix} \tag{2}$$

is the *design matrix*. Each component $\boldsymbol{X}_{i,j}$ of this matrix represents the value of feature j that has been measured as part of data point i. The response hence depends on the features, as well as on a vector of parameters $\boldsymbol{\beta} \in \mathbb{R}^p$, where p is the number of features. Incidentally, each parameter can thus be associated with one of the features. The systematic component of the response $\boldsymbol{z'} = \boldsymbol{\beta}\boldsymbol{X} \in \mathbb{R}^n$ represents the value that the independent variable would take if not for the stochastic component $\boldsymbol{\epsilon} \in \mathbb{R}^n$. This stochastic component represents the statistical noise that can result from random sources of error, in particular uncertainties in measurements. In this investigation, we will assume that each element of $\boldsymbol{\epsilon}$ follows a normal distribution, $\epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right)$. This is in general a good guess, given no further information, because random sources of error can be viewed as a sum of random variables, and such a sum tends to approach a Gaussian distribution, by the central limit theorem [1].

The aim in linear regression is to find the *regression parameters* $\hat{\boldsymbol{\beta}}$, which represent the approximation of $\boldsymbol{\beta}$ that

best fit the data. Mathematically, if we define the *predicted response* $\tilde{\boldsymbol{z}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}$, we aim to determine the values of $\hat{\boldsymbol{\beta}}$ such that the *residuals* $\tilde{\boldsymbol{\epsilon}} = \boldsymbol{z} - \tilde{\boldsymbol{z}}$ between the true response and our approximation are minimal. To make this concept more concrete, we define a cost function $C(\boldsymbol{\beta})$, which works as a summary statistic describing the error between the true response and the prediction.

A. Ordinary least squares

The cost function can be defined in different ways, each corresponding to a different linear regression method. The most common choice is the least squares method,

$$C(\boldsymbol{\beta}) = \frac{1}{n}(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta}) = \frac{1}{n}\tilde{\boldsymbol{\epsilon}}^T\tilde{\boldsymbol{\epsilon}} \tag{3}$$

To find the optimal estimate of the regression parameters $\hat{\boldsymbol{\beta}}$, we have to minimise the cost function with respect to $\boldsymbol{\beta}$. The first condition of a minimum is that the gradient with respect to $\boldsymbol{\beta}$ should be zero, $\nabla_{\boldsymbol{\beta}}\left(\tilde{\boldsymbol{\epsilon}}^T\tilde{\boldsymbol{\epsilon}}\right) = 0$, where

$$\begin{aligned}\tilde{\boldsymbol{\epsilon}}^T\tilde{\boldsymbol{\epsilon}} &= (\boldsymbol{z} - \tilde{\boldsymbol{z}})^T(\boldsymbol{z} - \tilde{\boldsymbol{z}}) = (\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta}) \\ &= \boldsymbol{z}^T\boldsymbol{z} - \boldsymbol{z}^T\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{z} + \boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} \\ &= \boldsymbol{z}^T\boldsymbol{z} - 2\boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{z} + \boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}\end{aligned}$$

here we have used that the term $\boldsymbol{z}^T\boldsymbol{X}\boldsymbol{\beta}$ is a scalar, so it is equal to its transpose, which means that the two cross-terms are equal. Differentiating with respect to $\boldsymbol{\beta}$ we get,

$$\frac{\partial\tilde{\boldsymbol{\epsilon}}^T\tilde{\boldsymbol{\epsilon}}}{\partial\boldsymbol{\beta}} = -2\boldsymbol{X}^T\boldsymbol{z} + 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}. \tag{4}$$

Setting this to zero we find the optimal estimate for the regression parameters,

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{z}, \tag{5}$$

provided that the matrix $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ is invertible and that the number of features does not exceed the number of measurements. The variance of the regression parameters can be found from equations (3.8) in Hastie et al. [1],

$$\text{var}[\hat{\boldsymbol{\beta}}] = \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1} \tag{6}$$

To be clear, this is the variance expected from making multiple "measurements" (or regressions) of $\hat{\boldsymbol{\beta}}$ with the same values $\boldsymbol{X}$ of the independent variables. The variation occurs due to the statistical noise $\epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right)$ (explained previously), as is reflected in the occurrence of σ in (6). The full derivation can be found in the appendix. We can estimate the variance σ^2 with,

$$\hat{\sigma}^2 = \frac{1}{n-p-1}\sum_{i=0}^{n-1}(z_i - \tilde{z}_i) \tag{7}$$

using $n - p - 1$ rather than n in the denominator makes this an unbiased estimate of σ^2 [1].

1. Problems with OLS-regression

The Gauss-Markov theorem states that the unbiased estimates of $\boldsymbol{\beta}$ produced using the OLS method have the smallest variance of all possible linear estimators.[1] They are simply the best, as long as the model complies with the assumption that none of the independent variables are a perfect[1] linear function of other explanatory variables. The OLS-method cannot distinguish between perfectly correlated variables, but it can still be used if there is a strong (not perfect) relationship between the independent variables. However, if the correlations are significant enough they can cause some big problems that reduces the precision of the OLS estimates. This situation, where an independent variable is a linear function of other variables, is referred to as multicollinearity, or super-collinearity.

Multicollinearity exists in two types; the first one is the one that is likely to be present in the data from observational experiments, while the other is the structural multicollinearity which occurs as a result of the chosen model. An example of this is when we want to model a curve using a polynomial, there would be a correlation between x, x^2 etc., so the columns in our design matrix $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ would be linearly-dependent. As a consequence, the square matrix $\boldsymbol{X}^T\boldsymbol{X} \in \mathbb{R}^{p\times p}$ is singular and non-invertible, which causes problems in our expression for the OLS-estimates in (5). There is also the problem of $\boldsymbol{X}$ being of high-dimension ($p > n$)[4] previously mentioned, when we have a lot of parameters included in our model, where the OLS-estimates are not well defined[2].

To deal with these problems we can instead try a different estimator that is not unbiased. Biased estimators are commonly used, and we are going to test two methods that works around the problem by shrinking the least squares coefficients, which results in a biased estimate.

