Exploring Polynomial Regression Methods for Predicting Terrain Data

Adam Jakobsen, Daniel Haas, Nils Taugbøl, Vilde Ung*
University of Oslo, Department of Physics

(Dated: January 13, 2023)

Ordinary least squares, Ridge and LASSO regression are implemented on two data sets: one generated by the Franke function and the other sampled from real-world terrain data from Norway. The models are evaluated using both bootstrapping and k-fold cross-validation in order to estimate the expected prediction error. We find that the results are highly data-dependent, particularly when modelling terrain data. Ridge regression yielded the best results for modelling both the Franke function and the terrain data. Our analysis showed that the optimal model for the terrain data resulted in high complexity relative to sample size, which could indicate over-fitting and poor generalization to other terrain samples.

Keywords: Regression, Resampling Techniques, Machine Learning

I. INTRODUCTION

Regression models are a fundamental part of statistical science, giving rise to simple yet powerful methods for inference and prediction. In machine learning, we are interested in making models that can make accurate predictions on new data, based on the data that are available for training. Linear regression models assume a functional relationship between the response variable and the predictor variables, where the model parameters are linearly related to the mean of the response. Finding the optimal parameters for the model involves minimizing the cost function, typically using least squares methods. To control over-fitting, one can additionally introduce a regularization term, as in Ridge and LASSO regression. The goal of regularization is to constrain the size of the parameters in the model, thereby reducing its variance.

An important aspect of modeling is evaluating how well the model generalizes to new data. Resampling techniques, like bootstrap and cross-validation, exploit the asymptotic behavior of statistics computed on multiple random samples. This allows us to assess how well the model is expected to generalize, by estimating the expected prediction error. This is important when dealing with limited amounts of data, since generating new data can be expensive or even impossible. By using resampling techniques, we get more statistically trustworthy models.

In this article, we explore polynomial regression methods with and without regularization. The methods are tested on a smooth surface given by the Franke function. In this case, we are able to generate data freely. The methods are further applied to real terrain data. We estimate the relationship between the xy-coordinates and the terrain height using polynomial regression models, exploring how the polynomial degree and regularization impacts the accuracy of the model.

First, we introduce the methodology of polynomial regression and the resampling methods used. We derive relevant analytical expressions for the linear model, and show how the expected prediction error can be decomposed into bias and variance terms. Second, we present and discuss our findings for the synthetic data, establishing that our methodology is reliable. Finally, we present and discuss our findings for real-world terrain data.

II. METHODS

A. Data

The goal of this article is to evaluate three different types of polynomial regression for predicting terrain data. To do this, we will first test the methods on the two-dimensional Franke function, defined as

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

for $x, y \in [0,1]$. In Figure 1 we have plotted the Franke function. We will use $f(x,y)+\epsilon$, with $\epsilon \sim \mathcal{N}(0,\sigma)$, to generate the synthetic data set, while the real-world terrain data was collected from `https://earthexplorer.usgs.gov/`. The surface of the chosen terrain data can be seen in figure 2. For fitting and testing the model, we use 25×25 data points of the map.

* Repository: https://github.com/ungvilde/FYS-STK4155

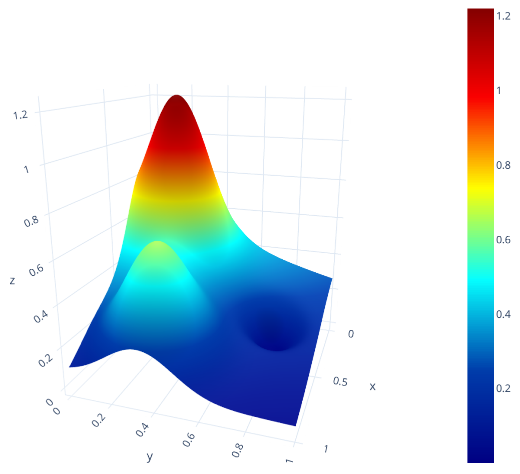

Figure 1. The Franke function is a well-known, smooth function. It is used for testing and evaluating the methodology in this report.

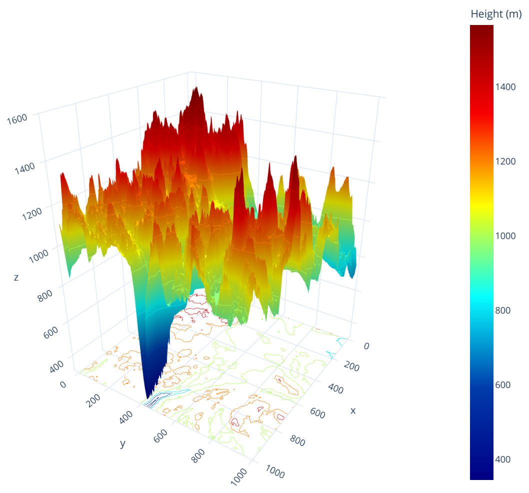

Figure 2. A landscape from Rogaland county, Norway. This is the terrain data that we have chosen to fit. We see the landscape surface with its contour lines below. Notice that the terrain is uneven with many variations.

B. Introducing the Linear Model

We will consider a linear model of a multivariate polynomial of degree d. Given a data set $\mathcal{D} = \{(x_i, y_i, z_i) \,|\, i = 0, \ldots, n-1\}$, our basic assumption is that there exists some function that describes the dependent variable z_i, such that

$$z_i = f(x_i, y_i) + \epsilon_i. \tag{2}$$

Here, $f(x_i, y_i)$ is a function of the independent variables x_i and y_i, and ϵ_i is a noise term. We assume $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. We model f as a polynomial of degree d, so we assume z_i can be approximated by

$$\hat{z}_i = \sum_{k=0}^{d} \sum_{\ell=0}^{k} \beta_{k-\ell,\ell} x_i^{k-\ell} y_i^{\ell}. \tag{3}$$

Each term, referred to as *monomials*, in the polynomial has degree less than or equal to d, meaning that for each $x_i^a y_i^b$ we have $a + b \leq d$. Rewriting (3) in terms of vectors gives

$$\hat{z}_i = \mathbf{x}_i^T \boldsymbol{\beta},$$

where $\mathbf{x}_i^T = \left(1, x_i, y_i, \ldots, x_i^d, y_i^d\right)^T$ is a row vector containing monomials of x_i and y_i up to degree d, and $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_{p-1})$ is a column vectorcontaining the coefficient of each monomial. Note that we from now on index the β-coefficients by $i = 0, \ldots, p-1$, where

$$p = \binom{d+2}{2} = \frac{(d+2)(d+1)}{2}.$$

The variable p denotes the number of *features* in our model, where each monomial is incorporated as a feature. With larger p, the model complexity increases, making the fit more flexible.

In matrix form, the model is given by

$$\mathbf{z} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}. \tag{4}$$

Here, $\mathbf{X}$ is the *design matrix* containing the feature values for all n observations. The vectors $\mathbf{x}_i^T$ are stacked on top of each other, so that we get

$$\mathbf{X} = \begin{bmatrix} 1 & x_0 & y_0 & x_0 y_0 & \ldots & x_0^d & y_0^d \\ 1 & x_1 & y_1 & x_1 y_1 & \ldots & x_1^d & y_1^d \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n-2} & y_{n-2} & x_{n-2} y_{n-2} & \ldots & x_{n-2}^d & y_{n-2}^d \\ 1 & x_{n-1} & y_{n-1} & x_{n-1} y_{n-1} & \ldots & x_{n-1}^d & y_{n-1}^d \end{bmatrix}.$$

For notational simplicity, let $\mathbf{X}_{i,*}$ refer to row i of $\mathbf{X}$. The size of the design matrix is then $n \times p$, and the length of the $\boldsymbol{\beta}$-vector is p. $\mathbf{z}$ and $\boldsymbol{\epsilon}$ are column vectors of length n containing the values of each z_i and ϵ_i, respectively.

