Regression analysis and resampling methods

FYS-STK4155 – Project 1

Stjepan Salatovic
github.com/stipesal/FYS-STK4155

October 10, 2021

Abstract

In order to parametrize data in two dimensions we utilize linear regression. The models investigated include ordinary least squares (OLS), as well as the penalty methods, Ridge and Lasso regression. More specifically, we use a polynomial regression model up to degree $p = 20$. Using the bootstrap resampling technique, we are able to confirm theoretical statements about the bias-variance tradeoff. Our data consists of sampling the so-called Franke's function, as well as of terrain data in Norway. The results suggest that a relatively low polynomial degree of $p = 5$ is sufficient to fit Franke's function. On the other hand, we find that for the highly irregular terrain data, high polynomial degrees $p \geq 10$ and a penalty method are needed to obtain good results. If a regularisation parameter is then chosen in a refined way, the terrain can be represented well, at least qualitatively.

Contents

1 Linear regression

In general, a regression analysis is concerned with describing the properties of a so-called target variable y as a function of regressors (or predictors) $x_1, \dots, x_k$. Hereby, it is assumed that this functional relationship is not observed exactly but is influenced by a random error ε, such that a model is often given by

$$y = f(x_1, \dots, x_k; \beta) + \varepsilon,$$

where f is the so-called regression function and β the unknown vector of parameters. Usually, one demands centred and homoscedastic random errors, that is $\mathbb{E}[\varepsilon] = 0$ and $\mathrm{Cov}[\varepsilon] = \sigma^2 I$, where I denotes the identity matrix.[1] Statistically, we are then interested in estimating β from given data $(y_i, x_{i1}, \dots, x_{ik})$ for $i = 1, \dots, n$. The best known is the class of linear models, where

$$f(x_1, \dots, x_k; \beta) = \beta_0 + \beta_1 x_1 + \dots + \beta_k x_k$$

is a linear function of the unknown parameter vector $\beta \in \mathbb{R}^{k+1}$. The parameter β_0 is often referred to as the intercept. It should be noted that in linear models the regression function must be a linear function of the parameter vector β, but not of the regressors $x_1, \dots, x_k$. Thus, the polynomial regression with

$$f(x_1, \dots, x_k; \beta) = \sum_{j=0}^{k} \beta_j x_i^j$$

is also a linear model. Given data $(y_i, x_{i1}, \dots, x_{ik})$ for $i = 1, \dots, n$, we can utilize matrix notation to state the linear model more compact as

$$y = X\beta + \varepsilon,$$

where

$$y = (y_1, \dots, y_n)^\top \in \mathbb{R}^n, \quad \varepsilon = (\varepsilon_1, \dots, \varepsilon_n)^\top \in \mathbb{R}^n,$$

and

$$\beta = (\beta_0, \dots, \beta_k) \in \mathbb{R}^{k+1}.$$

Further, we denote by

$$X = \begin{pmatrix} 1 & x_{11} & \dots & x_{1k} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \dots & x_{nk} \end{pmatrix} \in \mathbb{R}^{n \times (k+1)}$$

the so-called design matrix, which represents the actual features based on the regressors and thus, contributes significantly to the model construction (or design). We note the ones in the first column of the design matrix X enlivening the intercept β_0. In Section 3 we will construct the design matrix needed for a polynomial regression model in two dimensions.

Suppose an estimation of the unknown parameter vector β was made and denote this estimation by $\hat{\beta}$. Then, we can make predictions

$$\hat{y} = X\hat{\beta}$$

and measure the error

$$\hat{\varepsilon} = y - \hat{y}$$

between the responses and the predictions. The vector $\hat{\varepsilon} = (\hat{\varepsilon}_1, \dots, \hat{\varepsilon}_n)^\top \in \mathbb{R}^n$ is called the vector of residuals, which are regarded as empirical counterparts of the errors.[2] The main idea of performing a regression analysis is to minimize precisely the residuals $\hat{\varepsilon}$, the deviation between the predictions and the (true) responses. The choice of how to measure this particular distance (choosing a metric) plays a major role, it essentially determines the model. The functional expression which is minimized eventually, is called the loss (or cost) function.

In the following three subsections we give rise to the linear models we used extensively by stating the different loss functions and characteristics.

1.1 Ordinary least squares (OLS)

The best known loss function in regression analysis is the mean squared error (MSE)

$$L(\beta) = \frac{1}{n} \|\hat{\varepsilon}\|_2 = \frac{1}{n} \sqrt{\hat{\varepsilon}^\top \hat{\varepsilon}},$$

which utilizes the L2 (or Euclidean) norm and averages the squared residuals. Letting $p = k+1$, the minimization problem at hand is given by

$$\beta^* = \underset{\beta \in \mathbb{R}^p}{\arg\min}\, L(\beta)$$

[1] In the so-called classical linear model, one further demands $\varepsilon \sim N(0, \sigma^2 I)$.

[2] One can easily show that $\mathbb{E}[\hat{\varepsilon}] = 0$, if the estimator $\hat{\beta}$ is unbiased.

which can easily be solved analytically. Expanding

$$\begin{aligned} L(\beta) &= \|y - X\beta\|_2^2 \\ &= (y - X\beta)^\top (y - X\beta) \\ &= y^\top y - 2\beta^\top X^\top y + \beta^\top X^\top X\beta, \end{aligned}$$

differentiating with respect to β, and demanding equality yields the so-called normal equation

$$X^\top X\beta = X^\top y.$$

We note that omitting the constant factor $1/n$ and squaring the loss function $L(\beta)$ does not change the minimum. Finally, if $(X^\top X)^{-1}$ exists, we solve for the ordinary least squares (OLS) estimator

$$\beta_{OLS} = (X^\top X)^{-1} X^\top y.$$

We note that the minimization problem has a unique solution if $X^\top X$ is regular, which is the case when the $p = k + 1$ columns of X are linearly independent. In practice however, rather than inverting $X^\top X$, the normal equation is solved iteratively using a singular value decomposition or a LU decomposition due to numerical stability.

Since estimators are random variables in general, it is often insightful to compare them by studying their moments. In case of the OLS estimator, we have an unbiased estimator for β, that is, the expected value of β_{OLS} coincides with β itself. We can see this by calculating

$$\begin{aligned} \mathbb{E}[\beta_{OLS}] &= (X^\top X)^{-1} X^\top \mathbb{E}[y] \\ &= (X^\top X)^{-1} X^\top X\beta \\ &= \beta, \end{aligned}$$

since the error ε is centred and the design matrix X is purely deterministic in this paper.[3] Furthermore, one can show that the covariance matrix of the OLS estimator is given by

$$\mathrm{Cov}[\beta_{OLS}] = \sigma^2 (X^\top X)^{-1}.$$

Another unbiased estimator can be given for the unknown variance σ^2, namely by

$$\hat{\sigma}^2 = \frac{1}{n-p} \|\hat{\varepsilon}\|_2^2,$$

that is, the residual sum of squares divided by a specific statistical degree of freedom needed for unbiasedness. Since $\mathbb{V}[\beta_i] = \sigma^2 (X^\top X)^{-1}_{i+1,i+1}$, the standard deviation (or standard error) of the parameter β_i can be estimated by

$$\mathrm{se}(\beta_i) = \hat{\sigma} \sqrt{(X^\top X)^{-1}_{i+1,i+1}},$$

which can in turn be used for constructing confidence intervals for β_i. If a classical linear model is present, that is, if the errors ε are normally distributed, then a confidence interval for β_i at the $(1 - \alpha)$ confidence level is given by

$$[\hat{\beta}_i \mp t_{n-p;\, 1-\alpha/2} \cdot se(\beta_i)],$$

where $t_{n-p;\, 1-\alpha/2}$ denotes the $1 - \alpha/2$ quantile of Student's t-distribution with $n - p$ degrees of freedom. In case the number of observations n is sufficiently large, one can, due to the central limit theorem, at least speak of an asymptotic confidence interval for a non-classical model.