B. Ridge and Lasso Regression

One of the work-around's is to introduce the regression estimator called the Ridge Regression estimator[3]. It uses an ad-hoc fix to resolve the issue with the singularity by making the replacement,

$$\boldsymbol{X}^T\boldsymbol{X} \quad \rightarrow \quad \boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}_{p\times p} \tag{8}$$

Where the scalar $\lambda > 0$ is called a penalty parameter because it shrinks the regression parameters by imposing

[1] Perfect correlation is when two variables have a Pearson's correlation coefficient of +1 or -1

a penalty on their size. By increasing λ, more and more $\boldsymbol{\beta}$ values can be removed. When there is a large amount of noise in the sample, the variance is considerably large, hence the need to shrink the variance. With this replacement we get a different cost function, and thus another expression for the best estimated regression parameters,

$$C(\boldsymbol{\beta}(\boldsymbol{\lambda})) = \frac{1}{n}(\boldsymbol{z}-\boldsymbol{X}\boldsymbol{\beta})(\boldsymbol{z}-\boldsymbol{X}\boldsymbol{\beta})^T + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta} \quad (9)$$

$$\hat{\boldsymbol{\beta}}(\lambda) = \underbrace{(\boldsymbol{X}^T\boldsymbol{X}+\lambda\boldsymbol{I})^{-1}}_{\mathbb{R}^{p\times p}}\boldsymbol{X}^T\boldsymbol{z} \quad (10)$$

The variance of $\hat{\boldsymbol{\beta}}(\lambda)$ is derived in [4], and just repeated here,

$$\text{var}[\hat{\boldsymbol{\beta}}(\lambda)] = \sigma^2(\boldsymbol{X}^T\boldsymbol{X}-\lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{X}[(\boldsymbol{X}^T\boldsymbol{X}+\lambda\boldsymbol{I})^{-1}]^T \quad (11)$$

Ridge-regression is stable for a wide range of values for λ, but as the values increase it becomes more ineffective. Another method is the Least Absolute Shrinkage and Selection (LASSO) operator. For LASSO we have yet another cost function,

$$C(\boldsymbol{X},\boldsymbol{\beta};\lambda) = (\boldsymbol{X}\boldsymbol{\beta}-\boldsymbol{z})^T(\boldsymbol{X}\boldsymbol{\beta}-\boldsymbol{z}) + \lambda\sqrt{\boldsymbol{\beta}^T\boldsymbol{\beta}}. \quad (12)$$

Lasso have no analytical solution for the optimal parameter $\hat{\boldsymbol{\beta}}$ as it's cost function involves the absolute value (more in Appendix), we need to minimize it using a numerical method, the gradient descent method. We will not do this ourselves, but we will use the Lasso function provided by scikit learn. The LASSO method is suitable for low values of λ, but will diminish when we increase λ towards higher values.

C. The Bias-Variance Tradeoff

The Bias-Variance tradeoff is a property of all machine learning models, that imposes a tradeoff between flexibility of the model and it's performance on unseen data. With increasing complexity, the model variance tends to increase and squared bias tends to decrease. This implies that a model should maintain a balance between underfitting and overfitting. When training is exactly fitted to the data such that it learns from the noise and inaccurate data entries, it is considered overfitted (High variance and low bias). While on the contrary, when we have less amount of data and modelling is done with less complexity, such that algorithm doesn't fit the data enough to capture the underlying trend of the data, it is underfitting (High bias and low variance). While performing Bias-variance Tradeoff, new training is performed keeping the test data fixed i.e. calculation for Mean square error , variance , and bias is done from the same test

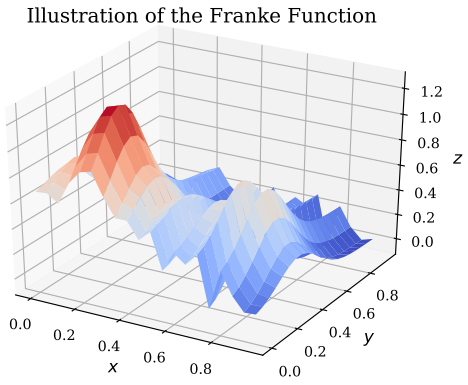

FIG. 1: An illustration of the Franke Function that is used to generate the test dataset, here with added noise $\mathcal{N}(0, 0.1)$

date. Cost function in terms of $\boldsymbol{\beta}$, which is obtained by optimizing the MSE is given by,

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

Which can be rewritten as

$$\mathbb{E}\left[(\boldsymbol{z}-\tilde{\boldsymbol{z}})^2\right] = \mathbb{E}\left[(\boldsymbol{z}-\mathbb{E}\left[\tilde{\boldsymbol{z}}\right])^2\right] + \text{Var}\left[\tilde{\boldsymbol{z}}\right] + \boldsymbol{\sigma}^2, \quad (13)$$

(for Full derivation see Appendix**??**)
Where Ist term is bias, deviation from the mean value of the model with the actual data(A model with high bias does not capture the underlying behaviour of true functional form); IInd term is the variance of the model. It determines how much the average model estimate deviates from actual training data. Overfitted training data results in model with high variance ; and last term is variance of the error ,$\boldsymbol{\epsilon}$, which is irreducible, it is the minimum lower bound for MSE.
To have improved prediction this tradeoff needs to be minimised by shrinking or setting some coefficient to zero where little bias is sacrificed to reduce variance . This is achieved with improved strategy of best subset selection as been some in Ridge and Lasso.

III. METHODS

A. The Franke Function

To evaluate and test the different regression methods we are using a dataset $z = f(x,y) + \mathcal{N}(0,\sigma^{\epsilon)}$ where f(x,y) is the Franke function (14) with $x, y \in [0,1]$, and $\mathcal{N}(0,\sigma^{\epsilon)}$ is normally distributed noise, centered at zero and with variance σ^{ϵ} that we add to the datapoints. An illustraion of this function can be seen in figure 1.

$$
\begin{aligned}
f(x,y) = &\frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) \\
&+ \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) \\
&+ \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) \\
&- \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right) \qquad (14)
\end{aligned}
$$

B. Polynomial Design Matrix

To model the data we use a polynomial with degree d, each column of the design matrix (2) will then be filled with a monomial x^α of degree $\alpha \leq d$ (for more about this see Appendix B). For instance, if we choose to use a model with polynomial degree 2, the design matrix will be on the form,

$$
\boldsymbol{X} = \begin{bmatrix}
1 & x_0 & y_0 & x_0^2 & x_0y_0 & y_0^2 \\
1 & x_1 & y_1 & x_1^2 & x_1y_1 & y_1^2 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
1 & x_{n-1} & y_{n-1} & x_{n-1}^2 & x_{n-1}y_{n-1} & y_{n-1}^2
\end{bmatrix} \qquad (15)
$$

C. Preprocessing

1. Scaling

Scaling of data let us get rid of outliers in the data, if there are any, i.e. it normalises the data within a particular range.It make sure one significant value in the data doesn't impact the model just because of their large magnitude. It sometimes speeds up the calculations in algorithms as well.

2. Splitting

Splitting the data enables us to evaluate our model general features and give an insight of how it would perform on unseen data.In this project we are splitting the data into train and test only as we don't have very large data set to be analysed so validation set is omitted. Usually split proportion is 3:1. 80 percent training data and 20 percent test data. Train-test split is a practice that is followed in the model building. The training dataset is the initial data-set used to train an algorithm to fit the model.Testing data allows us to test our model on data that is independent of our training data which helps us find whether the model is over-fitting or underfitting. If our model is actually a good model therefore, it should perform just as well on our testing data as training data.Splitting the data into train , validation and test make sure there is no over-fitting.If our model does much better on the training set than on the test set, then we're likely over-fitting. It is done with standard functionality in ***Scikitlearn***

D. Resampling methods

Resampling is statistical approach which counts on factual analysis, depending upon observed events instead of asymptotic and parametric theory. Main ideology behind this method is to repeatedly draw samples from training data set and performing fit in the sample so as to draw information from the model.

