Parameterizing Peaks and Passes
Project 1 in FYS-STK4155

Mads Saua Balto, Astrid Bragstad Gjelsvik, and Oskar Hafstad
(Dated: October 15, 2023)

Github repo for this paper: `https://github.com/Madssb/FYS-STK-4155/tree/main/project_1`

This paper explores linear regression methods by fitting two-dimensional polynomials on two different datasets. We first try to find an optimal fit to Franke's function and then extend the analysis to real topographical data of Møsvatn Austfjell. The models we focus on are Ordinary Least Squares, Ridge regression and Lasso regression. We conduct an error analysis focusing on mean square error and R^2-score, and assess model performance using bootstrap and (5-10)-fold cross-validation resampling techniques. We find that for the Franke function, our best model estimate is given with a fifth order polynomial approximation, and that Ordinary Least Squares actually gives the best result, with Lasso performing worse.

I. INTRODUCTION

Have you ever looked across a beautiful, hilly landscape or a spectacular fjord and wondered how it could be described mathematically? Surely there must exist some polynomial function that can describe the essentials of that landscape's depths and heights! While natural terrain always will have more complexity than what a simple polynomial function can capture, how close such an approximation can actually get poses an interesting problem. Increasing the polynomial degree can always lead to capturing more and more of the terrain's complexities, but if the goal is to give a description of the general tendencies in the landscape, there should exist a point where increasing the polynomial degree stops describing the general landscape and starts describing the minor details. In other words, a mathematical model for the terrain should be able to predict the elevation of the landscape well also in places where elevation data has not informed the model. In this study, we will attempt to find such an ideal polynomial approximation for the region Møsvatn Austfjell in Norway, using linear regression models.

Linear regression models, while simple, can provide an adequate and powerful tool for predictions. Here, we test the applicability of both the ordinary least squares (OLS) model, the Ridge model and the Lasso regression models. In section II we will give an introduction to these methods, as well as a description of how we implement them to solve our problem. As our problem consists not only of fitting a model to a certain data set, but also of predicting elevation in areas that are "unseen" by the model, we will also give an introduction to the so-called Bias-Variance trade-off, which denotes the necessary trade-off that arises from reducing the model error as much as possible while simultaneously keeping it fit for future predictions. In section III we show first how our methods perform on the idealized Franke function, before we show the performance on the real terrain data from Møsvatn. This section also includes our discussion of the results, before we conclude in section IV. All code used to produce our results can be found in our github repository.

II. METHODS

In our problem, as in all supervised learning, we want to predict some real-valued output Y (e.g. elevation) based on an input vector $X = (X_1, X_2, \ldots, X_{p-1})^T$ (e.g. functions of latitude and longitude). The linear regression model for Y, $\tilde{Y}$, will then have the form

$$\tilde{Y} = \beta_0 + \sum_{j=1}^{p-1} X_j \beta_j = \sum_{j=0}^{p-1} X'_j \beta_j \tag{1}$$

where $X' = (1, X_1, \ldots X_{p-1})^T$ and β_j are coefficients that need to be estimated, using a set of training data [2]. Typically we do not only want to predict a single data point, and Y is a vector $\boldsymbol{y} \in \mathbb{R}^n$ which we predict based on an input matrix $\boldsymbol{X} \in \mathbb{R}^{n \times p}$. This matrix is usually called the feature matrix. In our case the p features in the feature matrix would correspond to the different terms in a polynomial approximation of a certain degree.

A. Ordinary Least Squares Regression

We define the prediction $\tilde{\mathbf{y}}$ by some model $\boldsymbol{\beta}$ for data $\mathbf{y}$:

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

where X is the features matrix, (seeII E 1). We define the cost function $C(\boldsymbol{\beta})$ which evaluates how well the model predicts data. If $\boldsymbol{X}^T\boldsymbol{X}$ is invertible, we may define $C(\boldsymbol{\beta})$ as the MSE (see **??**), which we then minimize to compute the optimal parameters as predicted by Ordinary least Squares (OLS) regression $\hat{\boldsymbol{\beta}}_{\text{OLS}}$:

$$\hat{\boldsymbol{\beta}}_{\text{OLS}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbf{y}. \tag{3}$$

A derivation for which can be found in A.

To find the sampling properties of $\boldsymbol{y}$ and $\hat{\boldsymbol{\beta}}_{\text{OLS}}$, we need to make some assumptions about the dataset. We assume that the observations $\boldsymbol{y}$ are uncorrelated and given by a continuous function $f(\boldsymbol{x})$ and a normal distributed error $\epsilon \sim \mathcal{N}(0, \sigma^2)$, in other words

$$\boldsymbol{y} = f(\boldsymbol{x}) + \epsilon. \tag{4}$$

The ordinary least squares (OLS) method then allows us to approximate this unknown function f with $\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}$. Making use of the assumption in (4) and the linearity of the expected value, the mean value of y_i will be given by

$$\mathbb{E}\left[y_i\right] = \mathbb{E}\left[f(x_i) + \epsilon_i\right] = \mathbb{E}\left[f(x_i)\right] + \mathbb{E}\left[\epsilon_i\right] \tag{5}$$

$$= \sum_j x_{ij}\beta_j = \boldsymbol{X}_{i,*}\boldsymbol{\beta}, \tag{6}$$

since $\mathbb{E}\left[\epsilon_i\right] = 0$, and $f(x_i) = \sum_j x_{ij}\beta_j$ has no stochasticity.

We can also find the variance of y_i, assuming that $f(x_i)$ and ϵ_i are uncorrelated variables, as follows

$$Var(y_i) = Var\left(f(x_i) + \epsilon_i\right) \tag{7}$$

$$= Var\left(f(x_i)\right) + Var\left(\epsilon_i\right) \tag{8}$$

$$= \sigma^2. \tag{9}$$

In other words, we have $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2)$. Moreover, using the expression for the optimized parameters $\hat{\beta}$ in the OLS method in (3), we can find that the expected values of these parameters are

$$\mathbb{E}\left[\hat{\boldsymbol{\beta}}\right] = \mathbb{E}\left[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbf{y}\right] \tag{10}$$

$$= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}\left[\mathbf{y}\right] \tag{11}$$

$$= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} \tag{12}$$

$$= \boldsymbol{\beta} \tag{13}$$

The variance of the parameters $\hat{\boldsymbol{\beta}}$ is given by

$$Var(\hat{\boldsymbol{\beta}}) = \mathbb{E}\left[\hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T\right] - \mathbb{E}\left[\hat{\boldsymbol{\beta}}\right]\mathbb{E}\left[\hat{\boldsymbol{\beta}}^T\right] \tag{14}$$

The untangling of this expression can be found in appendix C. Here we assume that the matrix $\boldsymbol{X}$ is invertible and that $\boldsymbol{X}\boldsymbol{X}^{-1}$ and $\boldsymbol{X}^T(\boldsymbol{X}^T)^{-1}$ therefore gives us the identity matrix. This gives us the solution

$$Var(\hat{\boldsymbol{\beta}}) = \sigma^2\left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-1} \tag{15}$$

In other words, we have $\hat{\boldsymbol{\beta}} \sim \mathcal{N}\left(\boldsymbol{\beta}, \sigma^2\left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-1}\right)$.

B. Shrinkage methods

While ordinary least squares is a popular and straightforward method for linear regression, it can exhibit high variance and therefore higher prediction error [2]. Ridge and Lasso regression are two linear regression methods that attempt to combat this problem by shrinking the regression coefficients by imposing a penalty on their size – and are therefore termed shrinkage methods.

1. Ridge Regression

In Ridge regression, the optimal parameters $\hat{\boldsymbol{\beta}}$ are found by minimizing a penalized residual sum of squares

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \underset{\beta}{\text{argmin}}\left\{\sum_{i=0}^{n-1}\left(y_i - \beta_0 - \sum_{j=1}^{p-1}x_{ij}\beta_j\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2\right\} \tag{16}$$

where $\lambda \geq 0$ is a parameter that controls the amount of shrinkage. The solution for the optimal $\hat{\boldsymbol{\beta}}_{\text{ridge}}$ will then be given by

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \left(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^T\mathbf{y}, \tag{17}$$

where $\boldsymbol{I}$ is the identity matrix. The larger the hyperparameter λ is, the more shrinkage it applies.