Further advantages of the OLS estimator include the *best linear unbiased estimator* (BLUE) property, as well as the availability of an explicit formula.

1.2 Ridge regression

In contrast to unbiased OLS estimation, it can sometimes be useful to artificially add a bias to the model. For example, adding a so-called penalty term to the loss function prevents the parameter from deviating too far from a certain value (e.g. zero) and pushes (or favors) parameters in a certain direction (or area). An illustrative motivation for this is the idea that only a few regressors actually have a measurable influence on the response, resulting in a simpler and more interpretable model.[4]

One of the most popular penalty methods is the Ridge regression, originally introduced as the so-called Tikhonov regularization in the field of inverse problems by Andrey Tikhonov in 1943 [4]. The general idea is to impose more regularization to a ill-posed problem, e.g. by increasing the condition of a matrix.

In our case, adding the L2 norm of the parameter vector β as a penalty term to the MSE function,

[3] Sometimes the regressor quantities are random themselves, resulting in a so-called stochastic design matrix X.

[4] This is also the reason why penalty models are used to determine feature importance.

yields the loss function

$$L(\beta) = \|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2,$$

which gives rise to the Ridge regression model. Hereby, the regularization parameter $\lambda > 0$ controls the strength of the penalty. The larger the penalty term λ, the smaller the parameter coefficients β_i of a solution must be. Preference is given by this method to parameters β for which a large number of coefficients β_i is close to zero. We will see that λ changes the induced bias of the method. Due to the simple structure and smoothness of the additional penalty L2 term, an explicit expression for the minimization problem

$$\beta^* = \underset{\beta \in \mathbb{R}^p}{\arg\min}\, L(\beta)$$

is still available. Calculating

$$\begin{aligned} L(\beta) &= \|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2 \\ &= (y - X\beta)^\top (y - X\beta) + \lambda\beta^\top\beta \\ &= y^\top y - 2\beta^\top X^\top y + \beta^\top X^\top X\beta + \lambda\beta^\top\beta, \end{aligned}$$

differentiating with respect to β, and demanding equality yields the closed form expression

$$\beta_R = (X^\top X + \lambda I)^{-1} X^\top y.$$

In contrast to OLS estimation, there always exists a unique ridge estimator β_R for $\lambda > 0$. Since $X^\top X$ is positive semi-definite, adding λI makes it always invertible. This is also the reason why ridge regression is often used when dealing with highly correlated regressors where OLS estimates would cause near-singular matrix problems.

We note the relationship $\beta_{OLS} = \beta_R$ for $\lambda = 0$.

1.3 Lasso regression

Another widely used penalty method is the so-called Lasso[5] regression method which was firstly introduced by Santosa & Symes in 1986. Similar to Ridge regression, the idea is to add a penalty term for the parameters to the MSE loss function defined previously. The only difference is the metric (or norm) used to measure the size of the parameter vector β. Instead of using the L2 (or Euclidean) norm, Lasso relies on the L1 norm, which results in the loss function given by

$$L(\beta) = \|y - X\beta\|_2^2 + \lambda\|\beta\|_1,$$

where $\lambda > 0$ is again the regularization parameter. This change, small at first glance, makes a significant difference in model behaviour. As with the ridge estimator, our assumption is still that only a few parameters β_i are actually non-zero. Changing the penalty term provides solutions that more closely match this assumption.

To understand the really essential difference between the Ridge and Lasso estimators, it is useful to visualise the parameter space of a simple regression problem for $p = 2$. But firstly, we want to point out the possibility of reformulating the minimization problem into one with constraints. Without going to much into detail, one can determine constraints of the form $\|\beta\|_i \leq t$ for $i = 1, 2$, where t is given by a data-dependent relationship with the regularization parameter λ. Appending the constraint into the objective function itself, as introduced for the Ridge and Lasso method here, is a well-known practice in optimization theory called the Lagrangian approach. Figure 1 and 2 show the parameter space of a regression problem with two parameter coefficients $\beta = (\beta_1, \beta_2)^\top \in \mathbb{R}^2$, respectively.[6] Both Figures show the contours of the convex MSE loss function and the unique minimum, the OLS estimator $\beta_{OLS} = \hat{\beta}$. The admissible parameters given by the penalty in the Ridge and Lasso method are shown as a blue area for Ridge in Figure 1 and for Lasso in Figure 2.

One question that one may and must ask is where in the graph one can find the penalty estimator, Ridge β_R and Lasso β_L, respectively. The answer is where the red contour lines of the MSE function first touch the blue constraint area. One can see that this intersection will most likely be a corner of the set of admissible parameters where coefficients of β_L are exactly equal to zero.

Thus, unlike Ridge, Lasso favours coefficients that are equal to and not close to zero making it even more suitable for performing feature importance and developing sparse, interpretable models.

The calculation of the Lasso estimator β_L turns out to be more difficult than with the OLS or Ridge

[5] Lasso stands for *least absolute shrinkage and selection operator*.

[6] Hastie, Tibshirani, and Friedman [2], Figure 3.11.

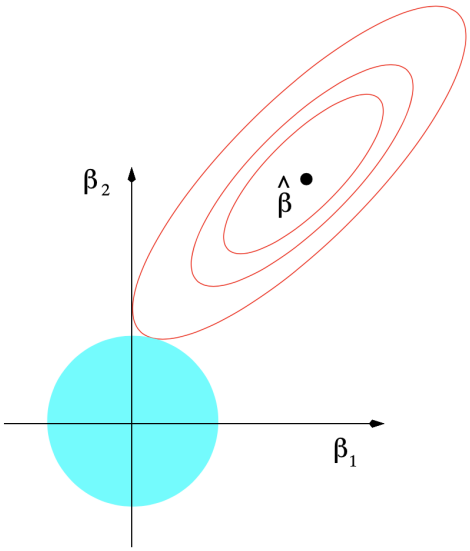

Figure 1. Contours of the MSE loss (red ellipses) and the OLS estimator $\beta_{OLS} = \hat{\beta}$. The Ridge estimator β_R is given by the intersection of the contours and the set of admissible parameters (blue).