1. Bootstrap

Its a kind of re-sampling technique first introduced by Bradely Efron in regression. While doing polynomial fit by using bootstrap the re-sampling is with replacement so we can get same points back again, which means If we have data $=z_1, z_2, z_3, z_4, z_5$ After re-sampling we may get, data* $=z_1, z_1, z_1, z_2, z_3$. In this method train data is being re-sampled and we perform a new fit on every test data. For that specific re-sampling we thus make a prediction , this specific prediction we make it with that specific training data which has been re-sampled and this thus goes through all the number of bootstrap we have . This process is repeated a large number of times , and for each of these bootstrap samples we compute its mean ,bias ,variance and MSE. As during this process test remain same all the time it is easier to perform bias variance trade off.
Since it does not require distributional assumptions (such as normally distributed errors), the bootstrap can provide more accurate inferences when the data are not well behaved or when the sample size is small.

2. Cross-Validation

There are two types of Cross Validation, One is k-fold CV, another one is LOOCV (Leave one out Cross validation)
I) In this type of cross validation data is split into k mutually exclusive folds and out of these k- folds k-1 folds corresponds to training set and 1 fold as test set . Then model is fitted on the training set and error is estimated on the test set. This procedure is repeated k-times with each repeating holding out k-1 fold for training and one for test.MSE is calculated for each fold and overall MSE is the average of all the individual MSE (there are k MSE's). Also K is chosen not so big so as to be computationally economic and to minimize (optimise) Bias Variance Tradeoff .
II) Simplest way of LOOCV is to take a dataset , leaving

one out of the dataset and performing fit on the remaining data. Then similar to the linear regression performing prediction on the left out datapoint. If we have n datapoints , then we fit the model n times, with only leaving out a single observation for each fitting.
This method reduces biass to a great extent at the expense of introducing high variance. it is computationally expensive then K fold CV as it involves fitting the model n times (if n is large) also result in bias variance tradeoff.

IV. RESULTS AND DISCUSSION

We first look at some results from our implementation of the OLS regression. We present the resulting regression parameters for one instance of this regression method for a polynomial of degree 5 in figure 2. At a first glance, these results seem quite promising, as the MSE seems quite low (although we have nothing to compare this value with yet), and the R^2-score is quite close to one, which is an indication of a good fit. It is also interesting to note that most of the parameter values' confidence intervals overlap with those of their nearest neighbours, and many of the central values lie within the values -10 to 10. This is intuitively a sign of a good regression, as the parameter values don't vary wildly. The parameter $\hat{\beta}_6$ stands out because of it's high value (more than 20), but also because its confidence interval far from overlaps with that of $\hat{\beta}_5$. This is however consistent with the image of the Franke Function in figure 14, as this has a maximum that stands out quite distinctly from the rest of the function. It makes intuitive sense that some degree of the fitted polynomial has to be quite dominant to match such a distinct feature. I is also notable that many of the confidence intervals are quite large compared to the absolute values of the parameters. This indicates that the stochastic noise in our data simulation is quite significant, which in turn meets that the predictions that we make based on our fits may be more uncertain. Intuitively, and based on equation (6), this could be improved by using more data for our fits.

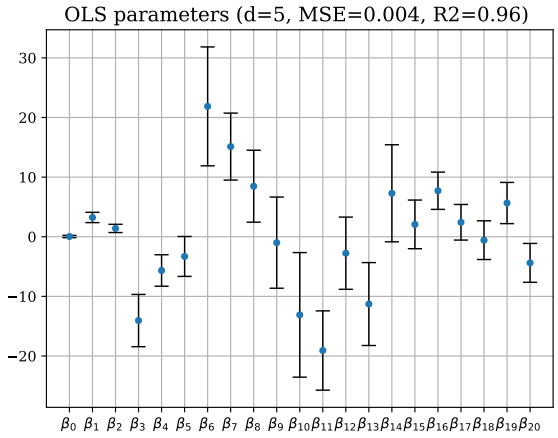

FIG. 2: Regression parameters for an OLS regression of simulated data. Error bars represent the confidence intervals of the parameter values. 100 data points were used for this regression.

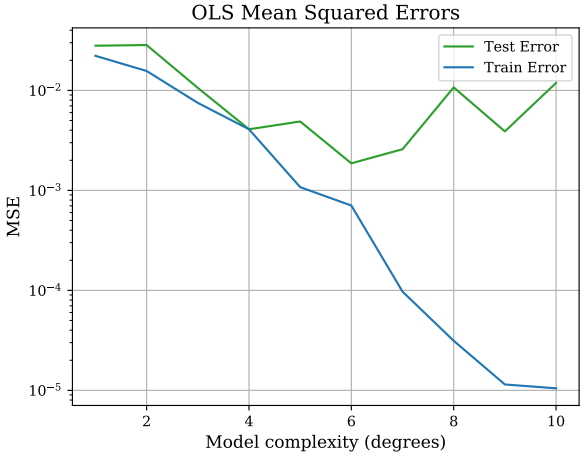

FIG. 3: MSE found for OLS regression of simulated data. 100 data points were used for this regression.

In figure 3 we plot the MSE of the OLS regression for both the training and the test data, and as a function of the polynomial degree (in a similar fashion as in figure 2.11 of [1]). As expected, the MSE falls rapidly for higher degrees in the case of the training data. For the test data on the other hand, the MSE is quite close to that of the training data up to intermediate degrees (here about 4-5), but then rises again for higher ones, indicating overfitting. From figure 3 it may seem as though the fit is best for a 6-degree polynomial, however, it should also be noted that the curve corresponding to the test data is very rugged, which makes it somewhat ambiguous which number of degrees in the range 5-7 theoretically gives the best result. This ruggedness is to be expected when calculating the MSEs based on a single data set, in particular considering the large variance in the regression parameters discussed above.

In figure 4 we in essence reproduce figure 3, except that here we have resampled the dataset using the bootstrap algorithm described in the methods section. It is immediately clear that the curve that corresponds to the MSE of the testing data is much smoother for the resampled case. This makes sense given that we in that case in effect take the mean value of the MSE over several regressions. It is however surprising that the minimum MSE in this case seems to occur for a 3-degree polynomial, rather than a 5-7-degree one.

To analyse the regression in more detail, we plot the MSE together with its bias and variance components in figure **??**. As expected, the bias dominates the MSE

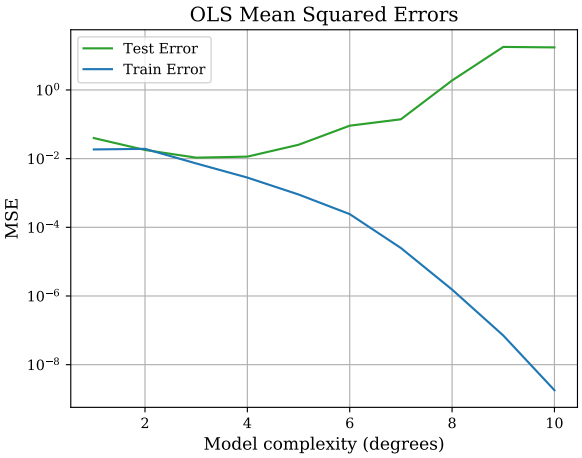

FIG. 4: MSE found for OLS regression of simulated data, with resampling using the bootstrap method. 100 data points and 100 resamplings were used for this regression.