2. Lasso regression

Similarly to Ridge regression, the Lasso method introduces a constraint on coefficient sizes, but applied to the magnitude of the coefficients instead of the squares:

$$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \underset{\beta}{\text{argmin}}\left\{\frac{1}{2}\sum_{i=0}^{n-1}\left(y_i - \beta_0 - \sum_{j=1}^{p-1}x_{ij}\beta_j\right)^2 \tag{18}$$

$$+\lambda\sum_{j=1}^{p}|\beta_j|\right\} \tag{19}$$

There is no closed form expression for Lasso regression, but the estimates can be found numerically. Here we use the built-in function from Scikit-Learn [4].

C. Validation

When evaluating the performance of our model, we want to compare it to independent test data to evaluate how well it generalizes to new cases. We therefore split our available data into a training data set, used for finding the optimal model coefficients, and a testing data set that does not inform the model construction. This introduces another choice for the modeller, namely how much of the available data should be reserved for training, and how much should be reserved for testing. We will touch upon this subject more in section II C 2, but where not otherwise defined we use 4/5 of the data set for training data and 1/5 for test data.

1. Mean squared error and R^2-score

There are multiple metrics that can be used for evaluating the error between the independent observations

and the predicted output. The mean squared error is an extremely common choice, that we will also make use of here.

We define the mean squared error for some model $\tilde{\mathbf{y}}$ w.r.t some observed $\mathbf{y}$:

$$\mathrm{MSE}(\mathbf{y}, \tilde{\mathbf{y}}) \equiv \frac{1}{N} \sum_{i=0}^{N-1} (y_i - \tilde{y}_i)^2, \tag{20}$$

for $\mathbf{y}, \tilde{\mathbf{y}} \in \mathbb{R}^N$. We further define the R^2-score as

$$\mathrm{R}^2(\mathbf{y}, \tilde{\mathbf{y}}) \equiv 1 - \frac{\sum_{i=0}^{N-1} (y_i - \tilde{y}_i)^2}{\sum_{i=0}^{N-1} (y_i - \bar{y})^2}, \tag{21}$$

where $\bar{y}$ is the mean value of $\mathbf{y}$, $\bar{y} = \frac{1}{N} \sum_{i=0}^{N-1} y_i$.

2. K-Fold Cross-validation

Cross-validation is method for evaluation of model performance where instead of splitting our data into one training set and test set, we simply split it into K different subsets, and use one subset after another for testing, while the others are used for training. This should help us in case of any skewness in choosing our testing and training data. In this study we will also test for different number of subsets K, in other words different proportions between training and test data set size. Our algorithm for k-fold cross validation looks the following way

```
n is number of datapoints
k is number of subsets

shuffle the n datapoint indexes
split indexes into k groups where
  the residual of n/k (n%k) gives
  +1 element in the first n%k groups

for i in range(k):
    test indices = index group i
    train indices = all index groups except i
    find beta using train indices data
    make prediction for test indices data
    find error between test data
                    and prediction

find the mean of the error
```

3. Bootstrap

Bootstrapping is a tool we can use to improve the statistical accuracy of our prediction error. The general idea is to generate a sample of $\hat{\boldsymbol{\beta}}$s from the training set that we have, simply by randomly drawing data sets with replacement from the training data set a number of times, and using these data sets to calculate new $\hat{\boldsymbol{\beta}}$s. The predicted error can then be estimated as

```
X, Y is independent test data
number of bootstraps N
for n in N:
    resample training data set
    find beta using resampled training data
    make prediction with X using new beta
find mean predicted value with X
find error between Y and
    mean predicted value for X
```

D. The Bias-Variance Trade-off

When our model complexity increases, we can fit our model closer and closer to the training data, but our prediction error may start to increase due to a so-called "overfitting" of the model. Finding the balance between precision and overfitting is usually called the trade-off between model bias and variance. In this section we will investigate these terms deeper.

We find the optimal parameters $\hat{\boldsymbol{\beta}}$ by minimizing the cost function, or the mean squared error

$$C(\boldsymbol{X}, \boldsymbol{\beta}) = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \tilde{y}_i)^2 = \mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right]. \tag{22}$$

Diving deeper into this expression, we see that we can tease out different components. By again assuming that our observations $\boldsymbol{y}$ are given by a function $f(\boldsymbol{x})$ and a normal distributed error $\epsilon \sim \mathcal{N}(0, \sigma^2)$ (as in equation (4)), we can write

$$\mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right] = (f(\boldsymbol{x}) - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2 + Var(\tilde{\boldsymbol{y}}) + \sigma^2 \tag{23}$$

$$\approx \mathbb{E}\left[(\boldsymbol{y} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2\right] + Var(\tilde{\boldsymbol{y}}) + \sigma^2 \tag{24}$$

Here we also assume that $\tilde{\boldsymbol{y}}$ and $\boldsymbol{\epsilon}$ are independent, as the randomness in $\tilde{\boldsymbol{y}}$ comes from the random sampling of the training data and not from $\boldsymbol{\epsilon}$. All steps can be found in appendix B. We simply approximate the first term in the final equation $(f(\boldsymbol{x}) + \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2 \approx \mathbb{E}\left[(\boldsymbol{y} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2\right]$ because we do not have access to the function $f(\boldsymbol{x})$, only the data $\boldsymbol{y}$. Setting them equal would entail $\mathbb{E}\left[(\boldsymbol{y} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2\right] = \mathbb{E}\left[\boldsymbol{y} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right]\right]^2$.

The term $\mathbb{E}\left[(\boldsymbol{y} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2\right]$ can be interpreted as the *bias*, corresponding to the mean squared distance between the mean predicted values and the data points. The term $Var(\tilde{\boldsymbol{y}}) = \mathbb{E}\left[(\tilde{\boldsymbol{y}} - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2\right]$, on the other hand, is the variance of the predicted values when the training data sample changes. Finally, the σ^2 component of the mean squared error is the variance of the noise in the model and constitutes an irreducible error that depends on the noisyness of the data itself.

E. Data Description

1. Features Matrix

In the following we define a feature matrix for a two-dimensional input data set, consisting of $\boldsymbol{x}$ and $\boldsymbol{y}$ values. Note therefore that from here on we will use the symbol y to describe input data, and not the dependent variable. First, we define a row vector containing features for a pth degree polynomial of x_i and y_j:

$$u(x_i, y_j) = \left[x_i^0[y_j^0 \cdots y_j^p] \;\; x_i[y_j^0 \cdots y_j^{p-1}] \;\; \cdots \;\; x_i^p\right]. \quad (25)$$

Next, we define the design matrix X which represents the set of all possible combinations of x and y values for our input data. The design matrix is constructed by evaluating the function $u(x_i, y_j)$ at each x and y data point and indexing the resulting row vectors. X has dimensions $n \times m$ and is defined as follows:

$$X = \begin{bmatrix} u(x_0, y_0) \\ u(x_0, y_1) \\ \vdots \\ u(x_0, y_{m-1}) \\ \vdots \\ u(x_{n-1}, y_0) \\ u(x_{n-1}, y_1) \\ \vdots \\ u(x_{n-1}, y_{m-1}) \end{bmatrix} \quad (26)$$

2. The Franke function

In this study we approximate the Franke function, which is defined as follows

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

using polynomial approximation in the domain $x, y \in [0, 1]$. In this study we explore adding some noise to the function, and add a noise term $\eta \sim \mathcal{N}(0, 1)$ which we scale with a factor of 0.1 to retain the similarity to the Franke function.

3. Terrain data acquisition

We acquire our terrain data for Møsvatn Austfjell via the github repository provided by Morten Hjorth-Jensen

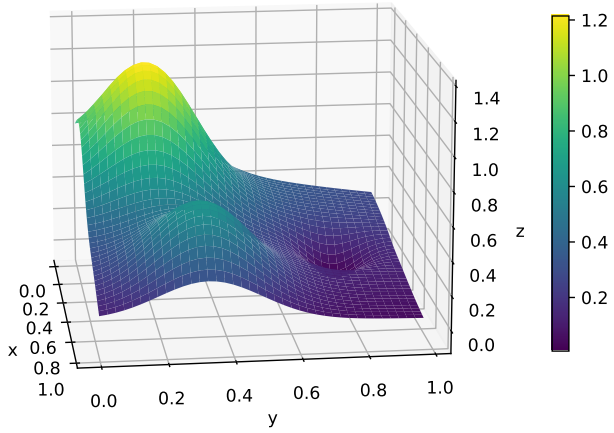

Figure 1. The Franke function for 40×40 uniformly distributed values of $x, y \in [0, 1]$.