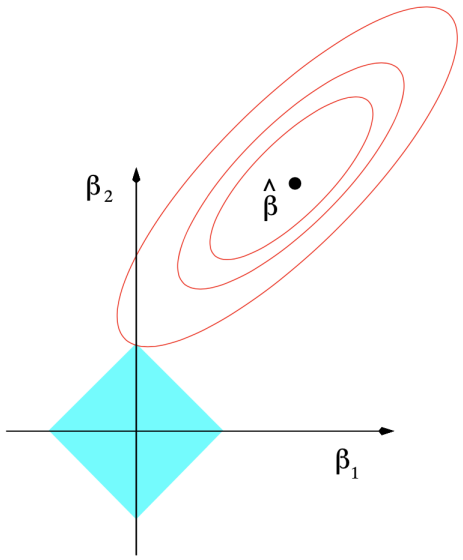

Figure 2. Contours of the MSE loss (red ellipses) and the OLS estimator $\beta_{OLS} = \hat{\beta}$. The Lasso estimator β_L is given by the intersection of the contours and the set of admissible parameters (blue).

method. In contrast to the OLS and Ridge estimator there are no closed-form solutions available in general for Lasso. However, it is possible to construct simple special cases that are not relevant for practice, for which one can indeed give explicit solutions, but which we do not want to go into here. The use of numerical iterative algorithms to determine the lasso estimator is therefore unavoidable. Since the L1 norm is not differentiable everywhere, one must resort to subgradient methods which are generalizations of gradient descent methods if the objective function is not differentiable at every point. Another widely used algorithm in context of determining the Lasso estimator β_L is the coordinate-wise descent, the derivation of which, however, is beyond the scope of this paper.

2 Model selection & assessment

In regression analysis and machine learning overall, it is extremely important to find out how well a model is performing on the training data but also on unseen, validation (or test) data. Evaluating the performance of a model on unseen data is known as *model assessment*, while selecting the complexity (or degrees of freedom) of a model is known as *model selection*. Often the loss function used for training, is taken as the decisive criterion for model selection (and assessment), e.g. the model with the smallest MSE on the training data set is chosen. However, this is by no means an indication of the performance when applying the model on unseen test data. Since test data is usually not available either, an expected prediction error has to be estimated in a sophisticated way.

In the following subsections we will derive the decomposition of the mean squared error, as well as introduce two important resampling techniques for estimating expectations on test data sets in order to perform a model assessment.

2.1 Bias-variance tradeoff

In order to decompose the MSE into the two main sources of error we use the classical model assumption $y = f + \varepsilon$ with normal $\varepsilon \sim N(0, \sigma^2)$, such

that

$$\begin{aligned}\mathbb{E}(y-\hat{y})^2 &= \mathbb{E}(f+\varepsilon-\hat{y})^2 \\ &= \mathbb{E}(f-\hat{y})^2 + 2\mathbb{E}((f-\hat{y})\varepsilon) + \mathbb{E}(\varepsilon^2) \\ &= \mathbb{E}(f-\hat{y})^2 + \sigma^2.\end{aligned}$$

Further, by adding zero in the expectation of the last equation we have that

$$\begin{aligned}\mathbb{E}(f-\hat{y})^2 &= \mathbb{E}(f-\mathbb{E}(\hat{y})+\mathbb{E}(\hat{y})-\hat{y})^2 \\ &= (f-\mathbb{E}(\hat{y}))^2 + \mathbb{E}(\hat{y}-\mathbb{E}(\hat{y})) \\ &= \mathrm{Bias}(\hat{y})^2 + \mathrm{Var}(\hat{y}).\end{aligned}$$

Putting it together, the expected MSE can be decomposed into the terms

$$\mathbb{E}(y-\hat{y})^2 = \mathrm{Bias}(\hat{y})^2 + \mathrm{Var}(\hat{y}) + \sigma^2,$$

where σ^2 is the irreducible error serving as a lower bound for the MSE. Finally, if we substitute the expectation with the sample mean over the whole data set we get

$$\begin{aligned}\mathbb{E}(y-\hat{y})^2 = &\frac{1}{n}\sum_i (f_i - \mathbb{E}(\hat{y}))^2 \\ &+ \frac{1}{n}\sum_i (\hat{y}_i - \mathbb{E}(\hat{y}))^2 \\ &+ \sigma^2.\end{aligned}$$

The bias describes the average expected systematic error of the algorithm. A systematic error occurs when the algorithm favours certain solutions by introducing additional constraints. This can happen by penalising undesirable properties or by restricting the search to a smaller class of functions, e.g. fitting a linear model to a quadratic data set.

The variance on the other hand measures how strongly the algorithm depends on the training data, i.e. how strongly a change in individual training data points affects the decision rule induced by the algorithm.

The problem is that a small bias is often associated with high variance and large bias with small variance. The problem of simultaneously minimising both sources of error, bias and variance is referred to as bias–variance trade off.

For example in linear regression, β_{OLS} is the estimator with the lowest variance among all linear estimators without bias (BLUE). Therefore one cannot hope to find unbiased estimators with lower MSE. However, motivated by the bias-variance decomposition of the MSE, there exist algorithms which trade a non-zero bias for a saving in variance and thus show a lower MSE. In practical terms, forcing a bias means that we specify constraints in the form of penalty terms when constructing the loss function (see Section 1.2 and 1.3).

2.2 Resampling techniques

In order to estimate certain statistical quantities resampling techniques are often used. The idea is to draw a sufficiently large number of samples from the same distribution (same dataset) and repeat certain fitting or training procedures. Note that resampling techniques can lead to high computational overhead if training of a single model is costly. Nevertheless, the use of resampling techniques is often unavoidable, especially when closed form solutions are not available for the statistics of interest.

2.2.1 Bootstrapping

In statistics, bootstrapping means to take a subsample from a dataset repeatedly and to store a corresponding quantity of interest. The true value can then be estimated by averaging over all calculated estimates. If the sample size (and the original data set) is sufficiently large, a sufficiently well approximation of the true population can be achieved. To estimate the bias, variance and MSE of our models we used already implemented resampling functions with replacement.[7]

2.2.2 Cross-validation

Cross-validation is used to assess a model and to investigate statistical results on test data sets. In the k-fold cross-validation, the whole data set is splitted randomly into k subsets. One of the k subsamples is then used as validation data, while the other $k-1$ subsets serve to train the model. This procedure is repeated exactly k times such that every subset (and every data instance) is used as a test data set exactly once. This way the performance of generalizing on independent test data sets can be assessed.

[7] To be more precise we used Sklearn's *resample*.

3 Data & Results

Before we can turn to the actual data and the associated results, we need to briefly discuss our construction of the design matrix.