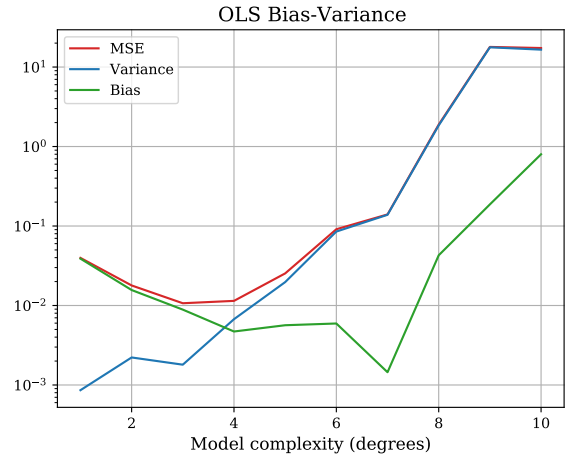

FIG. 5: MSE, bias and variance found for OLS regression of simulated data, with resampling using the bootstrap method. 100 data points and 100 resamplings were used for this regression.

for low degrees, while the variance dominates for lower degrees. What is unexpected though, is that the bias increases for degrees higher than 7. The bias is however expected to decrease with increased model complexity. This indicates that there may be something wrong with our code. We confirm this suspicion by looking at how the R^2-score varies with increasing complexity in figure 6. Evidently, the R^2 score plunges to highly negative values after the 7-degree mark, which should not happen under normal circumstances (loosly speaking this would imply that the deviation of the predicted responses from the "true" responses on average was higher than the actual range of the "true" responses — that would mean that the fitted model would not remotely reproduce any underlying trend in the real feature-response relationship). On the somewhat brighter side, the R^2-score calculated with the training data in place of the test data approaches 1, as one would expect. This indicates that the training procedure indeed matches predicted values with the given data to some extent. The error in our code is thus probably related to how we use the trained model after fitting.

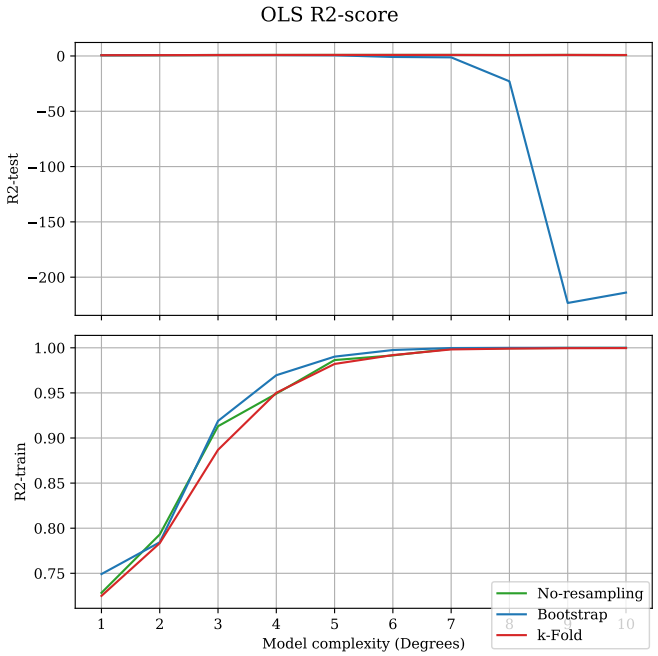

FIG. 6: R^2-score for OLS regression, using 100 bootstraps, 5 k-folds or no resampling. 100 data points were used for the regression.

We also look at how the bias-variance trade-off is affected by the number of data points that we use train our algorithm. We reproduce the plots on the Bias-variance trade-off, but for a higher number of data points in figure 7. As we expect from the lectures, the point where the MSE starts to increase is shifted to a higher degree of the fitted polynomial. This makes sense, because it is the variance that ultimately makes the MSE increase for higher degrees. The variance becomes large for higher degrees because the model has more degrees of freedom to overcompensate for statistical fluctuations. When the number of datapoints is high though, the fluctuations become more numerous, and the fitted model has to average over them to a greater extent, thus making for a smaller variance in the predicted values. Hence, larger datasets imply lower variances, and thus a lower MSE for a given degree.

Compared to the variance, the bias does not seem to change much from figure 5 to figure 7. This makes sense, as this values is only dependent on the average of the model's predicted values, which shouldn't change much with a higher data set. It is interesting to note that the

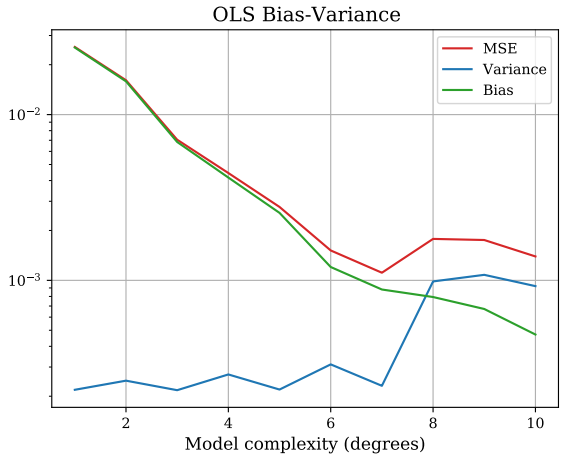

FIG. 7: MSE, bias and variance found for OLS regression of simulated data, with resampling using the bootstrap method. 500 data points and 100 resamplings were used for this regression.

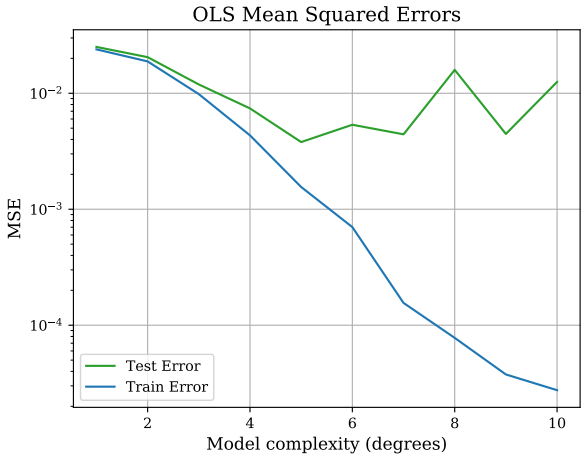

FIG. 8: MSE found for OLS regression of simulated data, with resampling using the cross validation method. 100 data points and 5 k-folds were used for this regression.

coding error that we discussed above does not seem to have a significant effect below a polynomial degree of 10 for a data set of size 500.