With the model given by (4), we see that the expectation of z_i is

$$\mathbb{E}(z_i) = \mathbb{E}(\mathbf{X}_{i,*}\boldsymbol{\beta}) + \mathbb{E}(\epsilon_i) = \mathbf{X}_{i,*}\boldsymbol{\beta}, \tag{5}$$

which follows from the fact that $\mathbf{X}_{i,*}\boldsymbol{\beta}$ is deterministic (non-random), and $\mathbb{E}(\epsilon_i) = 0$. Furthermore, we find that the variance of z_i is

$$\mathrm{Var}(z_i) = \sigma^2.$$

The full derivation may be found in Appendix A. From this, we see that $z_i \sim \mathcal{N}(\mathbf{X}_{i,*}\boldsymbol{\beta}, \sigma^2)$.

C. Ordinary Least Squares

In order to estimate the optimal parameters $\boldsymbol{\beta}$ for our model, we minimize the squared error of the model. To do this, we define a cost function and find the values of $\boldsymbol{\beta}$ where its derivative is zero. The cost function is given by

$$\begin{aligned} C(\boldsymbol{\beta}) &= \frac{1}{n}\sum_{i=0}^{n-1}(z_i - \hat{z}_i)^2 \\ &= \frac{1}{n}(\mathbf{z} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{z} - \mathbf{X}\boldsymbol{\beta}). \end{aligned}$$

This is a convex function, which implies that we want to solve

$$\frac{\partial C(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \mathbf{X}^T(\mathbf{z} - \mathbf{X}\boldsymbol{\beta}) = 0. \tag{6}$$

Solving (6) for the optimal $\boldsymbol{\beta}$ gives

$$\begin{aligned} \mathbf{X}^T(\mathbf{z} - \mathbf{X}\boldsymbol{\beta}) &= 0 \\ \mathbf{X}^T\mathbf{z} - \mathbf{X}^T\mathbf{X}\boldsymbol{\beta} &= 0 \\ \mathbf{X}^T\mathbf{X}\boldsymbol{\beta} &= \mathbf{X}^T\mathbf{z} \\ \hat{\boldsymbol{\beta}} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{z}, \end{aligned} \tag{7}$$

provided that the inverse $(\mathbf{X}^T\mathbf{X})^{-1}$ is well-defined. We denote the optimal parameter by $\hat{\boldsymbol{\beta}}$.

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

$$\begin{aligned} \mathbb{E}(\hat{\boldsymbol{\beta}}) &= \mathbb{E}\left[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{z}\right] \\ &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbb{E}(\mathbf{z}) \\ &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T X\boldsymbol{\beta} \\ &= \boldsymbol{\beta}, \end{aligned}$$

which shows that $\hat{\boldsymbol{\beta}}$ is an unbiased estimator (meaning the expected difference between $\hat{\boldsymbol{\beta}}$ and $\boldsymbol{\beta}$ is zero). We also find the variance of $\hat{\boldsymbol{\beta}}$ as

$$\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}. \tag{8}$$

The full derivation of (8) may be found in Appendix A. We can use these results to compute confidence intervals for the parameters $\boldsymbol{\beta}$. When n is large, we can safely assume that $\hat{\boldsymbol{\beta}} \sim \mathcal{N}(\boldsymbol{\beta}, \sigma(\mathbf{X}^T\mathbf{X})^{-1})$, which follows from the asymptotic properties of the estimator. Then a 95% confidence interval can be computed as

$$\hat{\beta}_j \pm 1.96 \cdot \mathrm{SE}(\hat{\beta}_j) \tag{9}$$

$$\mathrm{SE}(\hat{\beta}_j) = \hat{\sigma}\sqrt{[(\mathbf{X}^T\mathbf{X})^{-1}]_{j,j}} \tag{10}$$

$$\hat{\sigma}^2 = \frac{\sum_{i=0}^{n-1}(z_i - \hat{z}_i)^2}{n-p}. \tag{11}$$

We note that (11) is an unbiased estimator of σ^2. See [1, p. 47] for further details on confidence intervals for $\boldsymbol{\beta}$.

D. Ridge and LASSO Regression

With Ridge regression, we introduce the hyperparameter $\lambda \geq 0$, which penalizes the sum-of-squares of the parameters $\boldsymbol{\beta}$. That is, we find the optimal parameters with respect to the cost function

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

Here, $\|\cdot\|_2$ denotes the $\mathcal{L}_2$-norm. Solving $\frac{\partial C(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0$ for $\boldsymbol{\beta}$ gives us the Ridge estimator

$$\hat{\boldsymbol{\beta}}_{\mathrm{Ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{z}, \tag{13}$$

where $\mathbf{I}$ is the $p \times p$ identity matrix.

With LASSO regression we have a slightly different penalty term in the cost function,

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

In this case, $\|\cdot\|_1$ is the $\mathcal{L}_1$-norm. Because of this, the cost function is C^0 – its derivative is discontinuous – and there is no closed-form, analytical expression for the LASSO estimator of $\boldsymbol{\beta}$. In this article, we perform LASSO regression using the `scikit-learn` library, so we will not go into further detail on this.

E. Bias-Variance Trade-Off

We want to find a regression fit $\hat{f}(x, y)$ that approximates the true function $f(x, y)$, where the fit is based on minimizing the mean squared error (MSE) on some training data set $\mathcal{D}$. Ideally $\hat{f}$ should generalize, such that it can make accurate predictions for samples outside our training sample. The expected prediction error is given by

$$\begin{aligned}\mathbb{E}(\mathrm{MSE}) &= \mathbb{E}\left(\frac{1}{n}\sum_{i=0}^{n-1}(z_i-\hat{z}_i)^2\right)\\ &= \frac{1}{n}\sum_{i=0}^{n-1}\mathbb{E}[(z_i-\hat{z}_i)^2], \end{aligned}\tag{15}$$

which can be further decomposed into

$$\mathbb{E}[(z_i-\hat{z}_i)^2] = [\mathrm{Bias}(\hat{z}_i)]^2 + \mathrm{Var}(\hat{z}_i) + \sigma^2. \tag{16}$$

The expected prediction error is made up of a *bias* term, a *variance* term, and an *irreducible error* term. For the derivation of (16), see Appendix A. To be clear, this is an expectation which averages $\hat{f}$ over all possible training samples: the fit $\hat{f}$ depends on $\mathcal{D}$, and $\mathcal{D}$ is a random sample.