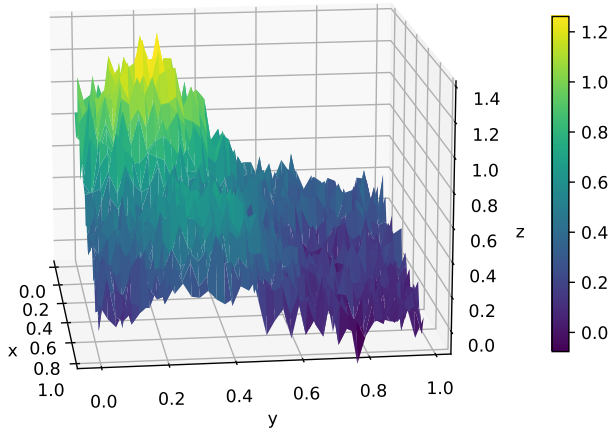

Figure 2. The Franke function for 40×40 uniformly distributed values of $x, y \in [0, 1]$ with noise η applied to it.

[3]. These are again provided by the U.S. Geological Survey's open-access elevation data who hosts the data via the EarthExplorer service [1].
The specific area of the terrain data we analyse are the following data coordinates $x, y \in [500, 1000, 500, 1000]$. Then out of computational considerations and to damped noise in the data, we down sample the data grid by a factor of 30. This results in a dataset of 34×34 datapoints.

4. Scaling

We are interested in reducing any error and model strangeness that could arise from having too large magnitude and variance in the training data. We therefore

use two methods to scale data.

- *Centering*: We center the data around the mean values in each feature matrix column, and do the same for the corresponding observations we wish to make a prediction for.. This entails setting the intercept to zero.
- *Standardization*: After centering, we standardize the data by dividing by the standard deviation in each column of the feature matrix, except for the first column which corresponds to the intercept. Effectively, we set the standard deviation of the data set to 1. We do the same for the corresponding observations we wish to make a prediction for.

While our terrain data has a variance with relatively large magnitude, the Franke function is between zero and one in the domain where we investigate it. Dividing by the standard deviation would therefore increase the variance instead of reducing it, and it is therefore counterproductive to perform standardization for the Franke function. Since the Ridge and Lasso methods introduce a penalizing factor, we center the data in these methods for both Franke and the terrain data, as we do not wish to apply a penalizing factor to the intercept. For simplicity, we therefore apply centering to all methods, including least squares, as this should not impare our results. We apply standardization only to the terrain data.

III. RESULTS AND DISCUSSION

In the following section we present the performance of the linear regression models Ordinary Least Squares, Ridge and Lasso. In section III A you will find how the models perform for the Franke function, with applied noise $\eta \sim \mathcal{N}(0,1)$. In section III B you will find how the model performs on terrain data from Møsvatn Austfjell.

A. The Franke function

In figure 3 you can find the performance of the ordinary least squares (OLS) model presented as the mean squared error (MSE). We investigate polynomial approximations up to the fifth degree. We can see that the model performs increasingly well for higher order polynomials. The minimum MSE is for a fifth-order approximation, but the difference between four and five degrees is not large. The lowest MSE is at 0.14, which is a low value value informing us that we already capture many of the features of the Franke function at this stage.

For the R^2-score, the model performance also increases with polynomial degree, finding a maximum around a fifth-order polynomial approximation as well. Here, $R^2 = 0.867$ for degree $= 5$.

In figure 4 we have visualized the optimal parameters $\hat{\boldsymbol{\beta}}$ arrived at with OLS regression for each polynomial degree. We can see that the optimal parameters increase drastically and grow large as the polynomial degree increases, with the $\hat{\boldsymbol{\beta}}$'s almost spanning coefficient values between -30 and 30 for a fifth-order polynomial approximation. While for lower orders the values are quite constrained. This makes sense, as the model complexity increases, the model will try to cover more points accurately so one can expect a greater spread of the coefficient values. The error bars of the coefficient values are drawn from the standard deviation $\mathrm{Var}[\hat{\boldsymbol{\beta}}]^{1/2} = \sigma$ as defined in C11.

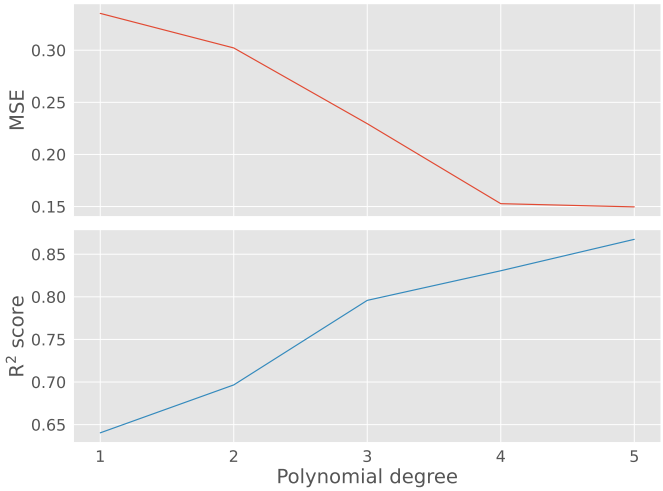

Figure 3. Mean Squared Error and R^2 score for OLS regression with polynomial approximation to the Franke function, using 40 $\times$40 datapoints.

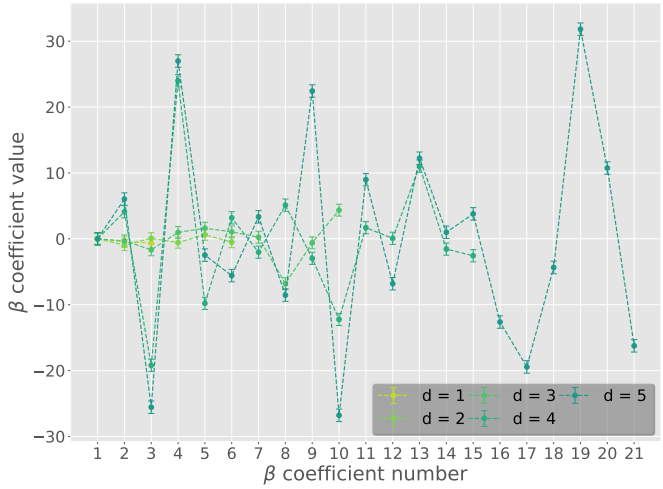

Figure 4. The β parameters associated with the polynomial approximations of the Franke function up to degree, $d = 5$. Each point has been plotted with an error margin given by its associated standard deviation σ.

Moving on to Ridge regression, we see that we get approximately the same values of MSE for Ridge regression as for OLS for the lower penalty parameters λ (see figure 5. However, as the parameters increase towards 1, the MSE begins to increase also for the higher polynomial approximations. This is a sign that the penalty parameter is too high, making the model lose not only excessively large coefficients, but also coefficients that matter.

The same can be seen for the R^2-score in figure 6. The highest R-score actually never goes beyond what we found for OLS, which was as high as $R^2 = 0.867$. This indicates that the Ridge regression does not perform better than OLS for the considered polynomial approximations.

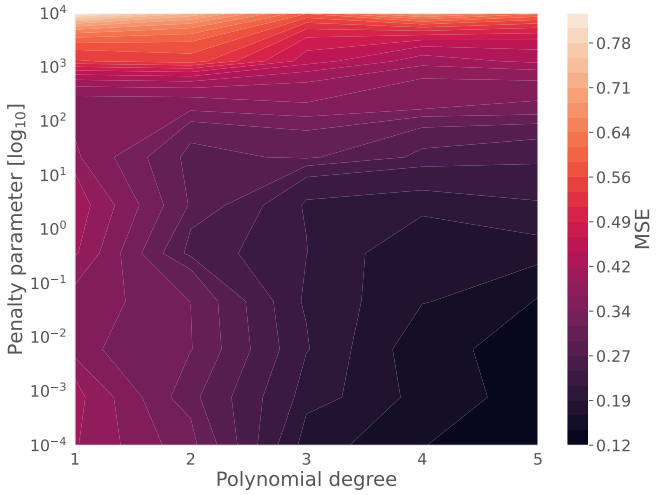

Figure 5. Mean Squared Error for Ridge Regression with polynomial approximation to the Franke function, using 40 ×40 datapoints.

