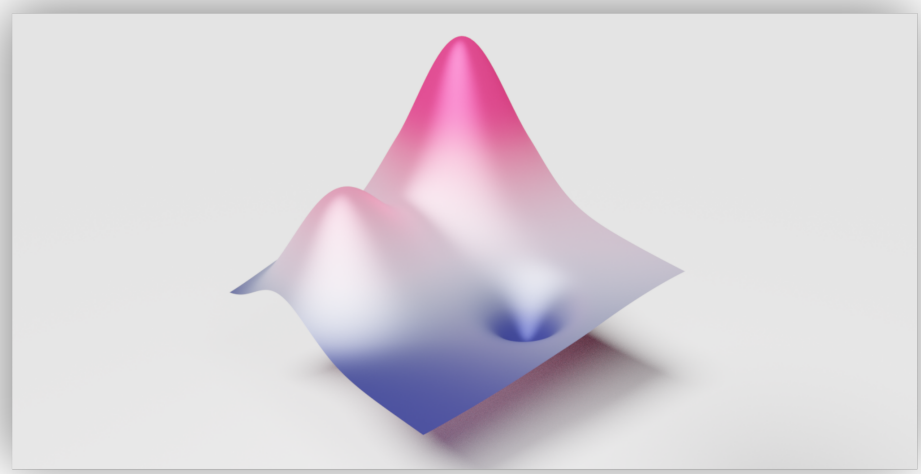

Regression And Resampling Techniques

FYS-STK4155

Assignment One

Ivar Haugaløkken Stangeby

September 18, 2019

Abstract

In this project, we implement and compare three different variants of linear regression. Namely, ordinary least squares (OLS), Ridge regression, and Lasso regression. We test the methods on terrain data taken from a region near Stavanger, Norway. Initial implementation tests are done on an artificial data-set based on noisy samples of the Franke function. Statistical quantities pertaining to the various model functions are computed and compared. We also give an introduction to the theory and motivation behind the three methods.

Contents

1 Introduction

Regression analysis is the act of estimating relationships among variables. In this project, we study various regression methods in more detail. In particular, we compare the *ordinary least squares* (OLS) method with the *Ridge regression* and *Lasso regression* techniques. As we shall see, these three methods are all variations over the same theme. We start by testing the methods on noisy data sampled from a function known as *Franke's* bivariate test function (visualized on the title page), which has been in widespread use in the evaluation of interpolation and data fitting techniques. Finally, we run regression on real terrain data, comparing the aforementioned methods.

2 Regression Techniques

In general, the goal of regression analysis is to fit a *model function* $f(\boldsymbol{x}, \beta)$ to a set of n data points $\Omega = (\boldsymbol{x}_i, y_i)_{i=1}^n$. A simple example is that of a linear polynomial with two parameters:

$$f(x, \beta) = \beta_0 + \beta_1 x. \tag{1}$$

The *model parameters* β are determined in order to minimize a suitable *cost function* $C(\Omega, \beta)$ which measures to which extent the model function manages to capture trends in the data Ω. It is the choice of cost function $C(\Omega, \beta)$ which distinguishes the three regression techniques, OLS, Lasso regression, and Ridge regression.

Assumption 1. It is often assumed a priori that the data is infact generated from a noisy model, such that each y_i can be described as

$$y_i = f(\boldsymbol{x}_i) + \varepsilon_i \tag{2}$$

where each $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ is normally distributed with zero mean and variance σ^2. This assumption on the error gives rise to what is known as *general linear models*.

The design matrix. We are often interested in finding the best model function in a specific function space. Assuming this space has a basis $\mathcal{B} = \left\{ \varphi_i \right\}_{i=1}^{M}$, we may write our model function in terms of the basis functions and the model parameters as

$$f(\boldsymbol{x}, \beta) := \sum_{i=1}^{M} \beta_i \varphi_i(\boldsymbol{x}). \tag{3}$$

As an example, if we were to use the space Π_2^2 of bi-quadratic polynomials, our basis functions would be

$$\mathcal{B} = \left\{ 1, x, x^2, y, y^2, xy \right\}. \tag{4}$$

With this formulation, we can represent the approximation $\hat{\boldsymbol{y}}$ to $\boldsymbol{y}$ as a matrix product

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\beta \tag{5}$$

where $\boldsymbol{X}$ is the *design matrix* defined by components $\boldsymbol{X}_{ij} = \varphi_j(\boldsymbol{x}_i)$, and $\beta = [\beta_1, \dots, \beta_M]$ is the model parameters.