Multivariate polynomials

Since we will parametrize two-dimensional data, we will briefly introduce multivariate polynomials. A polynomial $p : \mathbb{R}^n \to \mathbb{R}$ of degree $d \in \mathbb{N}$ is given by

$$p(x_1, \dots, x_n) = \sum_{i \in \mathbb{N}_0^n \,:\, \|i\|_1 \leq d} \beta_i x_1^{i_1} \dots x_n^{i_n},$$

where $i = (i_1, \dots, i_n) \in \mathbb{N}_0^n$ is a multi-index and $\beta_i \in \mathbb{R}$ are the coefficients. In this report we investigate so-called inhomogeneous polynomials, that is monomials are allowed to have a total degree of less then d. The total degree of a monomial is defined as the sum of all exponents. For example, if $n = d = 2$, then

$$p(x_1, x_2) = x_1^2 + x_1 x_2 + x_2^2$$

represents a homogeneous polynomial, because every monomial (term) has a total degree of exactly d, while

$$p(x_1, x_2) = x_1^2 + x_1 x_2 + x_2^2 + x_1 + x_2 + 1$$

is an inhomogeneous polynomial. In particular, note that the number of monomials for an inhomogeneous polynomial is much larger than for a homogeneous polynomial, such that the complexity of the model grows rapidly. In fact, one can easily check that the number of monomials for an homogeneous polynomials of degree d with $n = 2$ variables is $d + 1$, they are given by

$$x_1^i x_2^{d-i} \quad \text{for } i = 0, \dots, d.$$

If we include monomials with total degree smaller than d, we conclude

$$\sum_{k=0}^{d} k + 1 = \frac{1}{2}(d+1)(d+2).$$

Figure 3 compares the number of monomials as a function of the polynomial degree d for $n = 2$ variables

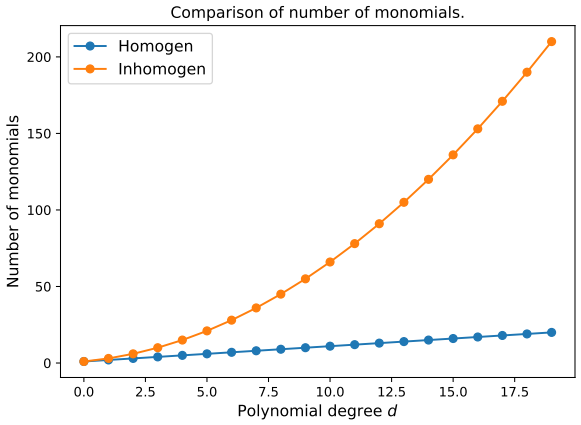

Figure 3. Number of monomials as a function of the polynomial degree $d \in \mathbb{N}$ for $n = 2$ for an homogeneous and inhomogeneous polynomial.

In this report $n = 2$ will be relevant, such that we investigate solely polynomials of the form

$$p(x_1, x_2) = \sum_{i \in \mathbb{N}_0^2 \,:\, i_1 + i_2 \leq d} \beta_i x_1^{i_1} x_2^{i_2}.$$

for different polynomial degrees $d \in \mathbb{N}$.

Our design matrix

In a polynomial regression model, the number of monomials is essentially the number of features, or equivalently, the number of columns in the design matrix X.[8]

Thus given data $x_i \in \mathbb{R}^2$ for $i = 1, \dots, n$ and a (inhomogeneous) polynomial regression model of order d, the design matrix has the size $X \in \mathbb{R}^{n \times p}$ where

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

is the number of monomial (or features), as shown in the previous Subsection. For example, if $d = 2$ then the design matrix $X \in \mathbb{R}^{n \times p}$ with $p = 6$ would be given by

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} & x_{11}x_{12} & x_{11}^2 & x_{12}^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} & x_{n1}x_{n2} & x_{n1}^2 & x_{n2}^2 \end{pmatrix}.$$

[8] This was implemented for verification purposes of the design matrix.

3.1 Franke's function

We will fit Franke's function, which was used in 1979 as a test function for interpolation problems [1]. It is defined as $f : \mathbb{R}^2 \to \mathbb{R}$ with

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

Figure 4 shows Franke's function on the unit square, which is also the domain we will try fitting a model on.

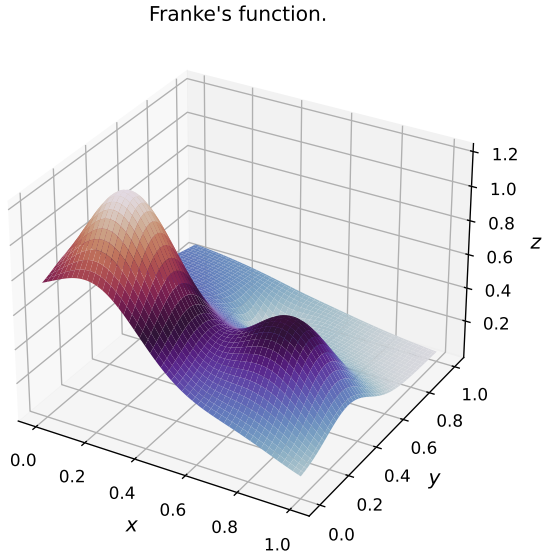

Figure 4. Franke's function on the unit square, that is $x, y \in [0, 1]$. The colors indicate different values in height.

To make it a little more interesting, of course, we will not use the regular grid as in Figure 4 representing the data set, but rather sample $N \in \mathbb{N}$ uniformly distributed points on the unit square and add white noise ε.[9] Figure 5 shows Franke's function again, together with $N = 200$ sampled train and test data points generated by adding noise with variance $\sigma^2 = 0.1$.

[9] The effect of adding white noise to the data on the model performance is shown in Section 3.1, Figure 12.

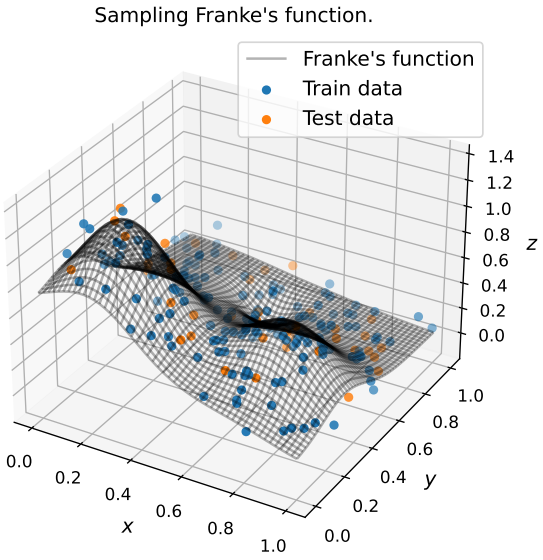

Figure 5. Franke's function on the unit square, as well as $N = 200$ train and test data points, generated by adding a noise level of $\sigma^2 = 0.1$. A train-test ratio of 80-20 was used.

In order to fit Franke's function we sampled $N = 200$ random and uniform data points in the unit square $[0, 1]^2$. Since the data or in particular x, y, is already contained in the interval $[0, 1]$ (and therefore also all monomial features), we did not further scale the data. This was also checked and it turned out that scaling is not necessary for Franke's function. However, scaling was established and useful for the results on the terrain data in Section 3.2.

OLS estimation for terrain data

Firstly, we want to investigate the regular train and test MSE and R2 curves as a function of model complexity. Figure 6 shows the train and test MSE and R2 for different polynomial regression models fitted using the classical OLS estimation. In total, $N = 400$ datapoints were used, a white noise $\varepsilon \sim N(0, 0.1^2)$ was added, and a train-test ratio of 80-20 was used. One can clearly see the training MSE decreasing (R2 increasing) with a higher model complexity, whereas the testing MSE increases (R2 decreases) from a specific model com-

plexity. The minimum in testing MSE (also testing R2) is achieved for a polynomial regression model of order $d = 5$. Overfitting is happening for $d > 5$, and underfitting for $d < 5$. This observation is further illuminated and explained using bias-variance decomposition later.