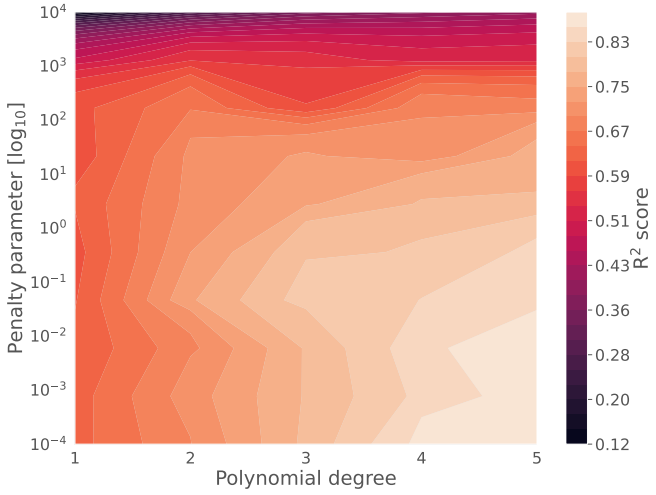

Figure 6. R^2 score for Ridge Regression with polynomial approximation to the Franke function, using 40 ×40 datapoints.

In figure 7 we see that the Lasso MSE deviates slightly further from the OLS results. The MSE never goes below 0.2, as opposed to OLS and Ridge, and for higher λs, the MSE approaches 1. Similarly for the R^2-score in figure 8, the method does not show any R^2 value higher than 0.81, which is worse than both OLS and Ridge. In conclusion, it appears as though Lasso regression is too strict a method in this case.

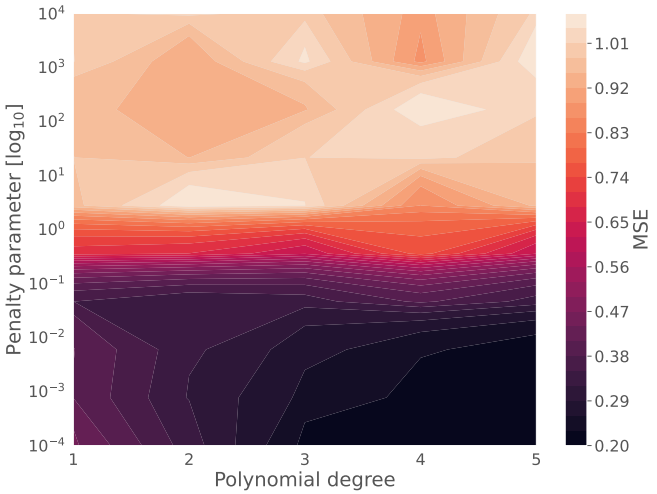

Figure 7. Mean Squared Error for Lasso Regression with polynomial approximation to the Franke function, using 40 ×40 datapoints.

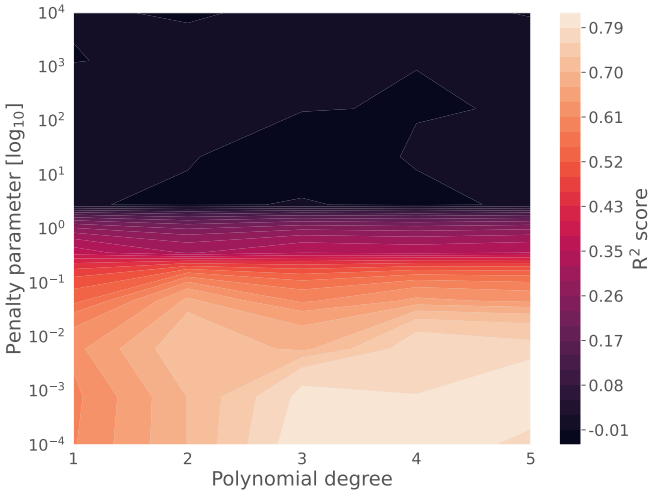

Figure 8. R^2 score for Lasso Regression with polynomial approximation to the Franke function, using 40 ×40 datapoints.

Next we wish to do a bias-variance trade-off analysis of the Franke function.To investigate this, we plot the bias, variance and MSE we obtain doing an OLS regression and see how it changes with increasing model complexity. To increase the accuracy of the assessment we do this while utilizing the bootstrap resampling method. We do

500 bootstrap resamplings to get results we are confident in. In 9 we see that for lower degree approximations the error is associated with the bias, but with increasing model complexity the error becomes a result of the increasing variance. Given this relationship, we have to decide the trade-off of what model complexity is ideal. In this case, the optimal case with low bias and low variance appears to be at a polynomial degree of 5. We reduced the number of data points to 20×20, as we saw that with a higher number of data points (eg. 40×40), both the bias and variance did not diverge until very high model complexities were reached.

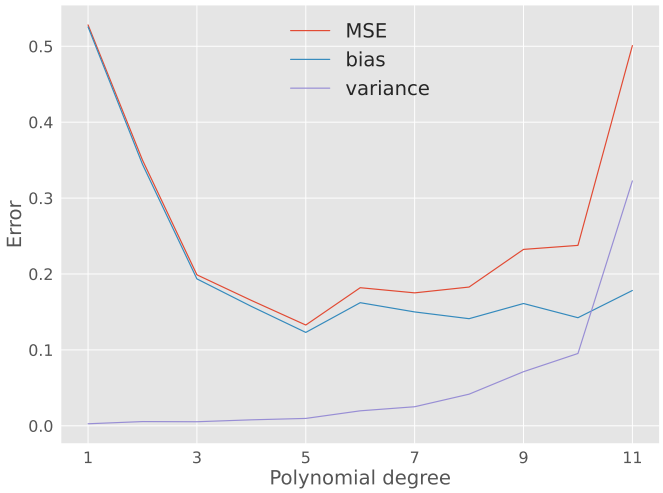

Figure 9. Bias-variance trade-off analysis on a 20×20 Franke function. Used OLS regression with 500 bootstrap samples. Evaluated polynomial models up to degree d=11.

In figure 10 and figure 11, the mean mean squared error for the Ridge and Lasso regression can be found. We can see that there are no clear large differences between the bootstrapped MSEs for these model, compared to the non-bootstrapped ones in figure 5 and 7. This is expected, as we are still comparing model with the same, albeit resampled, data set.

In figure 12, we plot the resulting OLS MSE when comparing a cross-validated data set for divisions into between 5 and 10 subsets. The MSE is remarkably similar for the different number of subsets up to a degree 10. At this point we know from our bias-variance analysis in figure 9 that the model complexity is already too high, giving us high variance and increasing MSE for a certain training data set. This gives us reassurance that our model performance is not too dependent training and test data split in our first case, and that a 5-fold cross-validation seems to work well for our data set.

We therefore apply a 5-fold cross-validation to investigate the cross-validated MSE for Ridge and Lasso. This can be found in figure 13 and 14. We can see again that the Ridge regression performs better than the Lasso regression, and see again that the strictness of Lasso

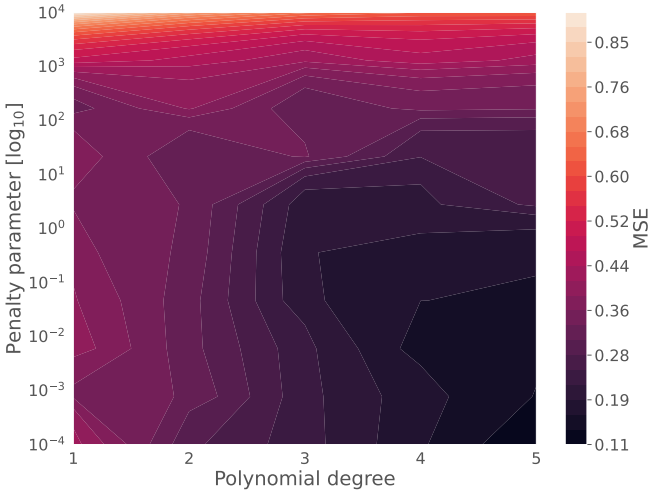

Figure 10. Ridge regression analysis on a 40x40 Franke function using 500 bootstrap samples to identify optimal model parameters.