Performance metrics. In order to evaluate the how accurately the model function $f(\Omega, \beta)$ captures trends in the target data $\boldsymbol{y}$, a few standard performance metrics are used. Firstly, the *mean squared error*:

$$\mathrm{MSE}(\boldsymbol{y}, \hat{\boldsymbol{y}}) := \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{6}$$

which simply averages the squared error over all samples and estimates. Secondly, we have the *coefficient of determination* or R^2*-score*:

$$\mathrm{R}^2(\boldsymbol{y}, \hat{y}) := 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \overline{y})^2}. \tag{7}$$

The R^2-score measures much of the variation in $\boldsymbol{y}$ which can be attributed to a simple linear relation between $\boldsymbol{x}$ and $\boldsymbol{y}$. It is a ratio, thus a value of one tells us that all the variation in the data can be attributed to an approximate linear relationship between the data $\boldsymbol{x}$ and the response variable $\boldsymbol{y}$. A value of zero indicates that a non-linear model may be preferrable.

2.1 Ordinary Least Squares

One common cost function is the one which involves the sum of squared residuals (or squared errors):

$$C_{\text{OLS}}(\Omega, \beta) = \frac{1}{n}\sum_{i=1}^{n} \varepsilon_i^2 := \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{8}$$

where $\hat{y}_i := f(\boldsymbol{x}_i, \beta)$. The method involving the minimization of this specific cost function is known as *least squares*. With the matrix notation from above in mind, we can also write the cost function as

$$C_{\text{OLS}}(\Omega, \beta) = \frac{1}{n}(\boldsymbol{y} - \hat{\boldsymbol{y}})^T(\boldsymbol{y} - \hat{\boldsymbol{y}}). \tag{9}$$

Optimizing the parameters in OLS.

Assume now that our model is of the form given in Equation (3) and that our cost function is the mean squared error defined in Equation (8). We are interested in finding the parameters β that minimize the cost function $C(\Omega, \beta)$. Since $C(\Omega, \beta)$ is convex, it suffices to differentiate with respect to β and equating to zero.

We have that

$$\frac{\partial C_{\text{OLS}}(\Omega, \beta)}{\partial \beta} = \boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\beta), \tag{10}$$

and equating this to zero yields the familiar *normal equations*

$$\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{X}^T\boldsymbol{X}\beta. \tag{11}$$

If the matrix $\boldsymbol{X}^T\boldsymbol{X}$ is invertible, we may obtain the solution by direct numerical matrix inversion. In this case, the optimal model parameters are found directly by

$$\beta_{\text{OLS}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{12}$$

However, these matrices may be ill-conditioned when the number of equations are very large, and it is therefore common to apply approximate solvers for the inverse, for instance using low-rank approximation based on SVD-decomposition, which we will briefly turn to in the following.

Singular value decomposition.

Recall that any matrix $\boldsymbol{A} \in \mathbb{C}^{n,m}$ can be decomposed as

$$\boldsymbol{A} = \boldsymbol{U}\Sigma\boldsymbol{V}^T, \tag{13}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are comprised of the eigenvectors of $\boldsymbol{A}\boldsymbol{A}^T$ and $\boldsymbol{A}^T\boldsymbol{A}$ respectively. As these eigenvectors are orthonormal, it follows that both $\boldsymbol{U}$ and $\boldsymbol{V}$ are unitary matrices. Furthermore, Σ is a matrix

$$\Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix}, \tag{14}$$

where Σ_1 is a square diagonal matrix of size $r \times r$ with the non-zero singular values of $\boldsymbol{A}$. The integer r is the *rank* of $\boldsymbol{A}$. As the matrix Σ is mostly containing zeros, the information stored in $\boldsymbol{A}$ is attributed to only some parts of $\boldsymbol{U}$ and $\boldsymbol{V}$. We can remove the redundant parts, and more compactly express $\boldsymbol{A}$ as $\boldsymbol{A} = \boldsymbol{U}_1\Sigma_1\boldsymbol{V}_1^T$ without loss of accuracy in the decomposition. Here, $\boldsymbol{U}_1$ is $m \times r$ and $\boldsymbol{V}_1$ is $n \times r$.