Figure 8 re-visits the Bias-Variance trade-off, but this time analysed using the cross-validation resampling technique. To our relief, this figure resembles figure 3 to a much better extent than figure 4, indicating that we don't have the same coding error here as we had for the bootstrap technique. However the curve for the testing data is still somewhat rugged, which may be explained by the fact that this figure corresponds to only 5 k-folds. In other words, there aren't very many measurements of the MSE to average over. Therefore, we make the same plot again in figure **??**, where the number of k-folds is now 10. Clearly, the Test Error curve is now smoother, and it is easier to determine the optimal number of degrees. This is apparently 8, which is somewhat unexpected, but plausible based on the original visualisation in figure 3.

We now look at how the introduction of a shrinkage coefficient affects the Bias-Variance trade-off. This is best illustrated by figure 10, which illustrates the variation of the bias and variance as a function of the shrinkage coefficient λ. Seing that $\lambda = 0$ represents the OLS method, it is clear that a small addition of lambda reduces the variance — and thus the MSE — significantly. This is expected as the shrinkage effectively reduces the number of features, thus reducing overfitting. At higher values of λ it turns out that the variance as well as the bias increase consistently. This makes sense, as the shrinkage coefficient also introduces a bias in the best fit. This means that an increasing λ will consistently increase or decrease the mean of the predicted values, on which both the variance and the

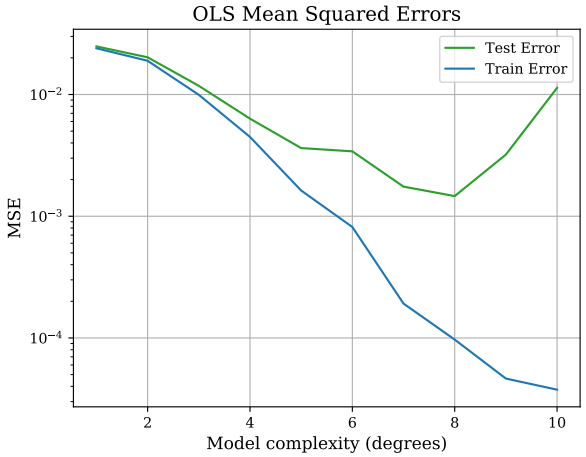

FIG. 9: MSE found for OLS regression of simulated data, with resampling using the cross validation method. 100 data points and 10 k-folds were used for this regression.

bias depend.

It is notable that the number of features in the regression model affects how the shrinkage coefficient affects the variance. When the number of degrees is low, there is a low potential for overfitting, hence the shrinkage coefficient doesn't have much effect at all — this can be seen in figure 11 where the degree is 2 (in figure 10 it is 6). When the number of degrees is higher, the potential for overfitting is higher, and thus the optimal variance occurs at a higher λ; however this variance may also be higher than for a lower degree, as the resulting bias in the regression will be stronger — this can be seen in figure 12 where the number of degrees is 9. Hence, finding the optimal shrinkage coefficient is complex, and must be done by looping over both λ and the number of

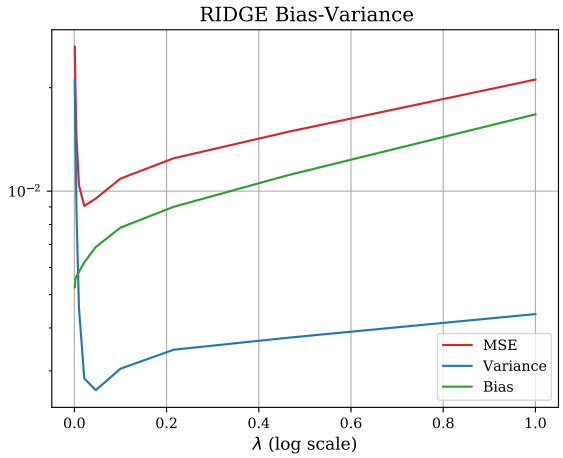

FIG. 10: MSE, bias and variance found for Ridge regression of simulated data, with resampling using the bootstrap method. 100 data points and 100 resamplings and 6 polynomial degrees were used for this regression.

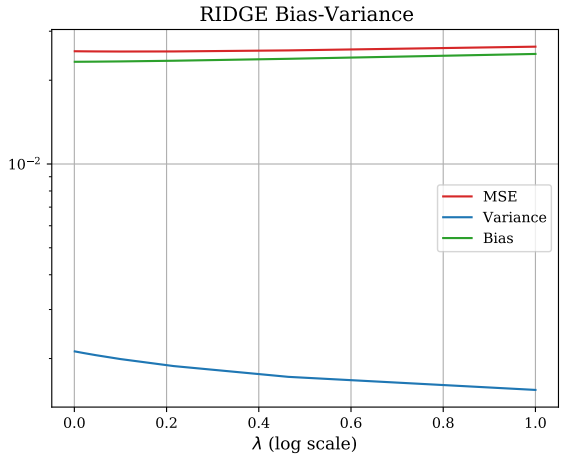

FIG. 11: MSE, bias and variance found for Ridge regression of simulated data, with resampling using the bootstrap method. 100 data points and 100 resamplings and 2 polynomial degrees were used for this regression.

degrees, and finding the minimum possible MSE.

The same kind of considerations should be possible for Lasso regression, as this shrinks (or effectively cancels) some parameters analogous to Ridge regression. However, due to many coding problems, we cannot present the appropriate graphs here.

A. Real data

As the real data we are using one of the terrain files that can be found in the github repository of this Ma-

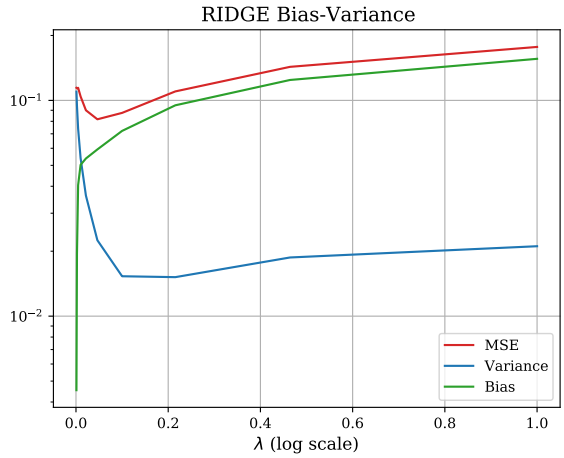

FIG. 12: MSE, bias and variance found for Ridge regression of simulated data, with resampling using the bootstrap method. 100 data points and 100 resamplings and 9 polynomial degrees were used for this regression.