The bias term captures how well the mean of the fit $\hat{z}_i$ (ranging across different training data) agrees with the true function f that we wish to model. Even with an infinite amount of training data, the model will have bias due to the simplifying assumptions we make about it (such as assuming the model is linear). The bias is typically small for complex, highly flexible models, and large for over-simplified (under-fitted) models. The squared bias is given by

$$[\mathrm{Bias}(\hat{z}_i)]^2 = [f_i - \mathbb{E}(\hat{z}_i)]^2. \tag{17}$$

The variance term describes how much the fit fluctuates around its mean. In other words, it measures how much the fitted values change with different training data. It tends to grow with model complexity (with over-fitting), when the model is "over-specialized" and captures the noise of the data rather than regularities. The variance term is

$$\mathrm{Var}(\hat{z}_i) = \mathbb{E}[(\mathbb{E}(\hat{z}_i) - \hat{z}_i)^2]. \tag{18}$$

Finally, there is the irreducible error term σ^2, which comes from the noise inherent to the data. This term cannot be reduced, no matter how well we estimate the true f, and is in effect a lower bound on the MSE of our model. From (16) it is clear that to find the best possible fit, we need to minimize both the bias and the variance of our model.

F. Resampling Techniques

In order to estimate the accuracy of our model, we use resampling techniques, namely *bootstrapping* and *cross-validation*. The main idea is that by resampling the training data, we can compute the test error with regression fits based on varying samples of training data.

1. Bootstrap

For a given data set, we randomly split it into a training set and a test set. Let $\mathcal{T}$ be the training data, with T data points. There are four main steps in our bootstrapping procedure:

1. Sample T data points from $\mathcal{T}$ *with replacement.*
2. Let $\mathcal{T}^{(i)}$ be the new data set containing the randomly sampled values.
3. Estimate $\boldsymbol{\beta}$ using the data in $\mathcal{T}^{(i)}$, giving $\hat{\boldsymbol{\beta}}^{(i)}$.
4. Repeat 1.-3. B times.

After B iterations, we have a collection of B regression fits. Based on these fits, we compute the test error on a separate test data set of $n-T$ points. Note that the test data are at no point included in the resampling procedure. We have B fits, which yield B different MSE values. Finally, we compute the average across the bootstrap samples. Let $\mathbf{X}_{j,*}\hat{\boldsymbol{\beta}}^{(i)} = \hat{z}_j^{(i)}$ be the prediction of z_j using the ith bootstrap sample for training. The estimated test error is

$$\overline{\mathrm{MSE}} = \frac{1}{B}\sum_{i=1}^{B}\left(\frac{1}{n-T}\sum_{j=0}^{n-T-1}[z_j-\hat{z}_j^{(i)}]^2\right).$$

Similarly, we compute the bias and variance terms as

$$\begin{aligned}\overline{\mathrm{Bias}} &= \frac{1}{n-T}\sum_{j=0}^{n-T-1}\left(z_j - \frac{1}{B}\sum_{i=1}^{B}\hat{z}_j^{(i)}\right)^2\\ \overline{\mathrm{Var}} &= \frac{1}{n-T}\sum_{j=0}^{n-T-1}\left[\frac{1}{B}\sum_{i=1}^{B}\left(\hat{z}_j^{(i)} - \frac{1}{B}\sum_{i=1}^{B}\hat{z}_j^{(i)}\right)^2\right].\end{aligned}$$

Note that we use the data values z_i rather than the function values $f(x_i, y_i)$ in order to compute the bias, as we generally do not know the true f.

2. Cross-Validation

With k-fold cross validation, we partition the data set into k roughly equally sized portions (after it has been randomly shuffled, to correct for any bias in the ordering of the data). At each iteration, one of the portions is dedicated to testing while the other $k-1$ are dedicated to training the model. We use the training data to get a regression fit. Then, the fit is used to compute the MSE on the test sample. This procedure is iterated k times, such that each portion is used as a test sample once. Finally, the average MSE is computed, giving an estimate of the expected prediction error.

G. Feature Scaling

Before finding the regression fit of our model, we scale the data. Specifically, we standardize it so that each feature column in the design matrix is centered around 0 and has standard deviation equal to 1. In mathematical terms, that means

$$x^*_{i,j} = \frac{x_{i,j} - \bar{x}_j}{\sigma_j}, \tag{19}$$

where $x_{i,j}$ is an element in the design matrix, $\bar{x}_j$ is the mean value of column j in the design matrix, and σ_j is the standard deviation of column j. There are a few advantages to this. First, it improves interpretability. By standardizing, we can understand the value of β_0 to be the expected value of z if the features are set to their mean value. Second, it ensures that the feature values will be roughly within the same order of magnitude, improving the numerical stability of our method. Finally, scaling is particularly important for Ridge and LASSO regression, as the λ parameter penalizes large $\boldsymbol{\beta}$ coefficients [2, p. 119]. This implies that with unscaled data we might unfairly penalize certain features.

III. RESULTS AND DISCUSSION

A. Franke Function

We generated a data set based on the Franke function (1) with a noise term $\epsilon \sim \mathcal{N}(0, 0.1)$. For this, a total number of $n = 25 \times 25 = 625$ data points was used.

1. *Ordinary Least Squares*

We first fit polynomial regression models using the ordinary least squares method, for polynomials up to fifth order. The resulting $\hat{\boldsymbol{\beta}}$ parameters and their corresponding 95% confidence intervals are shown in Figure 3. The figure shows that the variance of the optimal parameters increases with polynomial degree (i.e with increasing model complexity), and also that the estimated parameters get larger values.

In Figure 4, the mean squared errors are shown for both the training data and test data, up to polynomial degree $d = 10$. From this figure, we can assess the regions where there is high variance or high bias. Where both training and test errors are high, the bias tends to be large. On the other hand, if the training error becomes small but the test error increases, the variance increases.

It is often useful to plot the test data MSE together with the R^2 metric, as done in Figure 17 in appendix

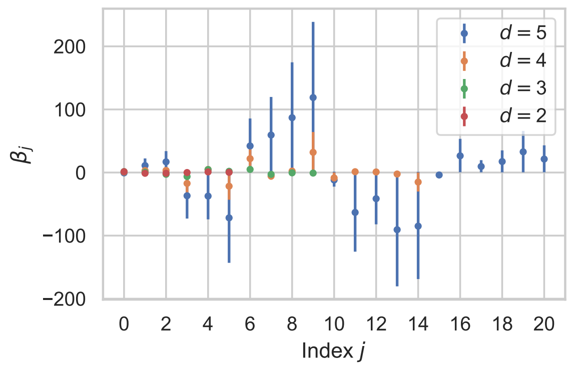

Figure 3. The optimal β_j coefficients of OLS polynomial regression are plotted, using polynomial degree up to $d = 5$. The corresponding 95% confidence intervals are included. We have used a data set based on the Franke function with normal noise ($\sigma = 0.1$) and $n = 25 \times 25 = 625$ data points.

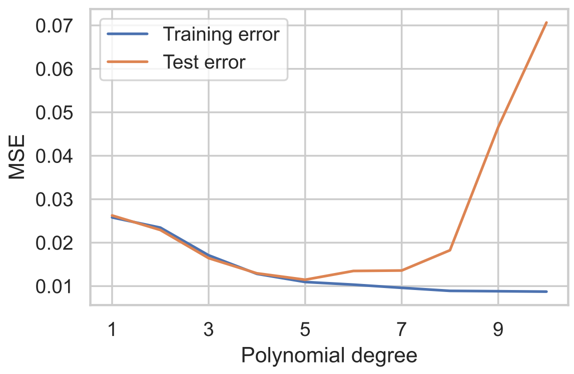

Figure 4. The figure shows the MSE computed within a training sample, and the MSE computed on a test sample, as a function of polynomial degree. The test error is based on 100 bootstrap samples. The data set is based on the Franke function with a normal noise term, where $\sigma = 0.1$ and $n = 625$. The fit is made using OLS regression.