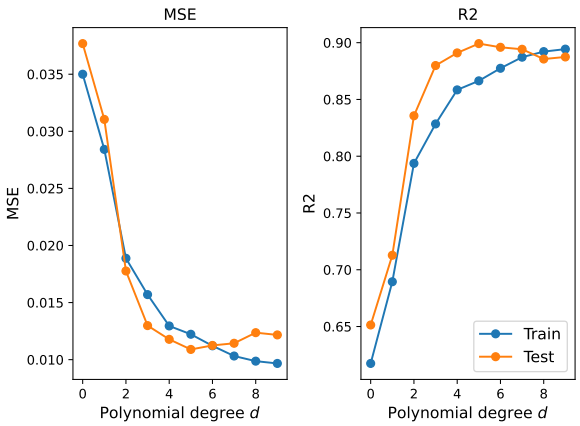

Figure 6. Train (blue) and test (orange) MSE (left) and R2 (right) for different polynomial models with order $d \in \mathbb{N}$.

In order to see the best model in action, Figure 7 shows the 2D image of Franke's function, as well as the prediction made by the polynomial model of order $d = 5$ on an uniform grid with 10^4 grid points. In addition, Figure 8 shows a 3D prediction of the test data together with Franke's function for the same polynomial model with degree $d = 5$.

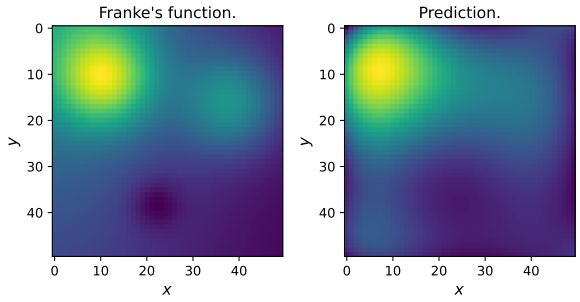

Figure 7. 2D Franke's function (left) and the prediction made by a polynomial model of order $d = 5$ (right).

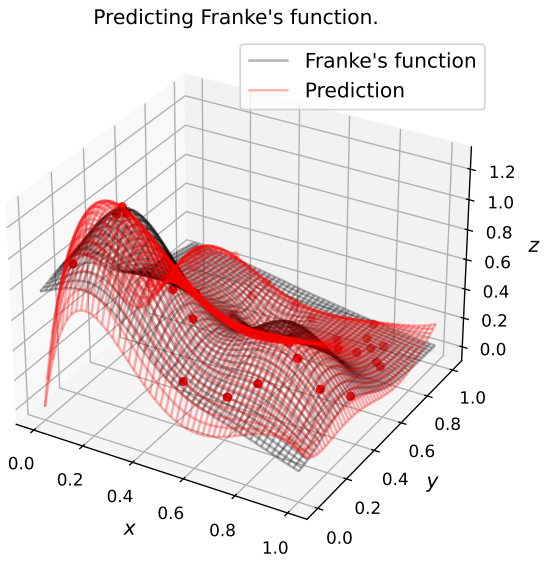

Figure 8. Shown are Franke's function (grey grid), 3D prediction (red grid), and test data points (red dots). Trained was an polynomial model of degree $d = 5$ with OLS estimation, and $N = 200$ training and test data points were used. A train-test ratio of 80-20 was used.

Confidence interval

In order to assess the uncertainty in the regression parameter estimation, we calcualted confidence intervals at the level of 95% for the OLS polynomial model with degree $d = 5$. Figure 9 shows the $p = 21$ OLS regression coefficients β_i together with error bars representing the confidence intervals. These were calculated using

$$[\hat{\beta}_i \mp t_{n-p;\,1-\alpha/2} \cdot se(\beta_i)],$$

as introduced in Section 1.1. One can see a rough pattern in the visualization of the confidence intervals. The error bars in the middle are larger than those further out. Since the regression coefficients are ordered in such a way, that the ones in the middle correspond to a higher mixture in the monomial degrees of x_1 and x_2 (think of as x and y), one can say that there is greater uncertainty the more x_1 and x_2 are mixed.

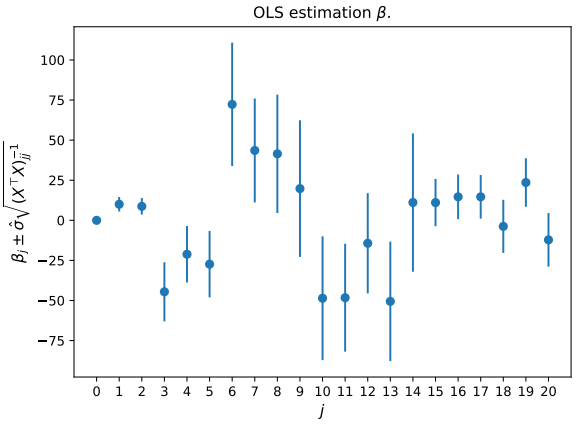

Figure 9. OLS regression parameters β_i for $i = 1, \ldots, 20$ of a polynomial model with order $d = 5$. The erroer bars represent the confidence intervals at a niveau of 95%.

Bias-variance tradeoff

As shown in Figure 6 there clearly exists a certain model complexity (here polynomial degree) where the testing MSE curve is minimized (testing R2 maximized). In Section 2.1 we saw that the MSE can be decomposed into two main sources of error, the bias and the variance. In order to investigate this, we utilized bootstrapping with a boostrap size of $n = 100$ to estimate the bias and variance of models with polynomial degree up to order $d = 10$. The results are shown in Figure 10, where one clearly sees the MSE (black curve) having the same bell shape. For a lower model complexity the variance (blue curve) is very low, while a higher model complexity leads to overfitting. Think of it as a constant function (low model complexity) can not vary much for different test data sets, while a model of high model complexity results in very different results when applied to different test data sets. On the other hand a low model complexity can't fit a complicated data set leading to high bias (red curve). This verifies the bias-variance decomposition and tradeoff.

Further, one can ask how the total sample size $N \in \mathbb{N}$ influences the bias and variance and thus the MSE. For this purpose we gathered 200 logarithmically spaced sample sizes between 10^2 and 10^5 and splitted them with a ratio of 80-20 in train and test data sets. Then we trained a model of fixed degree $d = 5$ for each of the 200 sample sizes. Again, bootstrapping was utilized to estimate the bias, variance, and MSE. Figure 11 shows the same three curves against the different sample sizes $N \in \mathbb{N}$. We can see that the overall MSE (black curve) consists of basically the bias (red curve) as the sample sizes grows since the variance decreases. This also makes sense, because the more data (more information about Franke's function) is available, the more likely the bootstrap samples will produce equal results, such that the variance becomes smaller. In addition, a certain asymptotics sets in, where MSE no longer decreases. One explanation for this is that a sufficiently large number of data points $N \in \mathbb{N}$ and a sufficiently good model complexity of $d = 5$ were used.