Applying the singular value decomposition to the matrix $\boldsymbol{X} = \boldsymbol{U}\Sigma\boldsymbol{V}^T$ we can analyze the expression for the prediction $\hat{\boldsymbol{y}}$ in terms of the matrix $\boldsymbol{X}^T\boldsymbol{X}$. First of all, we have that

$$\begin{aligned} \boldsymbol{X}^T\boldsymbol{X} &= \boldsymbol{V}\Sigma^T\boldsymbol{U}^T\boldsymbol{U}\Sigma\boldsymbol{V}^T \\ &= \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T, \end{aligned} \tag{15}$$

where we have used that $\boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{I}$ and defined $\boldsymbol{D} = \text{diag}(\sigma_1^2, \dots, \sigma_n^2)$. Thus, plugging this into Equations (5) and (12), we obtain the expression

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\beta = \boldsymbol{X}(\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T)^{-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{16}$$

Finally, substituting $\boldsymbol{X} = \boldsymbol{U}\Sigma\boldsymbol{V}^T$ and noting that diagonal matrices always commute, we end up with

$$\hat{\boldsymbol{y}} = \boldsymbol{U}\boldsymbol{U}^T\boldsymbol{y}. \tag{17}$$

The bias-variance tradeoff

Since we are attempting to estimate a true distribution based on finite samples, there are certain statistical properties it is worthwile discussing. We follow the derivation

from [Meh+19]. Recall Assumption 1 which states that our true data is generated from a noisy model

$$y = f(\boldsymbol{x}) + \varepsilon \tag{18}$$

with $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. We select a finite sample from the true data, which constitutes our finite data-set Ω. Furthermore, we have a model function $f(\Omega, \beta)$ which depends on the selected data-set and the model parameters β. Thus, our cost-function C will depend on the specific sample taken from the true distribution. It is of interest to examine the expected out-of-sample error $\mathbb{E}_{\Omega,\varepsilon}[C(\Omega, \beta)]$ of the model function over the sampled data set Ω and the noise ε. This can be decomposed into three distinct parts, *variance*, *bias*, and *noise*:

$$\mathbb{E}_{\Omega,\varepsilon}[C(\Omega, \beta)] = \frac{1}{n}\sum_{i=1}^{n}\left(f(\boldsymbol{x}_i) - \mathbb{E}_\Omega\left[\hat{y}_i\right]\right)^2 + \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_\Omega\left[\left(\hat{y}_i - \mathbb{E}_\Omega[\hat{y}_i]\right)^2\right] + \sigma^2. \tag{19}$$

See Appendix A for the full derivation. The bias-term measures the deviation of the expectation of the model function $f(\Omega, \beta)$ from the true value f, while the variance-term measures the fluctuations arising due to using different samples of the true data.

Now, note that the equation above suggests that we should minimize both the variance and the bias of the model, in order to obtain the smallest generalization error of the model function. The variance term will be larger for complex models, as performing a fit on one sampled data-set will not neccessarily translate well to another sampled data-set. The bias term measures errors introduced by making certain assumptions about the relations between the covariates $\boldsymbol{X}$ and the response $\boldsymbol{y}$. In our case, we are assuming a linear relationship (recall *linear regression*), and thus if the true underlying relation is non-linear, our bias will be high. More complex models will in general have lower bias.

This is known as the *bias-variance tradeoff* as having a highly complex model, we reduce the bias, yet increase the variance. Having a simple model (for instance a linear model), we increase the bias, and reduce the variance.