B, since this metric is a way of assessing the predictive power of the model [3, 115]. We see that the MSE and R^2 both indicate that the model becomes more accurate with greater complexity (up to $d = 5$).

To further analyze the bias-variance trade-off, we plotted the test MSE as a function of polynomial degree, together with the bias and variance. The estimates were computed from 100 bootstrap samples. Figure 5 indicates that a polynomial fit of a degree in the range of 5-7 is optimal. Beyond that range, the variance starts to increase, indicating a regime of overfitting. The ever-decreasing value of the bias for higher-degree models also reiterates the trade-off aspect discussed in section II E. We find that we have the best polynomial fit with

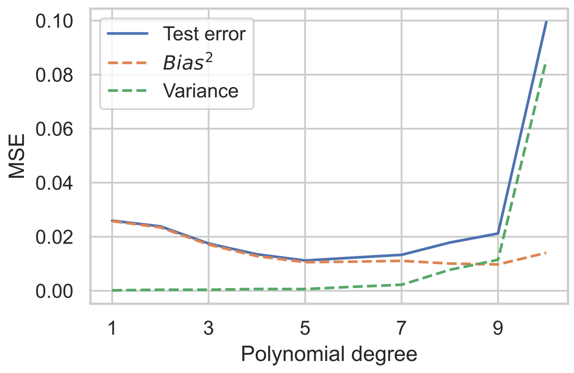

Figure 5. MSE as a function of polynomial degree, together with the bias and variance. The values have been computed from 100 bootstrap samples, using OLS regression. The data set is based on values from the Franke function with a normal noise term, where $\sigma = 0.1$ and $n = 625$.

degree 5. This gave

$$\text{MSE} = 1.20 \cdot 10^{-2} \pm 0.05 \cdot 10^{-2}.$$

We also explored the bias-variance trade-off for a varying number of data points. We found that the MSE and its behavior in relation to the model complexity were highly influenced by the number of data points n. In the case where n was small ($n \leq 100$), the MSE was large for larger degrees ($d > 7$). In the case where n was large ($n \geq 10.000$), the bias term dominated the mean squared error for low degrees. With increasing polynomial degree, the variance and the MSE remained small (at least up to $d = 25$).

Finally, we compared the MSE found using bootstrap and cross-validation, to assess that the resampling methods agree in their estimates of the expected prediction error. In figure 6 we can see that bootstrap and cross-validation agree at first but appear to diverge in terms of the MSE values for greater polynomial orders. With a larger number of sample data, we can expect the methods to diverge less, as the Central Limit Theorem ensures there will be less variance in the estimated MSE [2, 592] regardless of the resampling method. It is important to notice that, while the bootstrapped MSE seems to increase significantly after polynomial degree 8, cross-validation shows only a slight rise. This could indicate that cross-validation is less sensitive to measuring overfitting or simply be a consequence of the number of folds and samples.

2. *Ridge Regression*

We fit Ridge regression models, performing a grid search for the optimal polynomial degree d and regularization parameter λ. The mean squared errors were then computed using 10-fold cross-validation. The resulting contour plot with the ranges for the polynomial degrees and λ values can be seen in Figure 7. When comparing the optimal λ and polynomial degree d that we find with cross-validation, with the values we find using bootstrapping, we see that the two methods do not agree on the optimal d and λ. When bootstrapping, we find that $\log(\lambda) = -5.6$ and $d = 7$, while with cross-validation we find $\log(\lambda) = -10.3$ and $d = 8$. The contour plot for the bootstrapped MSE can be found in Appendix B, Figure 19.

Although there is a difference between values of d and $\log(\lambda)$ for bootstrap and cross-validation, we should notice the similarity between the minimum error in both

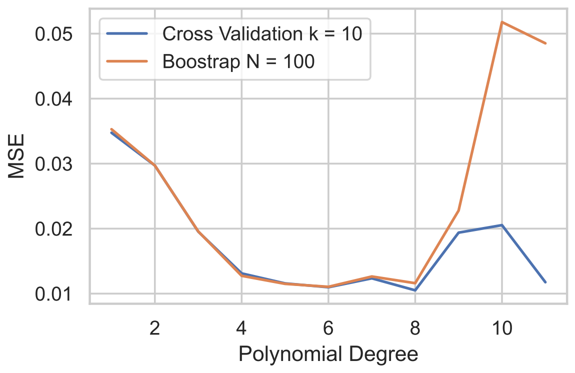

Figure 6. Estimates of MSE, from 100 bootstrap samples and cross-validation with k = 10 folds, as a function of polynomial degree. We are here using $n = 625$ and $\sigma = 0.1$.

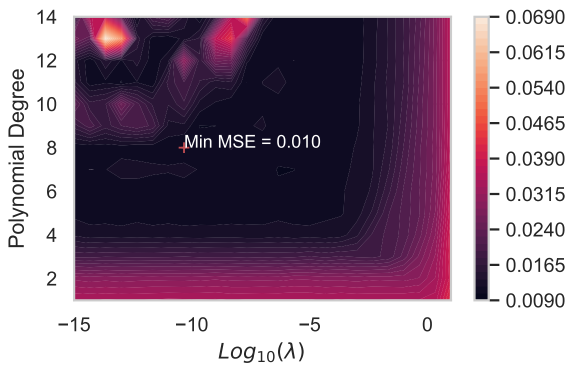

Figure 7. The colored grid shows the MSE as a function of the regularization parameter $\log(\lambda)$ and polynomial degree d. The red cross marks the location of the smallest MSE = 0.01, which has corresponding $d = 8$ and $\log(\lambda) = -10.3$. The MSE was computed using 10-fold cross-validation. The data set is based on the Franke function with a normal noise term $\sigma = 0.1$ and has $n = 25 \times 25 = 625$ data points.

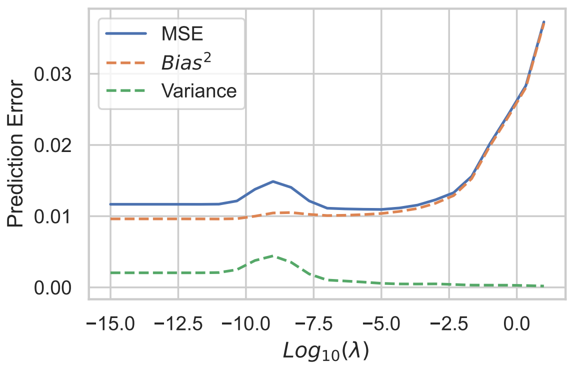

Figure 8. The MSE, bias and variance as a function of the logarithm of the regularization parameter λ. The estimates were computed from 100 bootstrap samples, with polynomial degree $d = 10$.