TABLE I: The model chosen by the analysis as the best fit to the real terrain data

method	Degree	MSE	R^2
None	4	1.50	0.995
Bootstrap	4	2.454	0.986
kFold	4	4.404	0.981

chine Learning course (Fig 13). Using the OLS-regression method without resampling, with bootstrap resampling and with 5-fold cross-validation yields the same model complexity as the best fit, but as can be read from table I the MSE values for these models are quite high. The Lasso and ridge regression methods both yield the best

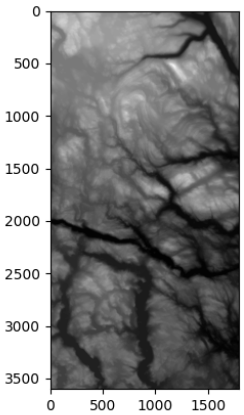

FIG. 13: Graphical illustration of the terrain file that is used. It shows an area of Norway, either near Stanvanger or near Møsvatn Austfjell.

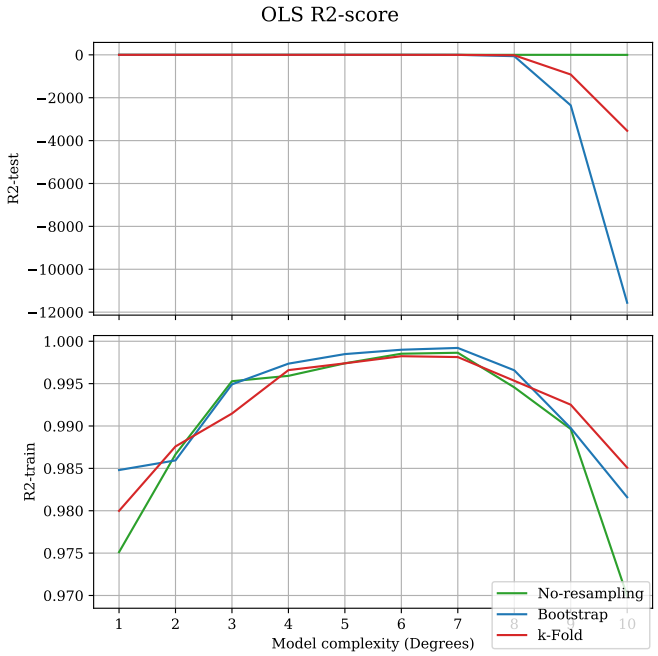

FIG. 14: Comparison of the explained R^2 score for the OLS method when using bootstrap, cross-validation and no-resampling. For the cross validation we have used $k = 5$ folds and for bootstrap we are using 100.

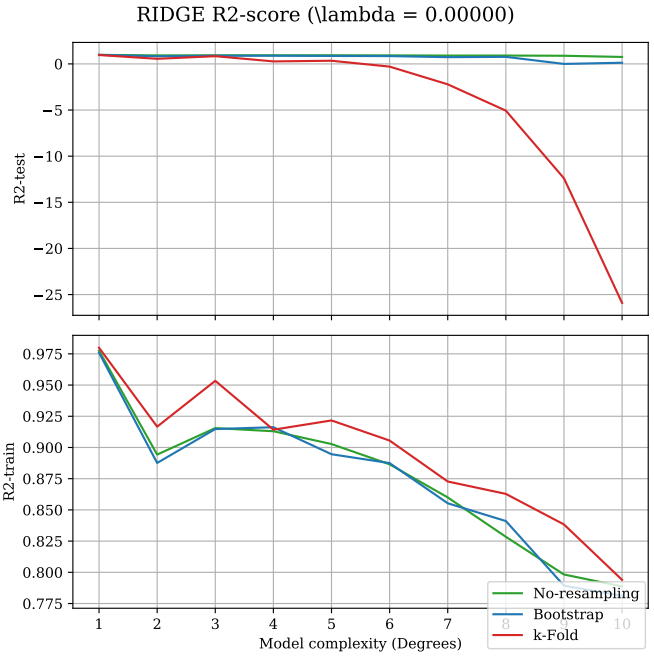

FIG. 15: Comparison of the explained R^2 score for the Ridge method when using bootstrap, cross-validation and no-resampling. For the cross validation we have used $k = 5$ folds and for bootstrap we are using 100. The λ value used is 0.001 (there is an error in the title of the figure)

fitted models for low polynomial degree and with penalty parameter $\lambda = 0.001$, with MSE ~ 5. Figure 14 shows the R2-score as a function of model complexity for the OLS method, and as we can see, the R2-score for the training does not approach 1 as complexity increases as it did when analysing the Franke function. One issue may be that the OLS function implemented in the file *functions.py* uses the numpy pseudoinverse function to invert the expression in (5). As can be seen from figures 15 and 16, the R2-training-scores for these methods decrease as the model complexity increases, which means that something numerical may be wrong with our code. Figure 17 shows the bias-variance tradeoff for ridge regression with zero penalty parameter, which is the same as we get for OLS regression.

V. CONCLUSION

In this project we have explored the various features of OLS, ridge and lasso regression, as well as of bootstrap and cross validation resampling. We have demonstrated that the optimal set-up for a given regression method can be effectively described in terms of the bias-variance trade-off. The latter is also highly consequential for determining which regression method is the most effective for a given model or data set. In particular, it turns out that while the OLS regression can effectively produce unbiased estimations of a given variable, it is quite vulnerable to over-fitting when the amount of data that

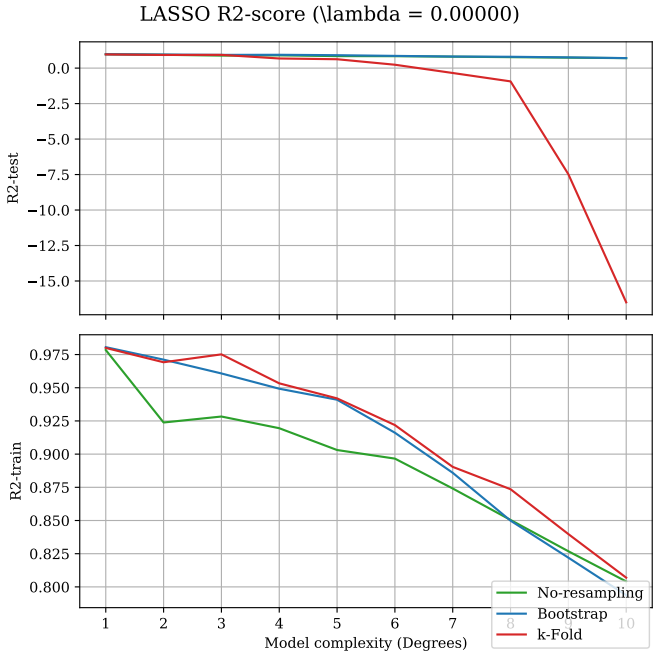

FIG. 16: Comparison of the explained R^2 score for the Lasso method when using bootstrap, cross-validation and no-resampling. For the cross validation we have used $k = 5$ folds and for bootstrap we are using 100.The λ value used is 0.001 (there is an error in the title of the figure)