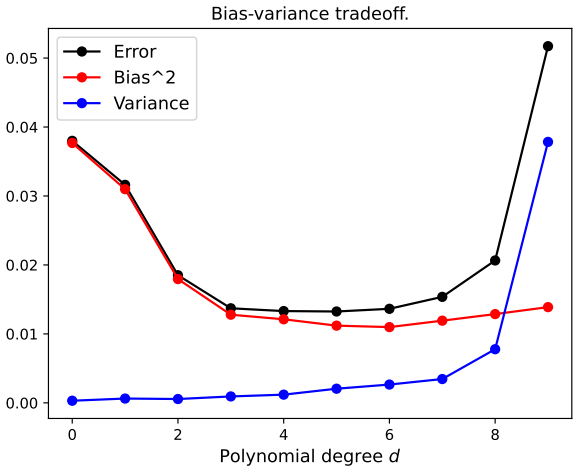

Figure 10. Visualizing the bias-variance tradeoff using polynomial regression models of different order $d \in \mathbb{N}$.

Level of white noise

As the MSE consists not only of bias and variance but also the irreducible error σ^2 coming from the white noise $\varepsilon \sim N(0, \sigma^2)$, we want to investigate this further. Our expectation that MSE becomes larger for larger variance σ^2 is verified. This can be observed in Figure 12, which clearly shows the effect of varying the variance σ^2 of the white noise $\varepsilon \sim N(0, \sigma^2)$ on the train and test MSE. The white noise was added to $N = 200$ sampled points from Franke's function and a polynomial regression

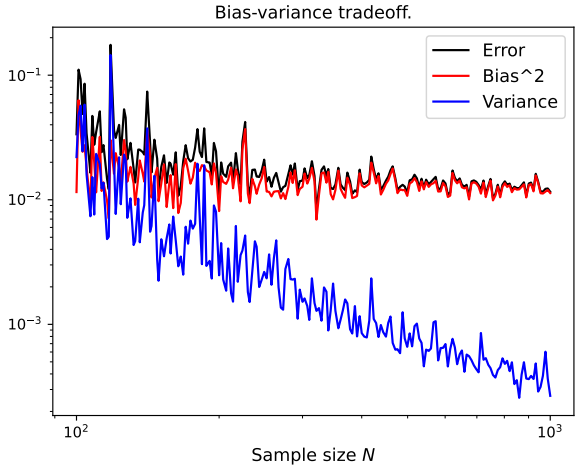

Figure 11. Visualizing the bias-variance tradeoff using a polynomial regression model of order $d = 5$ and different sample sizes $N \in \mathbb{N}$.

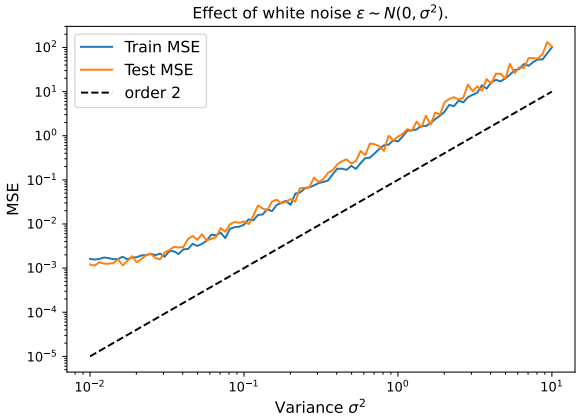

Figure 12. MSE performance on a train and test data set for a polynomial model with degree $d = 5$ against the noise level σ^2 in $\varepsilon \sim N(0, \sigma^2)$. In total, $N = 200$ datapoints were sampled from Franke's function.

model of order $d = 5$ was used. The plot also shows the reference line of order 2 which matches the slope of the results almost perfectly. This means that if one doubles the noise σ^2 in the data, the MSE performance deteriorates by a factor of four.

Bootstrap vs. Cross-validation

In order to investigate the difference on model performance between bootstrap and cross-validation we trained polynomial regression models from order $d = 1$ up to a maximal degree of $d = 8$. We used a white noise $\varepsilon \sim N(0, 0.1^2)$, $N = 400$ data points and a train-test ratio of 80-20. Further, we used a bootstrap sample size of $n = 50$ and 5-fold cross validation. Figure 13 shows the bootstrap (red) as well as cross-validation (blue) train and test MSE curves against the polynomial degree $d \in \mathbb{N}$. We can see that the MSE performance are quite similar for both of the resampling techniques, but the training MSE curves differ a bit. Furthermore, both methods show a minimum test MSE at the polynomial order of $d = 4$.

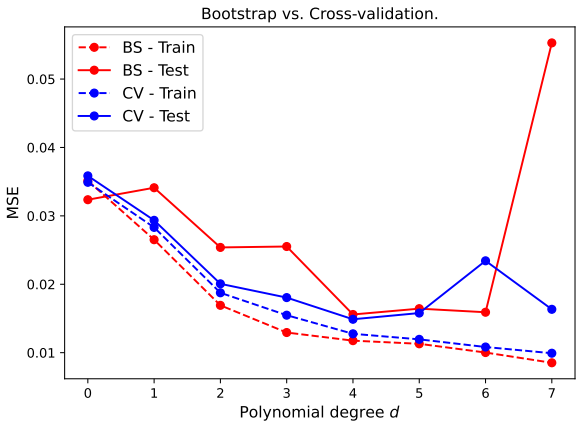

Figure 13. Shown are the train (dotted) and the test (solid) MSE curves generated by the bootstrap (red) and the cross-validation (blue) method. The x-axis is showing the polynomial degree $d \in \mathbb{N}$.

Ridge regression

Figure 14 shows the MSE loss for training and testing for both, OLS and Ridge, against the polynomial degree $d \in \mathbb{N}$. Here, we used $\varepsilon \sim N(0, 0.1^2)$,

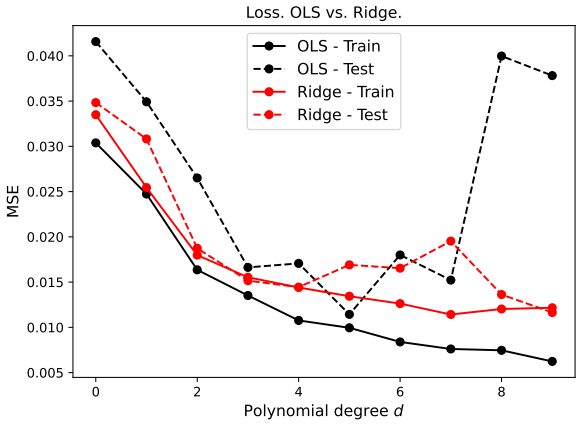

Figure 14. MSE train (solid) and test (dotted) loss for the OLS estimation (black) and Ridge regression (red) against the polynomial degree $d \in \mathbb{N}$.