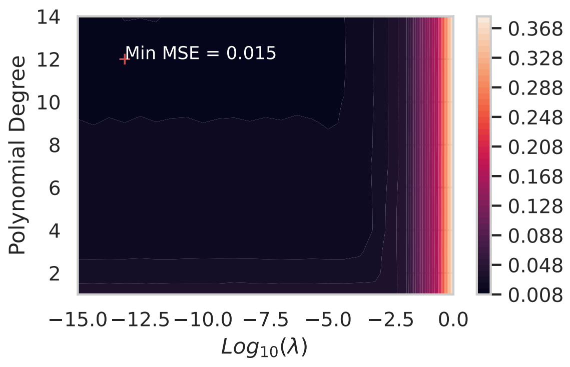

Figure 9. The colored grid shows the MSE as a function of the logarithm of the regularization parameter λ and polynomial degree d. The red cross marks the location of the smallest MSE $= 0.015$, which has corresponding $d = 12$ and $\log(\lambda) = -13.125$. The MSE was computed using LASSO regression with 10-fold cross validation. The data set is based on the Franke function with a normal noise term $\sigma = 0.1$ and has $n = 25 \times 25 = 625$ data points.

resampling methods. This might indicate there is not a clear minimum that optimizes the model further, and the models are approaching the desired limit of having expected prediction error equal to the irreducible term, as mentioned in subsection II E.

Both resampling methods estimate the minimal MSE to be approximately 0.01. More specifically, cross-validation gives an optimal MSE of

$$\mathrm{MSE} = 1.03 \cdot 10^{-2} \pm 0.15 \cdot 10^{-2}.$$

We also investigated the bias-variance trade-off for varying λ values using $d = 8$, which is the optimal model complexity found with cross-validation. We computed the MSE, bias and variance using 100 bootstrap samples, for varying values of λ. The results can be seen in Figure 8, where we have plotted the estimates as a function of $\log(\lambda)$. We see that the bias term dominates the MSE, and that the variance decreases with larger λ.

3. *LASSO Regression*

We fit LASSO regression models on the Franke function data, again doing a grid search to find the optimal polynomial degree and regularization parameter λ. Figure 9 shows the minimization of the MSE in terms of the model complexity and regularization parameter with 10-fold cross validation. The optimization yields $d = 12$ and $\lambda = -13.125$ as the ideal parameters with $\mathrm{MSE} = 0.015$.

Similarly, figure 10 shows the minimization of the MSE in terms of the model complexity and regularization parameter using LASSO regression with 100 bootstrap samples. The optimal MSE this time was given for polynomial order $d = 12$ and regularization term $\lambda = -5.667$, again with $\mathrm{MSE} = 0.015$.

We see that the model complexity and resulting MSE are in accordance with the results using cross-validation. Still, the optimal value for the regularization parameter differ between the two resampling methods. As previously discussed, this might be because there are almost insignificant differences in the MSE across a range of d values and λ values, so there is not a clear minimum to be found. The optimal MSE is given by

$$\mathrm{MSE} = 1.45 \cdot 10^{-2} \pm 0.04 \cdot 10^{-2}.$$

B. Terrain Data

Finally, we apply the previously built knowledge about the discussed methods in artificial data, now to real-world data. We selected a subset of the terrain data which consisted of $n = 25 \times 25 = 625$ points. We scaled the design matrix as well as the response variable. The terrain height was scaled by subtracting the mean and dividing by the standard deviation. We did this to get a more stable algorithm, by having the same scale between the predictor and the response variables.

1. *Ordinary Least Squares*

We fitted polynomial regression models on the terrain data using OLS to find the optimal parameters. We

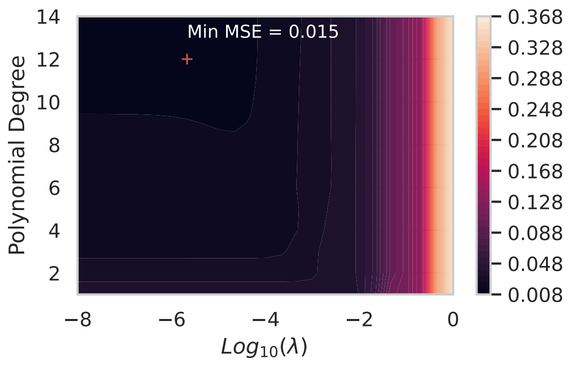

Figure 10. The colored grid shows the MSE as a function of the logarithm of the regularization parameter λ and polynomial degree d. The red cross marks the location of the smallest MSE = 0.015, which has corresponding $d = 12$ and $\log(\lambda) = -5.667$. The MSE was computed from 100 bootstrap samples using LASSO regression. The data set is based on the Franke function with a normal noise term $\sigma = 0.1$ and has $n = 25 \times 25 = 625$ data points.

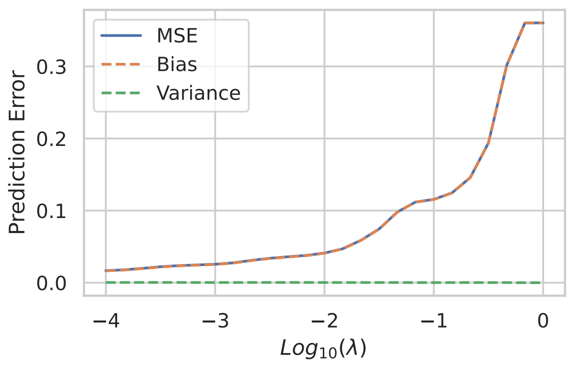

Figure 11. MSE as a function of $\log(\lambda)$, together with the bias and variance. The values have been computed from 100 bootstrap samples, using LASSO regression with polynomial order $d = 12$. The data set is based on values from the Franke function with $\sigma = 0.1$ and $n = 625$.

found the optimal MSE with $d = 11$, where the MSE is estimated to be

$$\text{MSE} = 7.4 \cdot 10^{-2} \pm 2.2 \cdot 10^{-2}.$$

We also looked at the bias-variance trade-off for OLS regression on terrain data. We plotted the MSE, squared bias and variance as functions of polynomial degree. The results can be found in Figure 12. In the figure, we see that the bias term dominates the MSE up to degree 11. Furthermore, the variance term begins to dominate the MSE for degrees greater than 13. Comparing this to the results we got for the Franke data, it is

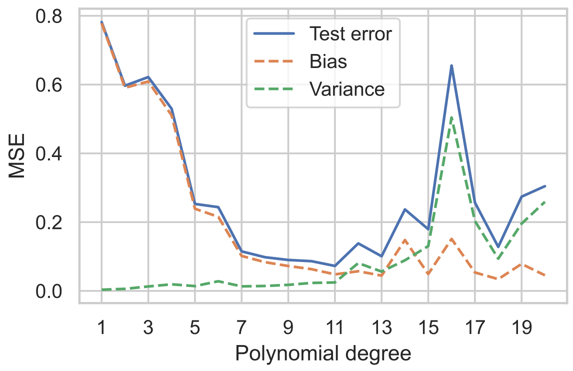

Figure 12. The figure shows the MSE, bias and variance for an OLS regression model of varying polynomial degrees. The values are estimated from 100 bootstrap samples. The model is fitted on terrain data consisting of $n = 625$ data points.

clear that a higher polynomial complexity is needed to get a reasonably accurate fit. This observation will be consistent for the following regression methods as well. This behavior was expected, as it is a consequence of the data being intrinsically more complex, with a higher variance.