2.2 Ridge Regression

When the dimensionality of the matrix $\boldsymbol{X}$ is high, the problem of near-collinear columns often arise, as mentioned earlier. Thus, the model parameters β_{OLS} of the ordinary least squares regression cannot be determined. The propsed fix to this problem is to slightly perturb the matrix $\boldsymbol{X}^T\boldsymbol{X}$ by adding small positive values along the diagonal to ensure its invertability. This is known as *ridge regression* and yields a model parameter estimator

$$\beta_{\text{ridge}}(\lambda) = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{20}$$

where $\lambda \geq 0$ is a hyper parameter. To see why the resulting modified matrix is invertible, one has to look at the singular values. It turns out that the model parameters $\beta_{\text{ridge}}(\lambda)$ directly minimize the *ridge loss function* defined as

$$C_{\text{ridge}}(\Omega, \beta; \lambda) := \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\sum_{i=1}^{p}\beta_i^2 = C(\Omega, \beta) + \lambda\beta^T\beta, \tag{21}$$

which we recognize as standard penalized least squares.

Ridge regregression carries several benefits. One of the more important is its resilience to overfitting. In a high-dimensional setting, if many of the data-features can be attributed to a few covariates, then the remaining covariates do nothing more than fit to the noise. Large values of β_i is then a good indication that your model might be overfitting as some covariates are given more weight than others. The Ridge regression solves this by directly penalizing large model parameters in the cost function.

2.3 Lasso Regression

One drawback of the OLS and Ridge regression approaches is that irrelevant features may still be present in the model. Lasso regression fixes this by — in addition to penalizing large model parameters — also setting sufficiently small parameters to zero. The associated cost function reads

$$C_{\text{lasso}}(\Omega, \beta; \lambda) := \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\sum_{i=1}^{n}|\beta_i|. \tag{22}$$

Again, $\lambda \leq 0$ is a model hyper-parameter. Note that the penalization-term now involves absolute values of the model parameters. As the Lasso estimator may set model parameters to zero, it is effectively producing a sparser model.

Attempting to differentiate the Lasso cost function with respect to the model parameters, we see that

$$\frac{\partial C_{\text{lasso}}(\Omega, \beta; \lambda)}{\partial \beta_i} = \frac{\partial C_{\text{OLS}}(\Omega, \beta)}{\partial \beta_i} + \lambda \operatorname{sign}(\beta_i). \tag{23}$$

The analysis of the optimal Lasso parameters is made difficult due to the discontinuity of the sign function, and no closed form solution as in OLS and Ridge-regression can be found in the general case. Thus, numerical optimzation techniques are often used, for instance gradient ascent/descent.

For further information on Ridge and Lasso regression, the excellent article "Lecture Notes on Ridge Regression" by van Wieringen [vWie15] is highly recommended.

3 Implementation and methodology

In order to verify the implementation, we test on sampled data from the Franke function [Fra79]. This bivariate function $F(x, y)$ is defined for $x, y \in [0, 1]$ as follows:

$$\begin{aligned} F(x,y) :=& \frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) \\ &+ \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)^2}{10}\right) \\ &+ \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) \\ &- \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right). \end{aligned} \tag{24}$$

We are interested in fitting a bivariate polynomial surface to this data, by means of the regression techniques discussed in Chapter 2. That is, our model function $f(\Omega, \beta)$ lies in the polynomial space Π_2^d of bi-degree (d, d) polynomials. We are mainly interested in bi-quintic polynomial surfaces, and will therefore restrict our attention to $d = 5$ in the numerical experiments.

We sample the function F both with and without noise at $N = 5000$ points in each