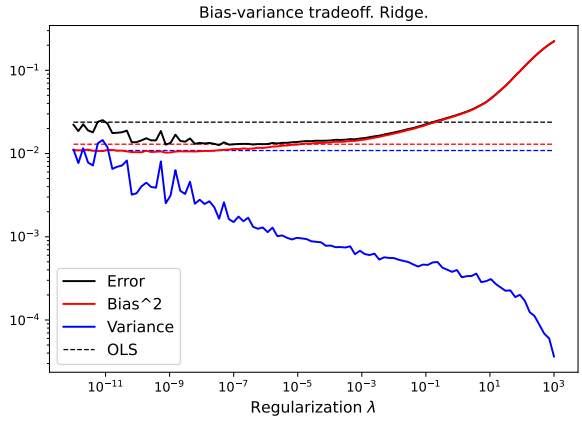

Figure 15. Bias, variance and MSE for the Ridge and OLS regression model as a function of the regularization parameter $\lambda > 0$.

$N = 500$ data points and a regularization parameter of $\lambda = 10^{-3}$. One can observe different behaviour for the two models. While a common OLS estimation gives, at first glance, better results (especially in training), the test curves are very different. This can theoretically be explained by the nature of the ridge regression. Penalising the parameters leads to a lower variance of the model, so that the test MSE for higher polynomial degrees is more stable and lower than that of the OLS sharpening. For the OLS estimation, we examine a very low train MSE for high model order while the test MSE explodes, which is a prototypical example of overfitting.

In order to investigate and explain the different behaviour of OLS and Ridge better bootstrapping a bias-variance analysis is inevitable. For this purpose we performed a Ridge estimation for 100 logarithmically spaced regularization parameters λ between 10^{-12} and 10^{3} and a fixed, relatively high polynomial degree of $d = 10$. Figure 15 shows the bias, variance, and MSE for the Ridge, as well as for the OLS estimation as a function of the above mentioned $\lambda > 0$. Note that the OLS estimation is independent of λ and is therefore marked as dotted, constant lines. We also note the behaviour $\beta_R \to \beta_{OLS}$ as $\lambda \to 0$. Above all, however, the basic idea of a penalty term becomes clear: the variance decreases the larger the regularisation parameter λ, since the parameters no longer have as much freedom as in a flexible OLS estimation. We see, that a regularization parameter of around $\lambda 10^{-6}$ results in the lowest overall MSE.

Lasso regression

Figure 16 shows the MSE loss for training and testing for both, OLS and Lasso, against the polynomial degree $d \in \mathbb{N}$. Here, we used $\varepsilon \sim N(0, 0.1^2)$, $N = 500$ data points, and a regularization parameter of $\lambda = 10^{-5}$. We observe a similar behaviour as with the Ridge regression in Figure 14. The penalty in lasso regression ensures lower test MSE values, as the parameters must not be too flexible and thus generalise better. Again, we see that OLS regression achieves better train MSE values due to the more flexible model design. Figure 17 also shows a similar situation as Figure 15 for the Ridge regression. The variance decreases with a larger regularization parameter $\lambda > 0$, but the bias increases. There is a regularization parameter $\lambda \approx 10^{-1}$ above which the MSE converges and there is no more change. Again the bias, variance, and MSE for the OLS estimation are marked as dotted, horizontal lines.

Comparison of the regression coefficients

Finally, we want to investigate the role of the regularization parameter $\lambda > 0$ on the Ridge β_R and Lasso β_L coefficients. Therefore, we fitted a Ridge

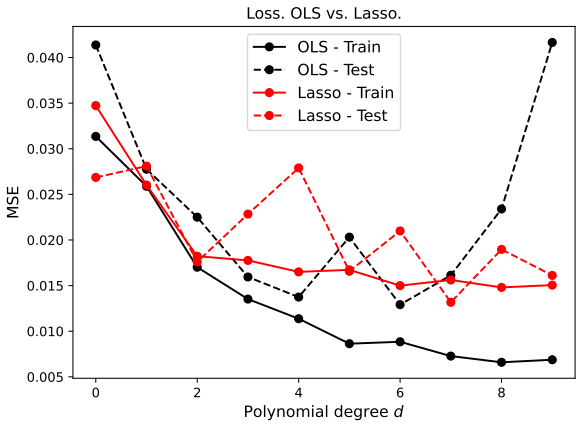

Figure 16. MSE train (solid) and test (dotted) loss for the OLS estimation (black) and Lasso regression (red) against the polynomial degree $d \in \mathbb{N}$.

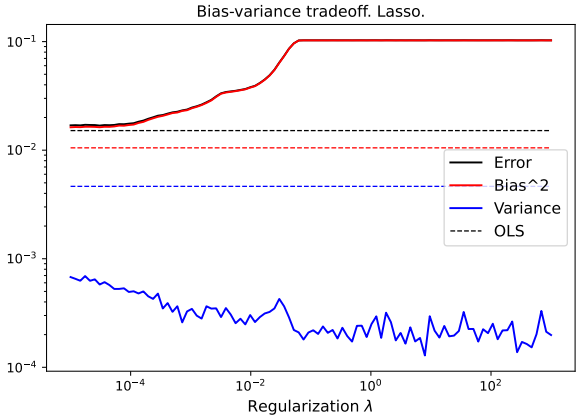

Figure 17. Bias, variance and MSE for the Lasso and OLS regression model as a function of the regularization parameter $\lambda > 0$.

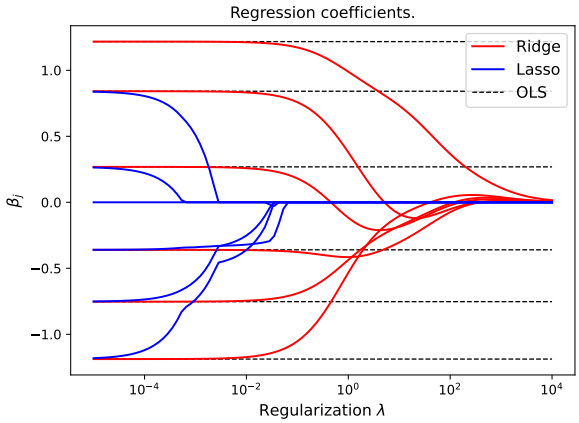

Figure 18. Regression coefficients for β_R and β_L as a function of the regularization parameter $\lambda > 0$.

and Lasso regression for 100 logarithmically spaced regularization parameters λ between 10^{-5} and 10^4. Figure 18 shows the regression coefficients for Ridge β_R and Lasso β_L regression as a function of the regularization parameter $\lambda > 0$, as well as the OLS estimation parameters β_{OLS} as dotted, constant lines. One can clearly observe the behaviour $\beta_R \to \beta_{OLS}$ and $\beta_L \to \beta_{OLS}$ as $\lambda \to 0$. Furthermore, we can see the difference between Ridge and Lasso regression coefficients. While the coefficients in Ridge regression β_R do become smaller (and converge towards zero), they fall to zero much faster in Lasso regression. This difference was also shown when the two penalty methods were introduced.

3.2 Terrain

We gathered the terrain data from the U.S. Department of the Interior U.S. Geological Survey's (USGS) EarthExplorer.[10] In particular, we chose a specific terrain in Møsvatn Austfjell area in Norway. Figure 19 shows the terrain on the left hand side, while $N = 2000$ uniform, random sampled train and test points are shown on the right hand side. Training data is shown in black, while testing (or valdiation) data is shown in red.