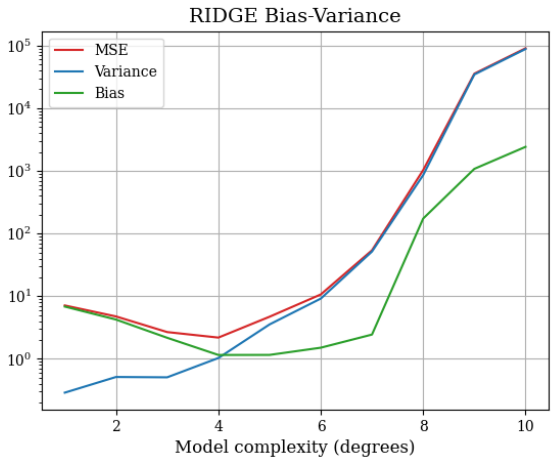

FIG. 17: The bias-variance trade off for terrain data when using ridge regression with $\lambda = 0$, 100 bootstraps and 100 datapoints.

is given is low. The shrinkage methods ridge regression and lasso regression are then good alternatives, as they effectively reduce overfitting, and thus can produce estimates that are closer to the "truth" than OLS, despite their inherent bias.

From the analysis on real data it would seem that the best method is OLS with degree 4, but when looking at the explained R2-scores for both test and training we can see that none of the models are preforming well. We expect the R2-score for the training set to approach 1 as the model complexity increases, but it does not. When using Lasso and Ridge regression, the next step will be to investigate if using a SVD would improve the results, and to test the methods using *scikitlearn*'s regression methods.

[1] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction; 2nd ed.* Springer Series in Statistics. Springer, Dordrecht, 2009. doi: 10.1007/978-0-387-84858-7. URL https://cds.cern.ch/record/1315326.

[2] M. Hjorth-Jensen. Data analysis and machine learning: Linear regression and more advanced regression analysis. (11):1–59, 09 2020. URL https://github.com/CompPhysics/MachineLearning/blob/master/doc/pub/Regression/pdf/Regression-minted.pdf.

[3] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. doi: 10.1080/00401706.1970.10488634. URL https://www.tandfonline.com/doi/abs/10.1080/00401706.1970.10488634.

[4] W. N. van Wieringen. Lecture notes on ridge regression, 2020.

Appendix A: Variance of the OLS regression parameters

The using the equation,

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{z} \tag{A1}$$

We can find that the expectation value of $\hat{\boldsymbol{\beta}}$ is simply $\boldsymbol{\beta}$,

$$\mathbb{E}[\hat{\boldsymbol{\beta}}] = \mathbb{E}[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{z}] = \boldsymbol{\beta}$$

The variance is then,

$$\begin{aligned}\text{var}[\hat{\boldsymbol{\beta}}] &= \mathbb{E}[(\hat{\boldsymbol{\beta}} - \mathbb{E}[\hat{\boldsymbol{\beta}}])(\hat{\boldsymbol{\beta}} - \mathbb{E}[\hat{\boldsymbol{\beta}}])^T] \\ &= \mathbb{E}[(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^T] \\ &= \mathbb{E}[\hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T] - \mathbb{E}[\hat{\boldsymbol{\beta}}\boldsymbol{\beta}^T] - \mathbb{E}[\boldsymbol{\beta}\hat{\boldsymbol{\beta}}^T] + \mathbb{E}[\boldsymbol{\beta}\boldsymbol{\beta}^T] \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}[\boldsymbol{z}\boldsymbol{z}^T]\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T\end{aligned} \tag{A2}$$

Here we have used that $\boldsymbol{X}$ and $\boldsymbol{\beta}$ are known. The expectation value for $\boldsymbol{z}\boldsymbol{z}^T$ can be found using the expression for our model in (1),

$$\begin{aligned}\mathbb{E}[\boldsymbol{z}\boldsymbol{z}^T] &= \mathbb{E}[(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon})(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon})^T] \\ &= \mathbb{E}[\boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T + \boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\epsilon}^T + \boldsymbol{\epsilon}\boldsymbol{\beta}^T\boldsymbol{X}^T + \boldsymbol{\epsilon}\boldsymbol{\epsilon}^T] \\ &= \boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T + \boldsymbol{X}\boldsymbol{\beta}\mathbb{E}[\boldsymbol{\epsilon}^T] + \mathbb{E}[\boldsymbol{\epsilon}]\boldsymbol{\beta}^T\boldsymbol{X}^T + \mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T] \\ &= \boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T + \sigma^2\end{aligned}$$

Remember that $\boldsymbol{\epsilon}$ is normally distributed noise $\mathcal{N}(0, \sigma^2)$, so the expectation value of $\boldsymbol{\epsilon}$ is zero, and $\mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T] = \text{var}[\boldsymbol{\epsilon}] = \sigma^2$. Inserting the expectation value of $\boldsymbol{z}\boldsymbol{z}^T$ into the expression for variance we get,

$$\begin{aligned}\text{var}[\hat{\boldsymbol{\beta}}] &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T + \sigma^2)\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \\ &= \boldsymbol{\beta}\boldsymbol{\beta}^T + \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \\ &= \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}\end{aligned} \tag{A3}$$

Appendix B: Polynomial Design Matrix

We can write any polynomial in k variables $x_1, x_2, ..., x_k$ as a linear-combination of the monomials $x^\alpha = x_1^{\alpha_1}x_2^{\alpha_2} \cdot \cdots x_k^{\alpha_k}$, where the exponents are non-negative integers $\alpha = (\alpha_1, ..., \alpha_k) \in \mathbb{N}^k$. The monomials x^α will form a basis for the polynomial space $\mathcal{P}$, so the polynomials of at most degree d will form a subspace which has the monomials $(x^\alpha)_{|\alpha|\leq d}$ as a basis and the dimension,

$$p = \binom{d+k}{d} = \binom{d+k}{k} = \frac{(d+1)\cdots(d+k)}{k!} \tag{B1}$$

And these monomials will make up the rows in the design matrix in (2). We can write the polynomial $g_d(x)$ of degree d using the multinomial theorem,

$$g_d(x) = \sum_{|\alpha|=d}\binom{d}{\alpha}x^\alpha = \sum_{|\alpha|=d}\binom{d}{\alpha_1, ..., \alpha_k}\prod_{t=1}^{k}x_t^{\alpha_t} \tag{B2}$$

where $|\alpha| = \alpha_1 + \alpha_2 + \cdots + \alpha_k$ is the total degree of the polynomial. The polynomial which will include all terms that have degree at least d can be written as,

$$g(x) = \sum_{i=0}^{d}g_i(x) = 1 + \sum_{i=1}^{d}\left[\sum_{|\alpha|=i}\binom{i}{\alpha}x^\alpha\right] \tag{B3}$$

For $k = 2$ variables we can use $\alpha_1 + \alpha_2 = d$ to write α_1 in terms of α_2 and the multinomial theorem reduces to the binomial theorem. The polynomial $g(x_1, x_2)$ with terms that have degree at least d is given by,

$$g(x) = 1 + \sum_{i=1}^{d}\left[\sum_{\alpha_1+\alpha_2=i}\binom{i}{\alpha_1, \alpha_2}x_1^{i-\alpha_2}x_2^{\alpha_2}\right] \tag{B4}$$