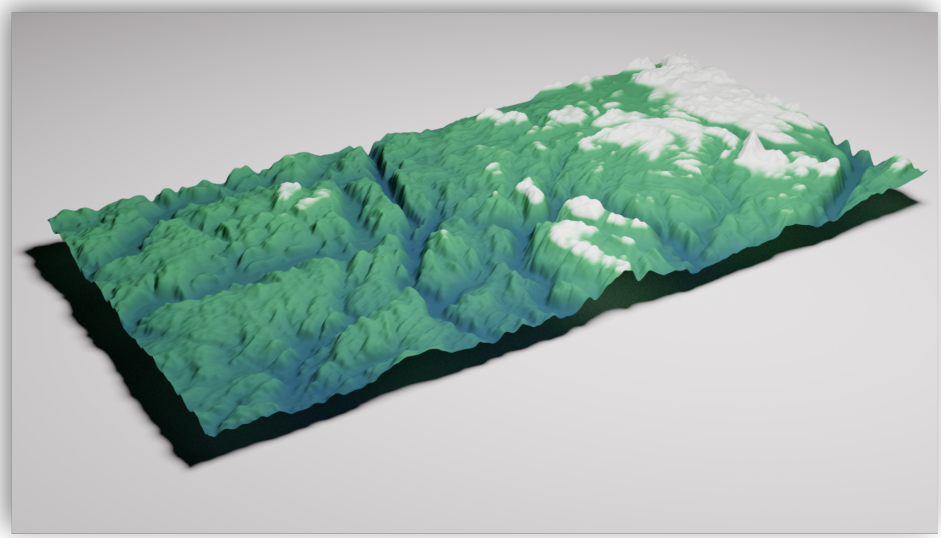

Figure 1: Real terrain data from a region close to Stavanger, Norway. For the regression tasks, we select data from small rectangular regions to keep the data-sizes manageable. The data is supplied in the file `SRTM_data_1_Norway.tif`.

parameter direction, yielding a matrix of function values

$$\begin{aligned} \boldsymbol{Z} &= \left[F(x_i, y_j)\right]_{ij}, \\ \boldsymbol{Z}^\star &= \left[F(x_i, y_j) + \alpha\varepsilon_{ij}\right]_{ij}, \end{aligned} \tag{25}$$

where each $\varepsilon_{ij} \sim \mathcal{N}(0, \sigma^2)$ and α is the *signal-to-noise ratio*. The design matrix $\boldsymbol{X}$ is computed using the PolynomialFeatures-method from the sklearn package. The dimension of the bi-quintic polynomial space is 21, yielding a design matrix of size $N^2 \times 21$.

For the real data set, we use the supplied Tiff data of an area close to Stavanger in Norway, as displayed in Figure 1. The data-set consists of approximately 1800×3600 data-points.

The ordinary least squares and the Ridge regression is performed by solving the corresponding normal equations. We use the Lasso regression available in sklearn package.

3.1 Model statistics and resampling methods

We are interested in computing certain statistical quantities pertaining to our data and corresponding model function. Firstly, the *variance* of the model parameters β over all possible data sets tells us how sensitive β is to changes in the sampled data set. Using Equation (12) for the optimal OLS-parameters and setting $\boldsymbol{A} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$, we compute:

$$\text{Cov}\left[\beta_{\text{OLS}}\right] = \text{Cov}\left[\boldsymbol{A}\boldsymbol{y}\right] = \boldsymbol{A}\,\text{Cov}\left[\boldsymbol{y}\right]\boldsymbol{A}^T = \boldsymbol{A}\boldsymbol{A}^T\sigma^2 = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\sigma^2. \tag{26}$$

In this calculation, we have used Assumption 1 to deduce that $\text{Cov}[\boldsymbol{y}] = \text{diag}(\sigma)$, a diagonal matrix, which commutes with other matrices. It then follows that the estimated variance of β_{OLS} is given by

$$\text{Var}[\beta_{\text{OLS}}] = [\sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}_{ii}]_{i=1}^{n}. \tag{27}$$

Furthermore, by direct computation, we see that β_{OLS} is an *unbiased* estimator, as

$$\mathbb{E}[\beta_{\text{OLS}}] = \beta. \tag{28}$$

With both the expected value and the variance of β, we may formulate a confidence interval for β. Thus for an estimated parameter β_i we construct the 95%-confidence interval as

$$\begin{aligned} I_i &= (\mathbb{E}[\beta_i] - 1.96 \times \text{SD}[\beta_i], \mathbb{E}[\beta_i] + 1.96 \times \text{SD}[\beta_i]) \\ &= \left(\beta_i - 1.96 \times \sigma\sqrt{(\boldsymbol{X}^T\boldsymbol{X})^{-1}_{ii}}, \beta_i + 1.96 \times \sigma\sqrt{(\boldsymbol{X}^T\boldsymbol{X})^{-1}_{ii}}\right). \end{aligned} \tag{29}$$

Here $\text{SD}[\beta_i]$ denotes the standard deviation of β_i. The interpretation of this interval is "under repeated experiments, the number of computed confidence intervals that actually include the true parameter, will tend to 95%".