We also performed a comparison between the MSE and R^2 plot, against polynomial degree, which can be seen in figure 18 at appendix B

2. *Ridge Regression*

Following the previous procedures, we now fitted polynomial regression models with a Ridge regularization term, finding the optimal parameters with a grid search. The MSE computed from 10-fold cross-validation can be seen in Figure 13. We find that the optimal parameters are $d = 21$ and $\log(\lambda) = -9.3$, which yields an MSE of

$$\text{MSE} = 3.7 \cdot 10^{-2} \pm 1.0 \cdot 10^{-2}.$$

We also looked at the bias-variance trade-off for varying λ with $d = 21$. The values are computed from 100 bootstrap samples. We see that the variance is reduced with larger λ values, and that eventually the λ values are so large that the bias dominates the MSE.

3. *LASSO Regression*

Similarly, we fitted LASSO regression models on the terrain data, again doing a grid search to find the optimal parameters, which can be assessed in figure 15. In

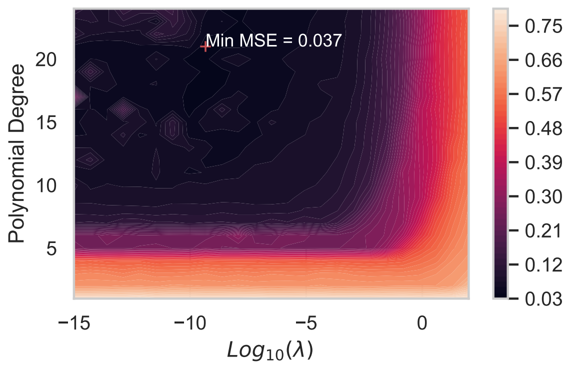

Figure 13. The figure shows a color map indicating the MSE as a function of polynomial degree and $\log(\lambda)$ for Ridge regression on terrain data, with $n = 625$. The MSE is computed using 10-fold cross-validation.

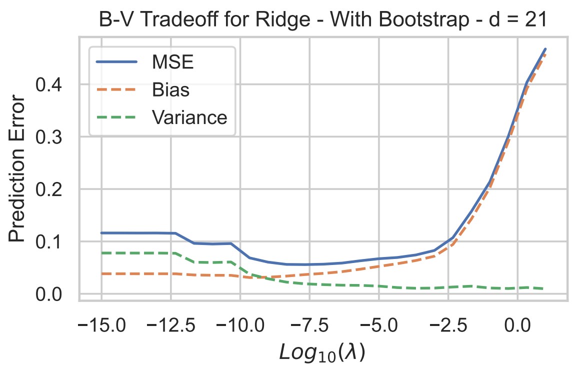

Figure 14. The MSE, bias and variance as a function of $\log(\lambda)$ for Ridge regression on terrain data, using polynomial of degree $d = 21$. The values are computed from 100 bootstrap samples. We have used terain data with $n = 625$ data points.

this case, the optimal parameters are given by $d = 28$ and $\log(\lambda) = -14.3$, where the MSE is

$$\text{MSE} = 2.50 \cdot 10^{-1} \pm 0.29 \cdot 10^{-1}.$$

It is evident that LASSO regression provided, for this case, a much poorer fit than Ridge and OLS regression, while also requiring an unusually large polynomial degree for the parameters optimization.

Finally we analyse in figure 16 the bias-variance tradeoff for the model with the optimized polynomial degree of $d = 28$, varying lambda values. Following what was observed for the analogous plot in Ridge regression, the effect of increasing the values of lambda is that we essentially decrease the model complexity. Therefore the variance of the model tend to decrease the larger the lambda values. On the other hand, the bias term dominates the mean squared error which increases, for that same regime.

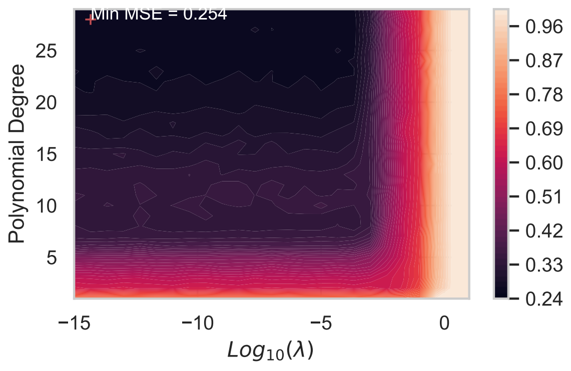

Figure 15. The figure shows a color map indicating the MSE as a function of polynomial degree and $\log(\lambda)$ for LASSO regression on terrain data, with n = 625. The red cross marks the location of the smallest MSE = 0.25, which has corresponding $d = 28$ and $\log(\lambda) = -14.3$. The MSE was computed using 10-fold cross-validation.

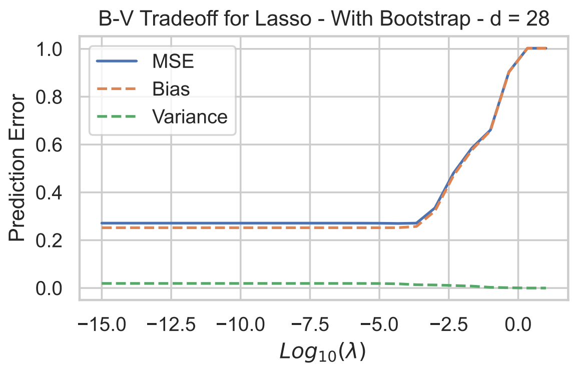

Figure 16. The MSE, bias and variance as a function of $\log(\lambda)$ for LASSO regression on terrain data, using polynomial of degree $d = 28$. The values are computed from 100 bootstrap samples and terrain data with $n = 625$ data points.

4. *Assessing the Models*

Ridge and LASSO regression are useful methods for modelling data where the features are correlated. For a polynomial model of terrain data, the features of the model are multiples of the xy-coordinates. This means that the features should not be highly correlated, and so one might expect there not to be a great advantage to using regularization. Still, we see that Ridge regression performs better than OLS or LASSO for both the

Franke function and the terrain data. This indicates that the advantage of reducing the model variance exceeds the cost of the bias that follows from shrinking the model coefficients.

For the terrain data, we found the optimal degree to be $d = 21$ and regularization to be $\lambda = 10^{-9.3}$, using Ridge regression. This is a model with many parameters (with $p = 253$ features). Considering the fact that we used a data set of $n = 625$ points to train and test, this should be an indication that we are over-fitting, as we effectively only have 2-3 data points per parameter. Training on such a small sample makes it hard for the model to generalize. We do not need as many parameters when modelling the Franke function. This might be due to the fact that the Franke function is smooth, making it easier to approximate with polynomials. The terrain data is not smooth and has greater variance, so more parameters are needed to model the surface.

A general trend observed in the analyses we have done in this report, was that the results heavily depended on the exact sample and sample size of the data set, as well as the stochasticity that follows from randomly sampling the data points. This indicates that the results are heavily data-dependent, particularly in the case of the terrain data. The optimal parameters we found differ somewhat, depending on the resampling method used. This means that the results in this report are probably not very useful when applied to a different region of the terrain, making our analysis highly specialized to the particular region that we have analysed.

Here it is useful to mention a conceptual weak point for the article‘s code implementation: the scaling process for the design matrix takes place before the train-test splitting execution. That means we leak test-data information when building the β predictors for the test data, as both the mean and standard deviations for the features encompasses the whole xy data. The effect, however, should be very small, as the train-test split selection is randomized. On average, there is no preferential information being inserted as bias for the test β calculation. Nevertheless, this fault should be assessed, motivating further experiments.