So for a given degree d the the subspace vector space $\mathbb{R}[x]_d$ has the basis $(x^\alpha)_{|\alpha|\leq d}$ and the columns of the design matrix will be,

$$(1, x_1, ..., x_k, x_1^2, x_1x_2, ..., x_1x_k, x_2^2, ..., x_k^2, ..., x_1^d, ..., x_k^d)^T$$

Appendix C: Singular value decomposition

The SVD theorem states that matrix, $\boldsymbol{X}$ of dimension $m \times n$ can be written in terms of diagonal matrix of dimensionality $m \times n$ and two orthogonal matrices $\boldsymbol{U}$ $(m \times m)$ and $\boldsymbol{V}$ $(n \times n)$.[2]

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}} \tag{C1}$$

We are using SVD, as it decomposes a matrix even if it is singular and non-invertible. The SVD gives eigenvalues $\sigma > \sigma_i + 1$ for all i, eigenvalues are zero for i=p.
OLS solutions in terms of the eigenvectors (the columns) of the right singular value matrix $\boldsymbol{U}$ as

$$\begin{aligned}\boldsymbol{X}\boldsymbol{\beta} =& \boldsymbol{X}\left(\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T\right)^{-1}\boldsymbol{X}^T\boldsymbol{z} \\ &= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}}\left(\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T\right)^{-1}(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}})^T\boldsymbol{y} = \boldsymbol{U}\boldsymbol{U}^T\boldsymbol{z}\end{aligned}$$

$$\boldsymbol{\beta} = \boldsymbol{V}\boldsymbol{\Sigma}^{+}\boldsymbol{U}^T\boldsymbol{z}. \tag{C2}$$

For Ridge regression

$$\begin{aligned}\boldsymbol{X\beta}^{\text{Ridge}} =&\boldsymbol{U\Sigma V^T}\left(\boldsymbol{VDV}^T+\lambda\boldsymbol{I}\right)^{-1}(\boldsymbol{U\Sigma V^T})^T\boldsymbol{z}\\ =&\sum_{j=0}^{p-1}\boldsymbol{u}_j\boldsymbol{u}_j^T\frac{\sigma_j^2}{\sigma_j^2+\lambda}\boldsymbol{z},\end{aligned}$$

it means that compared to OLS, we have

$$\frac{\sigma_j^2}{\sigma_j^2+\lambda}\leq 1.$$

Ridge regression finds the coordinates of z with respect to the orthonormal basis U , which then shrinks the coordinates by $\frac{\sigma_j^2}{\sigma_j^2+\lambda}$ $\sigma_i \geq \sigma_{i+1}$ For small eigenvalues σ_i their contributions become less important, which leads to decreasing number of degrees of freedom.or we can say A greater amount of shrinkage is applied to the coordinates of basis vector with smaller values of σ_j^2. . With a parameter λ we can thus shrink the role of specific parameters, as is done in Ridge Regression. In Ridge cost function[2] is

$$C(\boldsymbol{X},\boldsymbol{\beta})=\frac{1}{n}\sum_{i=0}^{n-1}(y_i-\tilde{y}_i)^2=\frac{1}{n}||\boldsymbol{y}-\boldsymbol{X\beta}||_2^2, \qquad \text{(C3)}$$

where L2 norm vector is defined as

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

cost function for Lasso regression[4]

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

here L1 norm is defined as

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

Appendix D: Bias Variance Tradeoff

The data from the franke function have normally distributed noise with zero mean and $\boldsymbol{\sigma}^2$ standard deviation.

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

And the cost function in terms of $\boldsymbol{\beta}$

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

which can be written as

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

To prove this we consider LHS and expand

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

Adding and subtracting $\mathbb{E}\left[(\tilde{\boldsymbol{z}})\right]$ we get

$$\begin{aligned}\mathbb{E}\left[(\boldsymbol{z}-\tilde{\boldsymbol{z}})^2\right]&=\mathbb{E}\left[(\boldsymbol{f}+\boldsymbol{\epsilon}-\tilde{\boldsymbol{z}})^2\right]\\ &=\mathbb{E}[(\boldsymbol{f}+\boldsymbol{\epsilon}-\tilde{\boldsymbol{z}}+\mathbb{E}[\tilde{z}]-\mathbb{E}[\tilde{\boldsymbol{z}}])^2]\\ &=\mathbb{E}[(\boldsymbol{f}-\tilde{\boldsymbol{z}})^2+\mathbb{E}[\boldsymbol{\epsilon}^2]+2\mathbb{E}[(\boldsymbol{f}-\tilde{\boldsymbol{z}})\boldsymbol{\epsilon}]],\\ &=\mathbb{E}[\boldsymbol{f}-\tilde{\boldsymbol{z}})^2]+\boldsymbol{\sigma}_{\boldsymbol{\epsilon}^2}+0,\\ &=\mathbb{E}[\boldsymbol{f}-\tilde{\boldsymbol{z}})^2]+\boldsymbol{\sigma}_{\boldsymbol{\epsilon}^2}\end{aligned}$$

Using the linear property of expectation and independence of random variables ϵ and $\tilde{y}$ expectation sum is expanded and Also, When two random variables are independent ,the expectation of their product is equal to the product of their expectations.

$$\begin{aligned}\mathbb{E}[\boldsymbol{f}-\tilde{\boldsymbol{z}})^2]&=\mathbb{E}[(f-\mathbb{E}[\tilde{\boldsymbol{z}}]+\mathbb{E}[\tilde{\boldsymbol{z}}]-\tilde{\boldsymbol{z}})^2]\\ &=\underbrace{\mathbb{E}[(\mathbb{E}[\tilde{\boldsymbol{z}}]-\boldsymbol{f})^2]}_{\text{Bias}^2}+\underbrace{\mathbb{E}[(\tilde{\boldsymbol{z}}-\mathbb{E}[\tilde{\boldsymbol{z}}])^2]}_{\text{Variance}}\\ &-2\mathbb{E}[(\boldsymbol{f}-\mathbb{E}[\tilde{\boldsymbol{z}}])(\tilde{\boldsymbol{z}}-\mathbb{E}[\tilde{\boldsymbol{z}}])]\\ \\ &=\text{Bias}^2+\text{Var}-2\ \mathbb{E}[(\boldsymbol{f}-\mathbb{E}[\tilde{\boldsymbol{z}}])]\ \mathbb{E}(\tilde{\boldsymbol{z}}-\mathbb{E}[\tilde{\boldsymbol{z}}]\\ &=\text{Bias}^2+\text{Var}-2(\boldsymbol{f}-\mathbb{E}[\tilde{\boldsymbol{z}}])\ (\mathbb{E}[\tilde{\boldsymbol{z}}]-\mathbb{E}[\tilde{\boldsymbol{z}}])\end{aligned}$$

Thus we have

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