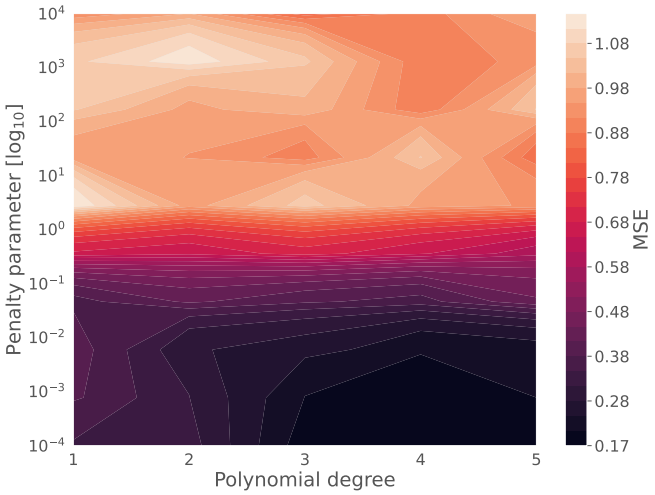

Figure 11. Lasso regression analysis on the Franke function using 500 bootstrap samples to identify optimal model parameters.

causes an even worse performance for the extremely high penalty parameters.

B. Terrain data

Having tested and verified our regression methods on the idealized case of the Franke function we will now apply them to our digital terrain data. We see in 15 that the β coefficient values don't behave that differently than from what we saw with Franke's function. If anything the lower coefficient values are a bit more constrained for the highest order approximation. In 16 we see a similar trend as we did for Franke, hinting that as we increase the

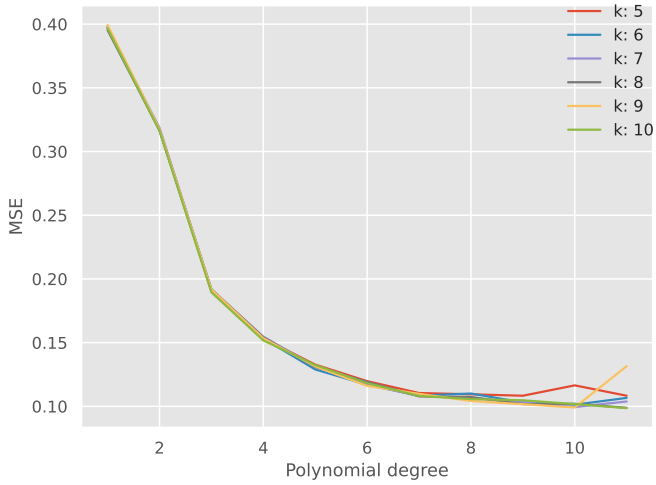

Figure 12. OLS regression on the Franke function using cross-validation resampling with $k \in [5, 10]$ folds.

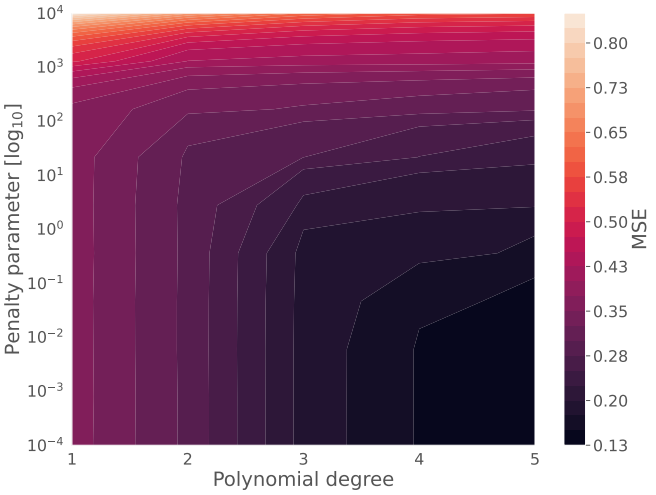

Figure 13. Ridge regression on a 40x40 Franke function using 5-fold cross-validation.

model complexity the error will further decrease. This is also reflected in 21. For the Ridge regression on the terrain data 17 we get a more complex parameter space than for Franke, but this is to be expected with the more varied features of the terrain. The MSE is generally higher for the terrain than for the Franke function, which is also to be expected as the terrain data is much more complex.

We don't see that big a difference on the overall performance off the Lasso regression 19 compared to Ridge, but we expect any such differences would become more apparent if we checked for higher order polynomials.

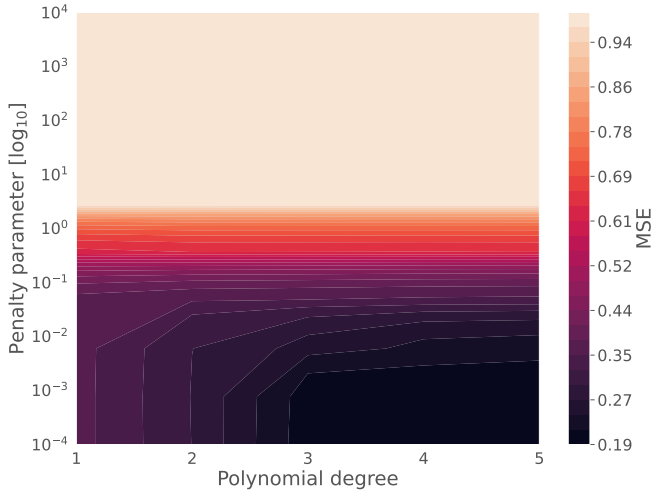

Figure 14. Lasso regression on a 40x40 Franke function using 5-fold cross-validation

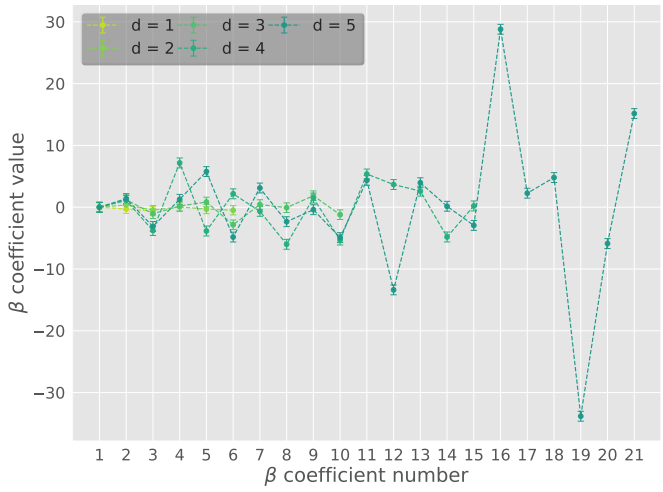

Figure 15. The β parameters associated with the polynomial approximations of the terrain data up to degree, $d = 5$.

1. *Bootstrap*

2. *Cross validation*

IV. CONCLUSION

For our analysis on Franke we in general get better results with Ridge regression. OLS provides slightly lower R^2-score than Ridge for the lower penalty parameters around 10^{-4}. As the penalty paramter increases beyond 1 the MSE increases drastically also for the higher polynomial orders in both Lasso and Ridge, which is to be expected as these are very high penalty parameters to add. For further studies, it would have been interesting to perhaps investigate even lower penalty parameters. Lasso performs slightly worse than Ridge, suggesting per-

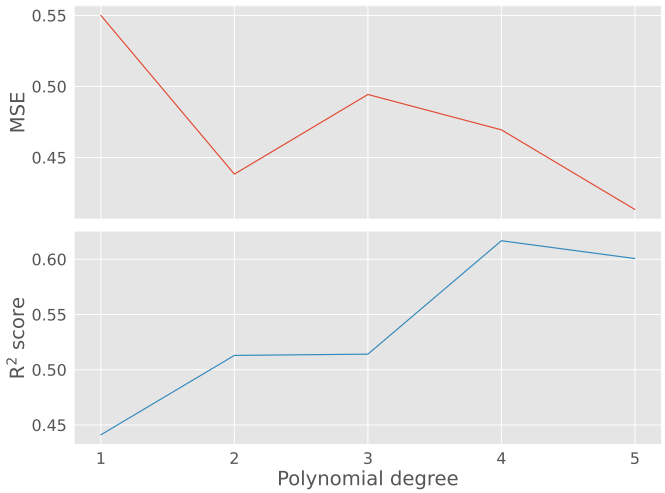

Figure 16. Mean Squared Error and R^2 score for OLS regression with polynomial approximation to the terrain data, using 34 ×34 datapoints.