In order to fit a polynomial regression model to the terrain data over Norway presented in section 5, we still need to go a little deeper into data manipulation. The original data set consists of

[10] `https://earthexplorer.usgs.gov/`.

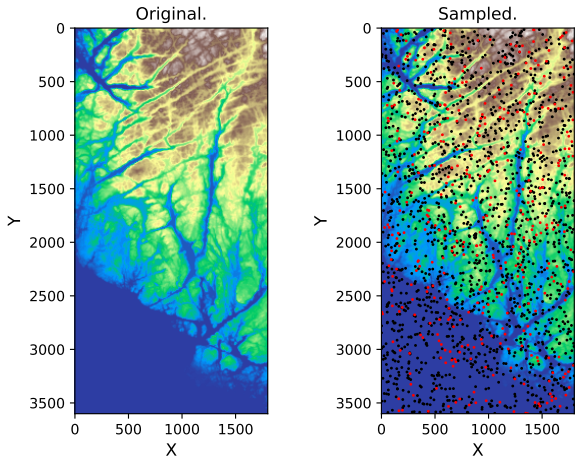

Figure 19. Terrain over Norway in original form (left), as well as $N = 2000$ uniformly sampled train (black) and test (red) data points (right). A train-test ratio of 80-20 was used.

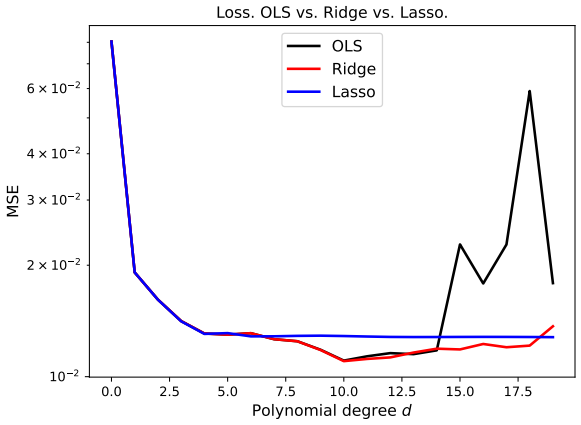

Figure 20. Loss of the three methods OLS, Ridge, and Lasso regression when applied to the terrain data problem.

$3601 \times 1801 = 6.485.401$ points, of which we will only use $N = 2000$ data points in total. The train-test ratio is set to be 80-20, such that we use 1600 points for training, and the remaining 400 for validating our model.

Data scaling

We prepare the data set by standardizing the features such that they follow (approximately) a standard normal distribution. This is done in order for the model to treat each feature the same and put different variables on the same scale. For this purpose, we apply the following transformation

$$s : \mathbb{R}^n \to \mathbb{R}^n, \quad x \mapsto \frac{x - \text{mean}(x)}{\text{std}(x)}$$

to every column of the design matrix X. On the other hand the response variable (height of the terrain) will be normalized into the interval $[0, 1]$ by applying the following normalization

$$n : \mathbb{R}^n \to \mathbb{R}^n, \quad x \mapsto \frac{x - \min(x)}{\max(x) - \min(x)}.$$

OLS, Ridge, and Lasso comparison

In this part we apply all three methods we have learned, that is OLS, Ridge, and Lasso, to this more complex regression problem. We decided on a regularisation parameter of $\lambda = 10^{-4}$ for the penalisation methods and fitted polynomial regression models from order $d = 1$ up to $d = 20$. A comparison of the MSE on the $N = 400$ test data points is shown in Figure 20. We observe a similar behaviour of all three methods for low and moderate model orders $d < 5$. The best performance is achieved by the OLS (MSE $= 0.0112$) and Ridge (MSE $= 0.0110$) regression at order $d = 10$ with a very similar MSE performance. In this example, again, we can clearly see the difference between OLS, Ridge and Lasso regression. Above a certain polynomial degree (here $d = 14$), OLS estimation is too error-prone (high variance), whereas the variance-saving penalty methods deliver more reliable results. Finally, Figure 21 shows the prediction made by the Ridge regression model with smallest MSE on the right hand side compared to the ground truth on the left hand side. At first glance, a good qualitative representation can be observed. However, a closer look reveals insufficient accuracy, so that the fine details in the original terrain cannot be indicated.

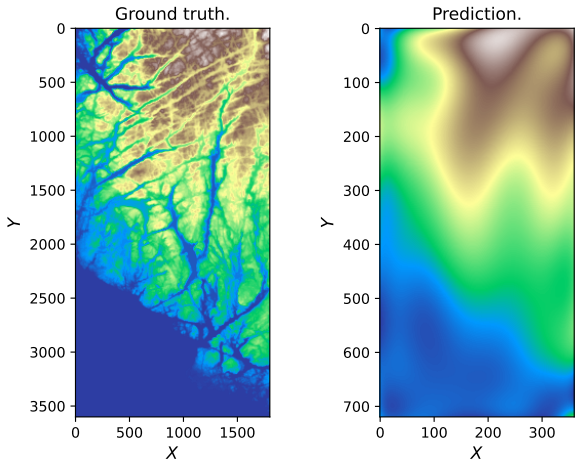

Figure 21. Original terrain (left), as well as the prediction (right) made by the best model: Ridge regression with a regularization parameter of $\lambda = 10^{-4}$.

4 Discussion

We have worked intensively with linear regression and estimated parameters for (inhomogeneous) polynomial models of degree up to $d = 20$. For the estimation we used ordinary least squares (OLS) estimation as well as penalty methods such as Ridge and Lasso. On the one hand, we sampled Franke's function as an analytical function expression, and on the other hand, we used real terrain data. The results show that the OLS estimator, as the best linear, unbiased estimator, always gives very good results, but has a large variance if the polynomial degree is too high. In contrast, Ridge and Lasso regression have proven their strengths at very high polynomial degrees by regularizing parameters and thus and generalising better. It does not seem sensible at first to introduce a bias by means of a penalty method. However, it has been shown that a moderate bias can lead to a considerable saving in variance. In this sense, algorithms with very small bias are not necessarily those with the smallest generalization MSE.

Ultimately, we conclude that linear models are sufficient for the smooth Franke function but not good enough to represent highly irregular and non-smooth terrain data. A qualitative representation of the terrain is possible, but not in such a way that one can actually compare exact data points. In the future, we would like to deal more intensively with a specific data preprocessing of the terrain. We believe that prior (moderate) smoothing of the terrain can lead to better results using even higher polynomial degrees $p > 20$ coupled with a well chosen regularisation parameter. One could also consider using non-linear models, such as neural networks, to fit irregular data such as terrain data.

References

[1] Richard Franke. *A Critical Comparison of Some Methods for Interpolation of Scattered Data*. Calhoun, 1979. URL: `https://calhoun.nps.edu/handle/10945/35052`.

[2] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[3] Fadil Santosa and William W. Symes. "Linear inversion of band-limited reflection seismograms". In: 7.4 (Oct. 1986), pp. 1307–1330. ISSN: 0196-5204. DOI: `https://doi.org/10.1137/0907087`.

[4] A. N. Tikhonov. "On the stability of inverse problems". In: *Proceedings of the USSR Academy of Sciences* 39 (1943), pp. 195–198.