IV. CONCLUSIONS

In this report, we have explored ordinary least squares, Ridge, and LASSO regression for modelling real terrain data and synthetic data based on the Franke function with added normal noise. We explored the optimal model complexity, as well as the optimal regularization parameter for Ridge and LASSO regression. The expected prediction error for the models was assessed using both cross-validation and bootstrapping.

We have found that Ridge regression offers the lowest prediction errors for both the synthetic data and the terrain data, indicating that minimizing the variance of the model is advantageous. In the case of real terrain data, the optimal model required high complexity to minimize the MSE, despite being limited to a relatively small sample size for training. With a highly complex model for the terrain data, we are risking over-fitting. That is, the model is highly specialised, and so our results are not expected to generalize well to new samples. When modelling the Franke function, this was not the case, which might be due to the fact that it is a smooth, less variable surface. Considering the low complexity and that the synthetic data is generated from the same underlying data distribution, we expect better generalization to new samples in that case.

[1] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed., Springer Series in Statistics (Springer New York Inc., 2001).

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016) `http://www.deeplearningbook.org`.

[3] G. Bonaccorso, *Machine Learning Algorithms: Popular algorithms for data science and machine learning* (Packt Publishing Ltd, 2018).

Appendix A

1. Variance of Dependent Variable

We find the variance of the dependent variable z_i as

$$\begin{aligned}
\mathrm{Var}(z_i) &= \mathbb{E}\left\{(z_i - \mathbb{E}(z_i))^2\right\} \\
&= \mathbb{E}(z_i^2) - \mathbb{E}(z_i)^2 \\
&= \mathbb{E}\{(\mathbf{X}_{i,*}\boldsymbol{\beta} + \epsilon_i)^2\} - (\mathbf{X}_{i,*}\boldsymbol{\beta})^2 \\
&= \mathbb{E}\{(\mathbf{X}_{i,*}\boldsymbol{\beta})^2 + 2\epsilon_i\mathbf{X}_{i,*}\boldsymbol{\beta} + \epsilon_i^2\} - (X_{i,*}\boldsymbol{\beta})^2 \\
&= \mathbb{E}(\epsilon_i^2) \\
&= \mathrm{Var}(\epsilon_i) = \sigma^2.
\end{aligned}$$

2. Variance of OLS Estimator

Here, we compute the variance of the OLS estimator. First, note that

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{z} \\
&= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}) \\
&= \boldsymbol{\beta} + (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{\epsilon}.
\end{aligned} \tag{A1}$$

Then we find the variance of the OLS estimator $\hat{\boldsymbol{\beta}}$ as

$$\begin{aligned}
\mathrm{Var}(\hat{\boldsymbol{\beta}}) &= \mathbb{E}\left\{[\hat{\boldsymbol{\beta}} - \mathbb{E}(\hat{\boldsymbol{\beta}})][\hat{\boldsymbol{\beta}} - \mathbb{E}(\hat{\boldsymbol{\beta}})]^T\right\} \\
&= \mathbb{E}\left\{(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}))(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})^T\right\} \qquad \text{(A2)} \\
&= \mathbb{E}\left\{[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{\epsilon}][(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{\epsilon}]^T\right\} \\
&= \mathbb{E}\left\{(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\right\} \\
&= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbb{E}(\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T)\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} \\
&= \sigma^2(\mathbf{X}^T\mathbf{X})^{-1},
\end{aligned}$$

where we have inserted (A1) in (A2).

3. Bias-Variance Trade-Off Decomposition

We assume that the dependent variable z_i can be fully described by a function f and a noise term ϵ_i, where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, such that $z_i = f(x_i, y_i) + \epsilon_i$. We approximate the true values as $\hat{f}(x_i, y_i) = \hat{z}_i$ on given a set of training data. Let f_i be shorthand for $f(x_i, y_i)$. Likewise, let $\hat{z}_i$ be the prediction value of our model. The expected prediction error is

$$\begin{aligned}
\mathbb{E}[(z_i - \hat{z}_i)^2] &= \mathbb{E}[(f_i + \epsilon_i - \hat{z}_i)^2] \\
&= \mathbb{E}[(f_i + \epsilon_i - \hat{z}_i + \mathbb{E}(\hat{z}_i) - \mathbb{E}(\hat{z}_i))^2] \\
&= \mathbb{E}[(f_i - \mathbb{E}(\hat{z}_i) + \epsilon_i + \mathbb{E}(\hat{z}_i) - \hat{z}_i)^2] \\
&= \mathbb{E}[(f_i - \mathbb{E}(\hat{z}_i))^2 + \epsilon_i^2 + (\mathbb{E}(\hat{z}_i) - \hat{z}_i)^2 \\
&\quad + 2(f_i - \mathbb{E}(\hat{z}_i))\epsilon_i + 2\epsilon_i(\mathbb{E}(\hat{z}_i) - \hat{z}_i) \\
&\quad\ + 2(f_i - \mathbb{E}(\hat{z}_i))(\mathbb{E}(\hat{z}_i) - \hat{z}_i)] \\
&= \mathbb{E}[(f_i - \mathbb{E}(\hat{z}_i))^2] + \mathbb{E}[\epsilon_i^2] + \mathbb{E}[(\mathbb{E}(\hat{z}_i) - \hat{z}_i)^2] \\
&\quad + 2\mathbb{E}[(f_i - \mathbb{E}(\hat{z}_i))\epsilon_i] + 2\mathbb{E}[\epsilon_i(\mathbb{E}(\hat{z}_i) - \hat{z}_i)] \\
&\quad\ + 2\mathbb{E}[(f_i - \mathbb{E}(\hat{z}_i))(\mathbb{E}(\hat{z}_i) - \hat{z}_i)] \\
&= \mathbb{E}[(f_i - \mathbb{E}(\hat{z}_i))^2] + \sigma^2 + \mathbb{E}[(\mathbb{E}(\hat{z}_i) - \hat{z}_i)^2].
\end{aligned}$$

In the final step, we use the fact that $\mathbb{E}(\epsilon_i) = 0$ and that $\hat{z}_i$ and ϵ_i are independent variables. Also, we note that

$$\begin{aligned}
\mathbb{E}[(f_i - \mathbb{E}(\hat{z}_i))(\mathbb{E}(\hat{z}_i) - \hat{z}_i)] &= \mathbb{E}[f_i\mathbb{E}(\hat{z}_i) - f_i\hat{z}_i - \mathbb{E}(\hat{z}_i)^2 \\
&\quad + \mathbb{E}(\hat{z}_i)\hat{z}_i] \\
&= f_i\mathbb{E}(\hat{z}_i) - f_i\mathbb{E}(\hat{z}_i) \\
&\quad - \mathbb{E}(\hat{z}_i)^2 + \mathbb{E}(\hat{z}_i)^2 \\
&= 0,
\end{aligned}$$

which follows from the fact that f is deterministic. Finally, to get the bias term, we have

$$\begin{aligned}
\mathbb{E}[(f_i - \mathbb{E}(\hat{z}_i))^2] &= \mathbb{E}[f_i^2 - 2f_i\mathbb{E}(\hat{z}_i) + \mathbb{E}(\hat{z}_i)^2] \\
&= f_i^2 - 2f_i\mathbb{E}(\hat{z}_i) + \mathbb{E}(\hat{z}_i)^2 \\
&= (f_i - \mathbb{E}(\hat{z}_i))^2,
\end{aligned}$$

where we again use the fact that f is deterministic.