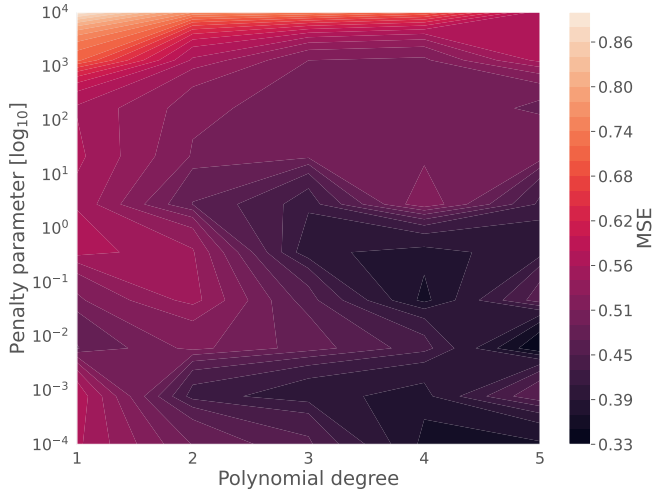

Figure 17. Mean Squared Error for Ridge Regression with polynomial approximation to the terrain with 34 ×34 datapoints.

haps that the ability of Lasso to set parameters directly to zero leads to a too strict model in our case. The performed bias-variance analysis we have performed suggests that our best polynomial fit for Franke is of around a fifth-order approximation, after which the variance of the model starts to dominate the error. The cross-validation performed suggests that this result is robust.

A main limiting factor in our analysis of the terrain data has been not investigating higher order polynomials for all of the regression analyses. So for future work we envision analysing the terrain data over much greater model complexities.

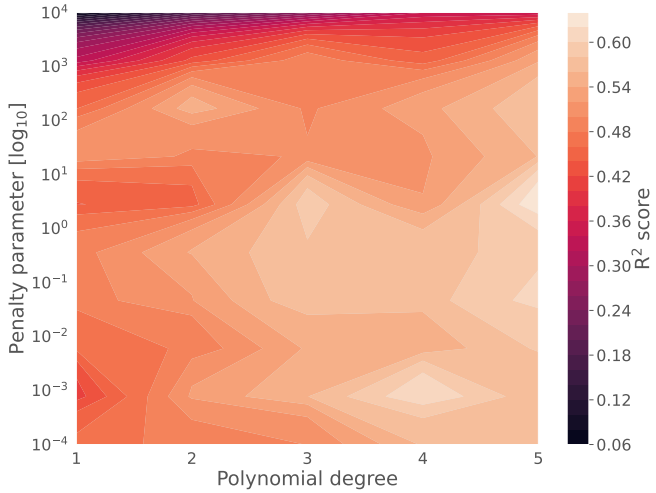

Figure 18. R^2 score for Ridge Regression with polynomial approximation to the terrain with 34 ×34 datapoints.

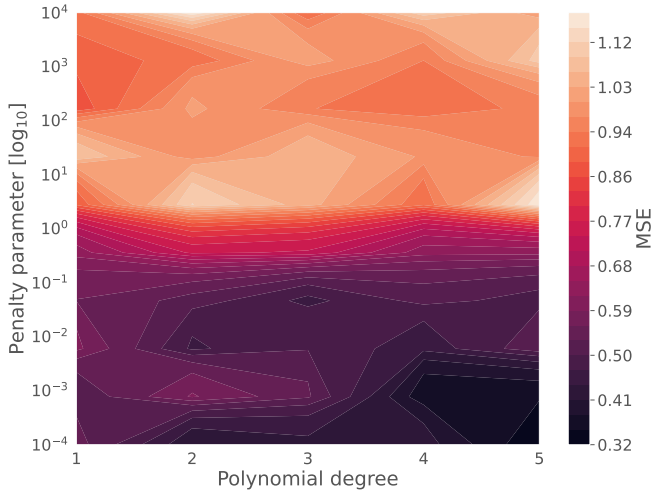

Figure 19. Mean Squared Error for Lasso Regression on the terrain with 34 ×34 datapoints.

REFERENCES

[1] US Gov. Earthexplorer. `https://earthexplorer.usgs.gov/`, 2023. Last accessed 13 October 2023.

[2] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York, NY, 2009.

[3] Morten Hjorth-Jensen. Project 1 terrain data. `https://github.com/CompPhysics/MachineLearning/tree/master/doc/Projects/2023/Project1/DataFiles`, 2023.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

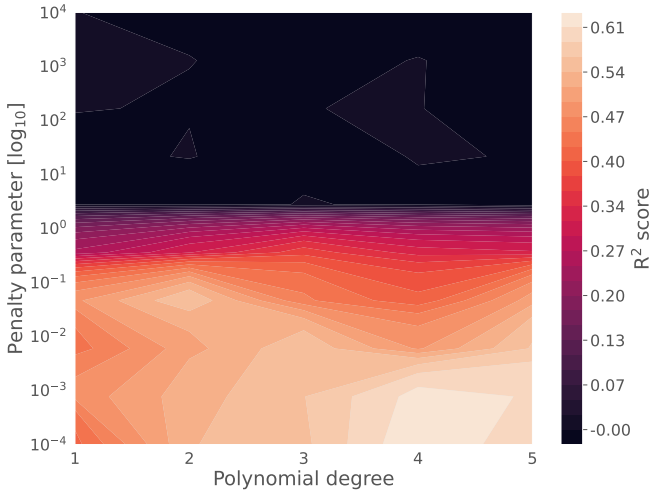

Figure 20. R^2 score for Lasso Regression on the terrain with 34 $\times$34 datapoints.

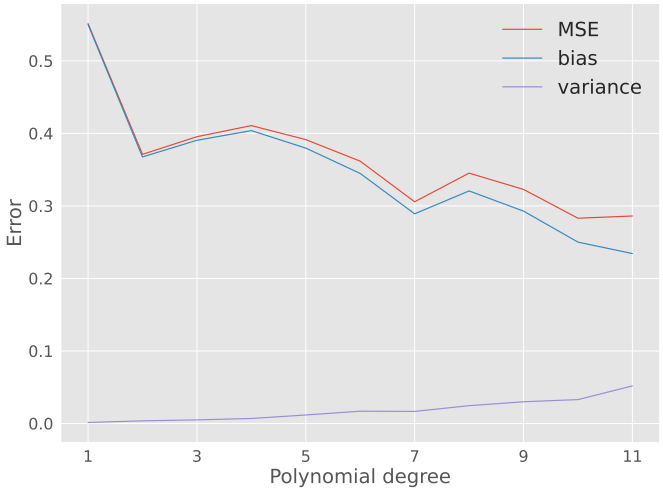

Figure 21. BVT analysis of the terrain data.

Appendix A: Deriving ordinary least squares

We may use the mean squared error (MSE) to define some cost function C to measure the quality of some model:

$$
\begin{aligned}
C &\equiv \frac{1}{N}\sum_{i=0}^{N}(y_i - \tilde{y}_i)^2, \\
&= \frac{1}{N}(\mathbf{y} - \tilde{\mathbf{y}})^T(\mathbf{y} - \tilde{\mathbf{y}}), \\
&= \frac{1}{N}(\mathbf{y} - X\boldsymbol{\beta})^T(\mathbf{y} - X\boldsymbol{\beta}),
\end{aligned}
$$

where N is the number of samples, $\mathbf{y}$ is the observed data, and X are the corresponding features. We define the optimal parameters $\boldsymbol{\beta}_{\text{opt}} \equiv \text{argmin}\left(\frac{\partial C(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}\right)$, and solve

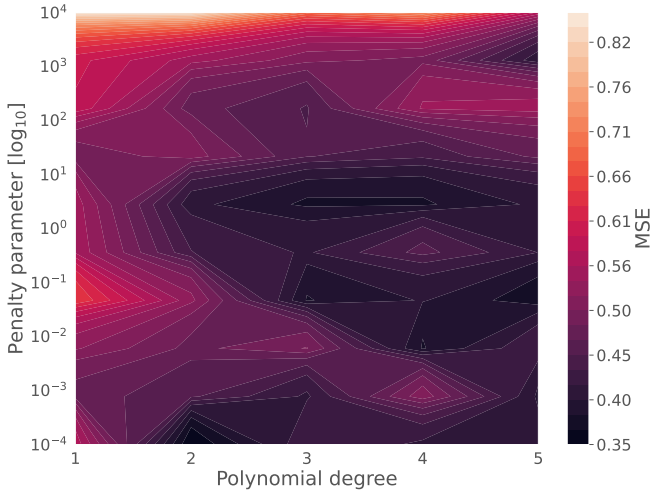

Figure 22. Ridge regression analysis of terrain data using 500 bootstrap samples.