4 Numerical Results

In this section we present some of the results of the numerical simulations. We start simple by first considering the effects of model complexity — in terms of polynomial

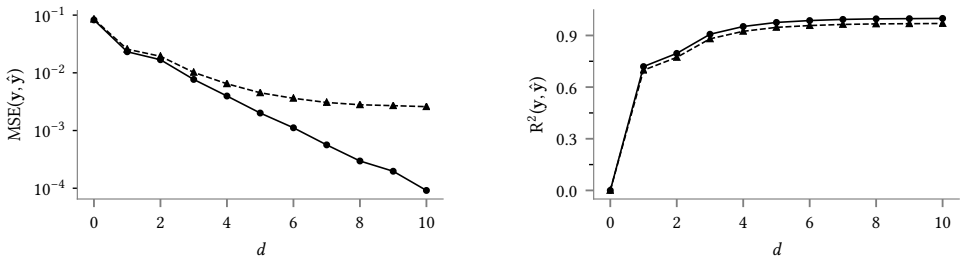

Figure 2: Ordinary least squares regression on samples from the Franke function. The mean squared error (left) and the R^2-score (right) as a function of the polynomial degree d. The values are displayed both for the true sample data (solid) and the noisy samples (dashed). The noise is distributed as $\mathcal{N}(0, \frac{5}{100})$.

degree — using OLS regression on the Franke dataset. We assess the model accuracy by means of the mean squared error and the R^2-score discussed previously. In order to increase the reliability of the statistical estimates, we employ k-fold cross validation.

We then take a closer look at Ridge and Lasso regression techniques, and find the optimal hyper-parameter λ. Again, we use the Franke dataset. In this context, we also see the effects of the bias-variance tradeoff.

Finally, we compare our methods on real world data, namely the terrain data in Figure 1.

4.1 Toy Data

Ordinary Least Squares

We sample $N = 1000$ points in each parameter direction and compute the corresponding N^2 funciton values of the Franke function both with and without added noise. The noise was selected to be distributed as $\mathcal{N}(0, 1)$ with a signal-to-noise ratio $\alpha = 0.05$.

We then perform ordinary least squares, and compute the mean squared error as well as the R^2-score for both the true and the noisy data. These are presented in Figure 2. With the Franke function being a weighted sum of exponentials, the MSE-error tends to zero as the polynomial degree increases. The added noise reduces the convergence

Table 1:

rate, as is to be expected.

In addition to the evaluation metrics, we also compute confidence intervals for each model parameter. These are displayed in Table 1 for each polynomial degree up to and including $d = 5$.

4.2 Terrain Data

5 Conclusion

5.1 Further work

In this project we have discussed three different methods of linear regression. In all three cases, we have fitted polynomial surfaces to the data sets. One draw-back with using polynomial surfaces is that they are inherently global methods. Thus, in order to capture local trends in the data, a very complex model is required, which (in the polynomial case) means surfaces of higher polynomial degree. In recent years, several attempts at constructing efficient local methods have been proposed. Spline-based (piecewise polynomial) methods in particular have been important, and several approaches to locally refined spline spaces have started to gain momentum. We note in particular that of *hierarchical B-splines* (HB), *truncated hierarchical B-splines* (THB) and *locally refined B-splines* (LRB). These methods allow us to place more degrees of freedom in regions of high data-variance, thus locally increasing the model complexity.

It would be interesting to make a comparison of the three methods in such a simple setting as linear regression, as done in this project, taking into account adaptive refinement of the spline spaces with respect to the sampled data set.

A Derivation of the bias-variance-noise relationship

Combining the two previous equations, we obtain

References

[Fra79] Richard Franke. *A Critical Comparison of Some Methods for Interpolation of Scattered Data*. NPS53-79-003. NAVAL POSTGRADUATE SCHOOL MONTEREY CA, Dec. 1979.

[Meh+19] Pankaj Mehta et al. "A High-Bias, Low-Variance Introduction to Machine Learning for Physicists". In: *Physics Reports* 810 (May 2019), pp. 1–124. ISSN: 03701573. arXiv: 1803.08823.

[vWie15] Wessel N. van Wieringen. "Lecture Notes on Ridge Regression". In: (Sept. 30, 2015). arXiv: 1509.09169 [stat].