Appendix B

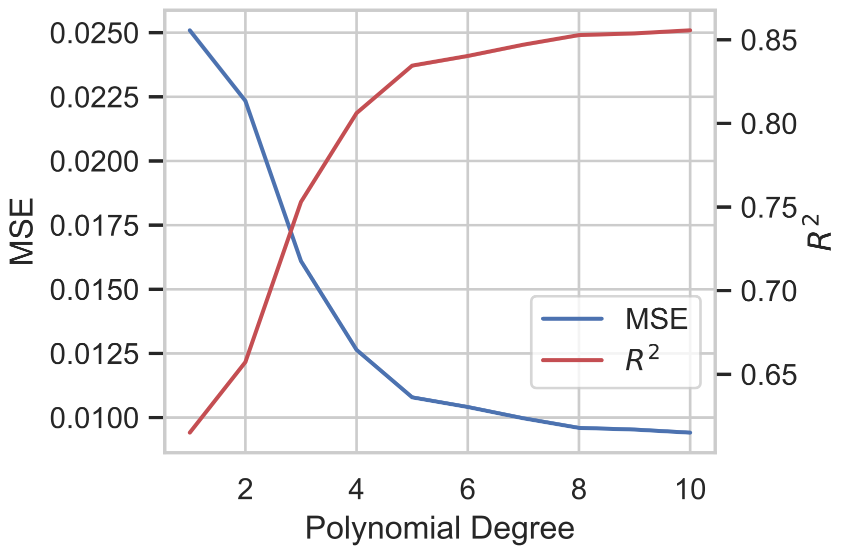

Figure 17. MSE and R^2 test metrics for Franke function and OLS fit, using $B = 100$ bootstrap samples and $n = 25 \times 25$ data points. The R^2 indicates the amount of variance that can be explained through predictor variables. The closer R^2 is to 1, the better.

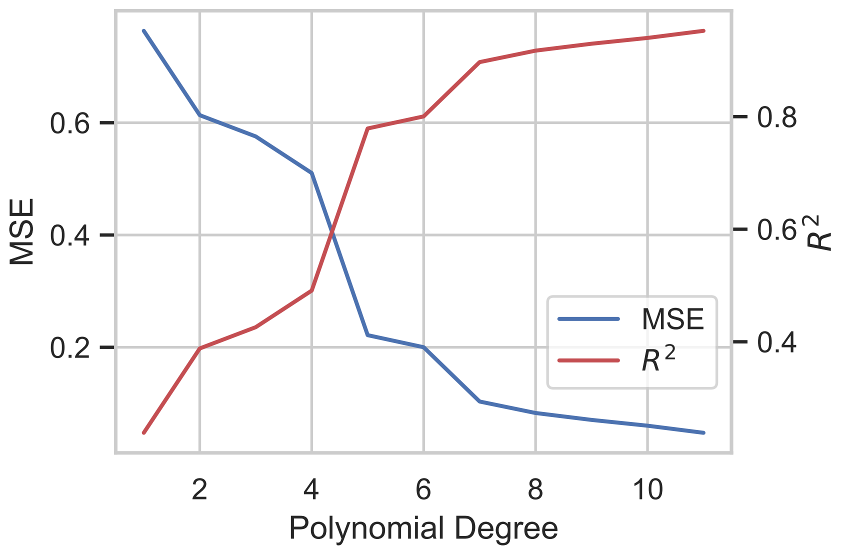

Figure 18. MSE and R2 test metrics for terrain data given an OLS fit, using $B = 100$ bootstrap samples and $n = 25 \times 25$ data points.

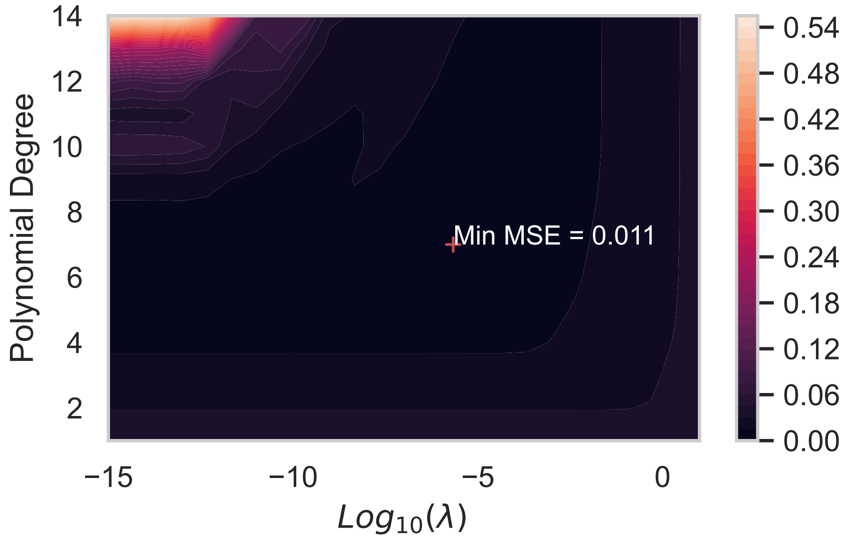

Figure 19. The colormap shows the MSE as a function of both polynomial degree and regularization term λ. We computed the MSE using $B = 100$ bootstrap samples. This plot is based on a Ridge regression model for the Franke function.

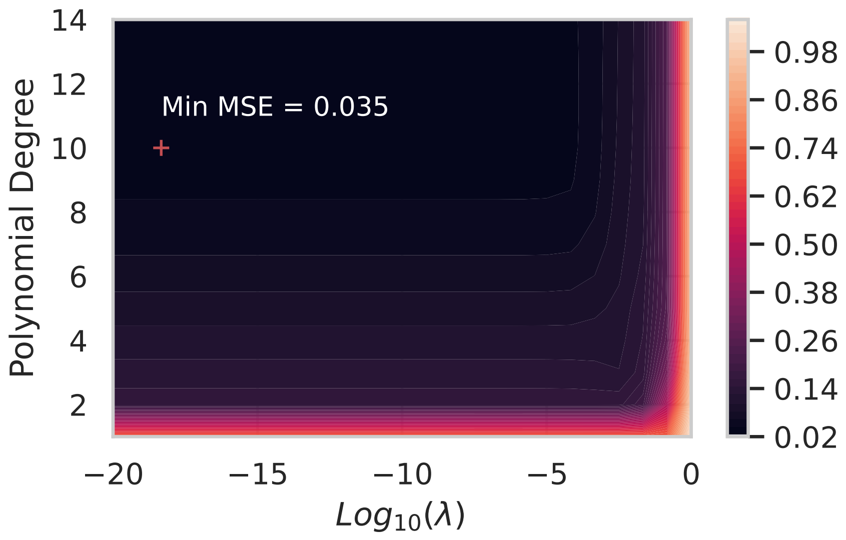

Figure 20. The colored grid shows the MSE as a function of the logarithm of the regularization parameter λ and polynomial degree d. The red cross marks the location of the smallest MSE = 0.035, which has corresponding $d = 10$ and $\log(\lambda) = -18.33$. The MSE was computed from 100 bootstrap samples using LASSO regression. The data set is based on the Terrain data and has $n = 10 \times 10 = 100$ data points.