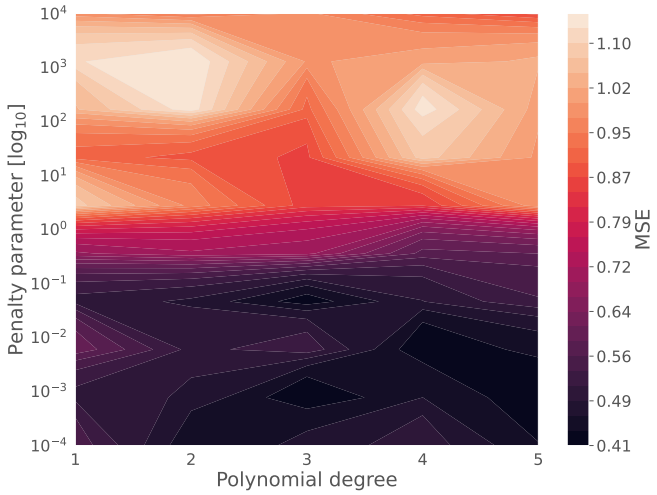

Figure 23. Lasso regression analysis of terrain data using 500 bootstrap samples.

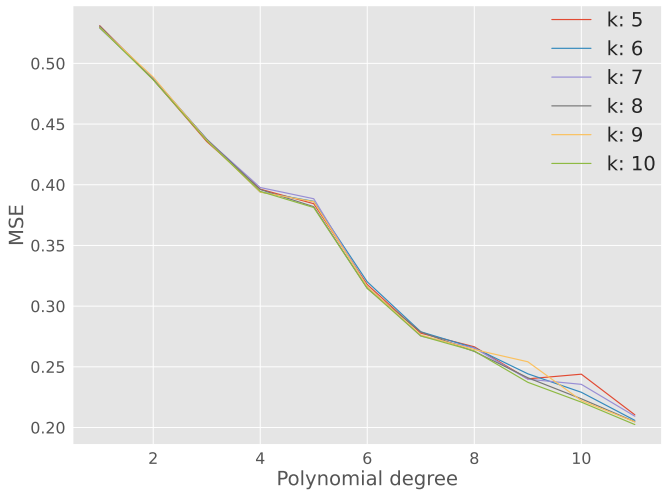

Figure 24. OLS regression on the terrain data using cross-validation resampling with $k \in [5, 10]$ folds.

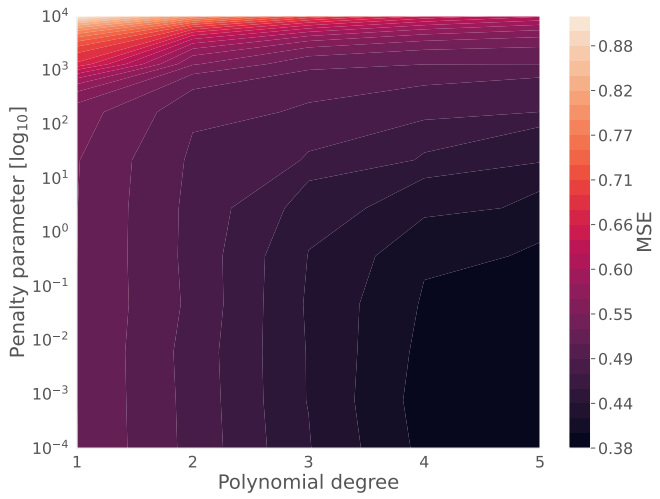

Figure 25. Ridge regression analysis on terrain data using cross-validation resampling with 5 kfolds.

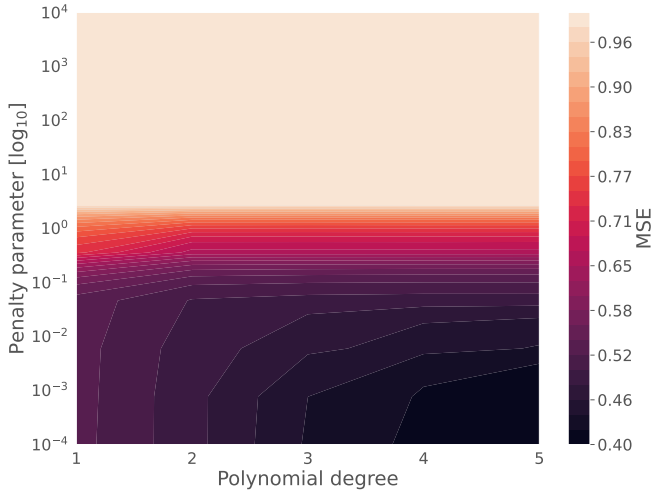

Figure 26. Lasso regression analysis on terrain data using cross-validation resampling with 5 kfolds.

for $\frac{\partial C(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$:

$$
\begin{aligned}
\frac{\partial C(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \frac{\partial}{\partial \boldsymbol{\beta}} \frac{1}{N} (\mathbf{y} - X\boldsymbol{\beta})^T (\mathbf{y} - X\boldsymbol{\beta}), \\
&= \frac{\partial}{\partial \boldsymbol{\beta}} \sum_{i=0}^{N-1} \frac{1}{N} (y_i - \sum_{j=0}^{N-1} x_{ij}\beta_j)^2, \\
&= \frac{1}{N} \sum_{k=0}^{N-1} \frac{\partial}{\partial \beta_k} \sum_{i=0}^{N-1} (y_i - \sum_{j=0}^{N-1} x_{ij}\beta_j)^2 \hat{\mathbf{e}}_k, \\
&= \frac{1}{N} \sum_{k=0}^{N-1} \sum_{i=0}^{N-1} \frac{\partial}{\partial \beta_k} (y_i - \sum_{j=0}^{N-1} x_{ij}\beta_j)^2 \hat{\mathbf{e}}_k, \\
&= \frac{1}{N} \sum_{k=0}^{N-1} \sum_{i=0}^{N-1} 2(y_i - \sum_{j=0}^{N-1} x_{ij}\beta_j)(-x_{ik}) \hat{\mathbf{e}}_k, \\
&= \frac{2}{N} \sum_{k=0}^{N-1} \sum_{i=0}^{N-1} x_{ik} x_{ij} \beta_j - x_{ik} y_i, \\
&= \frac{2}{N} (X^T \boldsymbol{\beta} X - X^T \mathbf{y}).
\end{aligned}
$$

Solve for $\frac{\partial C(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0$:

$$
\begin{aligned}
&\frac{2}{N} (X^T \boldsymbol{\beta}_{\text{opt}} X - X^T \mathbf{y}) = 0 \\
&\Rightarrow X^T \boldsymbol{\beta}_{\text{opt}} X - X^T \mathbf{y} = 0 \\
&\Rightarrow \boldsymbol{\beta}_{\mathbf{opt}} = (X^T X)^{-1} X^T \mathbf{y}.
\end{aligned}
$$

As such: $\boldsymbol{\beta}_{\text{opt}} = (X^T X)^{-1} X^T \mathbf{y}$.

Appendix B: Deriving bias-variance terms

The full derivation of the bias variance terms can be found underneath.

$$\mathbb{E}\left[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2\right] = \mathbb{E}\left[(\boldsymbol{y}^2 - 2\boldsymbol{y}\tilde{\boldsymbol{y}} + \tilde{\boldsymbol{y}}^{\boldsymbol{2}})\right] \tag{B1}$$

$$= \mathbb{E}\left[\boldsymbol{y}^2\right] - 2\mathbb{E}\left[\boldsymbol{y}\tilde{\boldsymbol{y}}\right] + \mathbb{E}\left[\tilde{\boldsymbol{y}}^{\boldsymbol{2}}\right] \tag{B2}$$

Assuming that the data $\boldsymbol{y}$ comes from the model $f(\boldsymbol{x})+\boldsymbol{\epsilon}$, we rewrite the first term in (B2) as

$$\mathbb{E}\left[\boldsymbol{y}^2\right] = \mathbb{E}\left[\left(f(\boldsymbol{x})+\boldsymbol{\epsilon}\right)^2\right] \tag{B3}$$

$$= \mathbb{E}\left[f(\boldsymbol{x})^2\right] + \mathbb{E}\left[2f(\boldsymbol{x})\boldsymbol{\epsilon}\right] + \mathbb{E}\left[\boldsymbol{\epsilon}^{\boldsymbol{2}}\right] \tag{B4}$$

Since $f(\boldsymbol{x})$ is a non-stochastic function, we simply have $\mathbb{E}\left[f(\boldsymbol{x})\right] = f(\boldsymbol{x})$, while $\mathbb{E}\left[\boldsymbol{\epsilon}\right] = 0$, as always. Using the definition of variance, we can also see that

$$\mathbb{E}\left[\boldsymbol{\epsilon}^{\boldsymbol{2}}\right] = Var(\boldsymbol{\epsilon}) + \left(\mathbb{E}\left[\boldsymbol{\epsilon}\right]\right)^2 = \sigma^2 + 0 = \sigma^2 \tag{B5}$$

We can therefore see that we have

$$\mathbb{E}\left[\boldsymbol{y}^2\right] = \mathbb{E}\left[f(\boldsymbol{x})^2\right] + 2\mathbb{E}\left[f(\boldsymbol{x})\boldsymbol{\epsilon}\right] + \mathbb{E}\left[\boldsymbol{\epsilon}^{\boldsymbol{2}}\right] \tag{B6}$$

$$= f(\boldsymbol{x})^2 + +2f(\boldsymbol{x})\cdot 0 + \sigma^2 \tag{B7}$$

$$= f(\boldsymbol{x})^2 + \sigma^2 \tag{B8}$$

Moving on to the second term in (B2), we can write it as

$$\mathbb{E}\left[\boldsymbol{y}\tilde{\boldsymbol{y}}\right] = \mathbb{E}\left[\left(f(\boldsymbol{x})+\boldsymbol{\epsilon}\right)\tilde{\boldsymbol{y}}\right] \tag{B9}$$

$$= f(\boldsymbol{x})\mathbb{E}\left[\tilde{\boldsymbol{y}}\right] + \mathbb{E}\left[\boldsymbol{\epsilon}\tilde{\boldsymbol{y}}\right] \tag{B10}$$

$$\tag{B11}$$

Making the assumption that $\tilde{\boldsymbol{y}}$ and $\boldsymbol{\epsilon}$ are independent, we can write

$$\mathbb{E}\left[\boldsymbol{y}\tilde{\boldsymbol{y}}\right] = f(\boldsymbol{x})\mathbb{E}\left[\tilde{\boldsymbol{y}}\right] + \mathbb{E}\left[\boldsymbol{\epsilon}\right]\mathbb{E}\left[\tilde{\boldsymbol{y}}\right] \tag{B12}$$

$$= f(\boldsymbol{x})\mathbb{E}\left[\tilde{\boldsymbol{y}}\right] + 0\cdot\mathbb{E}\left[\tilde{\boldsymbol{y}}\right] \tag{B13}$$

$$= f(\boldsymbol{x})\mathbb{E}\left[\tilde{\boldsymbol{y}}\right] \tag{B14}$$

Finally, for the third term in (B2), we can use the variance defintion again to write

$$\mathbb{E}\left[\tilde{\boldsymbol{y}}^{\boldsymbol{2}}\right] = Var(\tilde{\boldsymbol{y}}) + \left(\mathbb{E}\left[\tilde{\boldsymbol{y}}\right]\right)^2 \tag{B15}$$

Returning to (B2), we get (all steps can be found in appendix B)

$$\begin{aligned}
\mathbb{E}\left[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2\right] &= \mathbb{E}\left[\boldsymbol{y}^2\right] - 2\mathbb{E}\left[\boldsymbol{y}\tilde{\boldsymbol{y}}\right] + \mathbb{E}\left[\tilde{\boldsymbol{y}}^{\boldsymbol{2}}\right] \\
&= f(\boldsymbol{x})^2 + \sigma^2 - 2f(\boldsymbol{x})\mathbb{E}\left[\tilde{\boldsymbol{y}}\right] + Var(\tilde{\boldsymbol{y}}) + \left(\mathbb{E}\left[\tilde{\boldsymbol{y}}\right]\right)^2 \\
&= \left(f(\boldsymbol{x})^2 - 2f(\boldsymbol{x})\mathbb{E}\left[\tilde{\boldsymbol{y}}\right] + \left(\mathbb{E}\left[\tilde{\boldsymbol{y}}\right]\right)^2\right) \\
&\qquad + Var(\tilde{\boldsymbol{y}}) + \sigma^2 \\
&= \left(f(\boldsymbol{x}) - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right]\right)^2 + Var(\tilde{\boldsymbol{y}}) + \sigma^2
\end{aligned}$$

Appendix C: Variance of the parameters in least square regression

The variance of the parameters $\hat{\boldsymbol{\beta}}_{\text{OLS}}$ is given by

$$Var(\beta) = \mathbb{E}\left[\beta\beta^T\right] - \mathbb{E}\left[\beta\right]\mathbb{E}\left[\beta^T\right] \tag{C1}$$

Here we show how to find the expression for the variance, using the OLS expression for $\boldsymbol{\beta}$ in equation (3). Untangling $\beta\beta^T$, using the fact that $\left((\boldsymbol{X}^T\boldsymbol{X})^{-1}\right)^T = (\boldsymbol{X}^T\boldsymbol{X})^{-1}$, gives us

$$\beta\beta^T = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbf{y}\left((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbf{y}\right)^T \tag{C2}$$

$$= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbf{y}\mathbf{y}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}, \tag{C3}$$

which entails that

$$\mathbb{E}\left[\beta\beta^T\right] = \mathbb{E}\left[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbf{y}\mathbf{y}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\right] \tag{C4}$$

$$= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}\left[\mathbf{y}\mathbf{y}^T\right]\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} \tag{C5}$$

Another way to express $\mathbb{E}\left[\mathbf{y}\mathbf{y}^T\right]$ is by again making use of the expression for variance as follows

$$\mathbb{E}\left[\mathbf{y}\mathbf{y}^T\right] = Var(\boldsymbol{y}) + \mathbb{E}\left[\boldsymbol{y}\right]\mathbb{E}\left[\boldsymbol{y}\right]^T \tag{C6}$$

$$= \sigma^2 + \boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{X}^{\boldsymbol{T}} \tag{C7}$$

$$= \boldsymbol{X}\left(\boldsymbol{X}^{-1}\sigma^2\left(\boldsymbol{X}^{\boldsymbol{T}}\right)^{-1} + \boldsymbol{\beta}\boldsymbol{\beta}^{\boldsymbol{T}}\right)\boldsymbol{X}^T \tag{C8}$$

$$= \boldsymbol{X}\left(\sigma^2\left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-1} + \boldsymbol{\beta}\boldsymbol{\beta}^{\boldsymbol{T}}\right)\boldsymbol{X}^T \tag{C9}$$

Here we assume that the matrix $\boldsymbol{X}$ is invertible and that $\boldsymbol{X}\boldsymbol{X}^{-1}$ and $\boldsymbol{X}^T(\boldsymbol{X}^T)^{-1}$ therefore gives us the identity matrix. Inserting equation (C9) into equation (C5) will lead to cancellations on both sides of the expression in the parenthesis, $\left(\sigma^2\left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-1} + \boldsymbol{\beta}\boldsymbol{\beta}^{\boldsymbol{T}}\right)$, leading the to expression

$$\mathbb{E}\left[\beta\beta^T\right] = \sigma^2\left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-1} + \boldsymbol{\beta}\boldsymbol{\beta}^{\boldsymbol{T}} \tag{C10}$$

and thereby the solution

$$Var(\hat{\boldsymbol{\beta}}) = \mathbb{E}\left[\beta\beta^T\right] - \mathbb{E}\left[\beta\right]\mathbb{E}\left[\beta^T\right] \tag{C11}$$

$$= \sigma^2\left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-1} + \boldsymbol{\beta}\boldsymbol{\beta}^{\boldsymbol{T}} - \boldsymbol{\beta}\boldsymbol{\beta}^T \tag{C12}$$

$$= \sigma^2\left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-1} \tag{C13}$$

Appendix D: Supplementary figures