Regression analysis and resampling methods

Nicholas Karlsen and Thore Espedal Moe
University of Oslo
(Dated: October 9, 2020)

In this paper we aim to explore the application of three distinct regression methods to both a sampling of the Franke function, as well as a set of terrain data. We utilize the ordinary-least-square (OLS) regression, Ridge regression and LASSO regression along with bootstrap and k-fold cross-validation re-sampling techniques in order to create and asses predictive polynomial models. We analyze the performance of the different regression models on the two data sets, with the ultimate goal of determining the best regression method and the best predictive polynomial model for each data set.

I. INTRODUCTION

In essence, Linear Regression is the process of taking points from a function, or a set of measurements, and mapping them linearly to coordinates in a chosen basis in order to create an approximation, or model, explaining the original dataset and estimating unseen points in the domain spanned by the original data set. That is, from a limited set of data try to infer a linear functional relationship between some chosen prediction variables and the measured/sampled responses, and use this functional relationship to predict new data points in the domain. Typically, one is either interested in extracting the functional relationship between the measured responses and the prediction variables in order to say something about the relative importance of the conjectured predictors; or one wants to create a "best possible" predictive map of the (mostly unmeasured) response variables as a function of the (mostly unmeasured) prediction variables. In both cases what one really wants is some form of optimal mapping from prediction variables to response variables, and that mapping can be estimated by the use of linear regression methods.

Obviously, this way of doing predictions and inferences has an enormous range of possible applications; in basically every instance where one expects some linear relationship between selected prediction variables and a response variable one can, if one has a data set containting measurements of the variables, try to use regression methods to create an optimal model. Examples might be the construction of a model for housing prices as a function of distance from city center, living area, building year and whatever else one could imagine might affect the prices. Or one could wish to determine the best coefficients in a model for nuclear binding energy as a function of nucleon numbers.

In this paper we investigate two illustrative and similar applications: The reconstruction of the Franke function from sampled values, and the (re-)construction of an elevation map from a donwsampled set of terrain data. In a way, our data sets are "low-resolution images" of the Franke function and the terrain. Our goal is then to find the best possible "high-resolution images" by using different regression methods and resampling techniques on the sampled data, using tow-dimensional polynomials as basis functions. In other words, we want to find the most faithful recreation of the original terrain data and the Franke function, in the basis of two-dimensional polynomials.

We begin with a theoretical discussion of the regression and resampling methods that will be used, as well as some comments on the significance of the so-called Bias-Variance-Tradeoff in the context of linear regression. We then proceed to investigate the application of ordinary-least-squares (OLS), Ridge regression and LASSO regression to samplings of the Franke function. We use k-fold cross-validation to assess the mean-squared-error (MSE) of our different models, and we use bootstrap resampling in order to estimate the bias and variance of our models. We discuss the influence of the number of points sampled, the polynomial degree we try to fit, the values of the hyper-parameter lambda for the Ridge and LASSO regressions and the presence/absence of noise in our samplings. Afterwards we will move on to the terrain data, where we perform much of the same analysis as for the Franke function. We then try to evaluate which regression methods perform the best for different use cases, and which resulting models might be considered the 'best'.

II. THEORY

A. Linear Regression

Consider a set of data points $\{(x_1, y_i), \dots, (x_N, y_N)\}$ which we wish to fit to some linear model $\tilde{\boldsymbol{y}}(\boldsymbol{\beta})$ where $\boldsymbol{\beta}$ is a vector containing the free parameters of the model. The model $\tilde{\boldsymbol{y}}$ will then be related to the true data points $\boldsymbol{y}$ by

$$\boldsymbol{y} = \tilde{\boldsymbol{y}}(\boldsymbol{\beta}) + \boldsymbol{\varepsilon} \tag{1}$$

where $\boldsymbol{\varepsilon} = (\varepsilon_1, \dots, \varepsilon_N)$ represents the error of the model. The aim of Linear regression models is thus to find the optimal parameters $\boldsymbol{\beta}$ such that not only the error of the model on some particular dataset is minimized, but also the error of the model applied to different data sets sampled in the same domain is minimized such that we may use our model to make predictions about new datasets.

1. *The Design Matrix*

When constructing a model, we first have to chose a set of basis functions. A popular choice for which is $\mathbb{P}_n$, which models a wide range of different phenomena and is also the choice we will make for this paper. But in principle, any set of linear basis functions may be chosen. For a single variate polynomial, a model would then take the form

$$\tilde{y}(x) = \beta_0 + \beta_1 x + \cdots + \beta_n x^n \tag{2}$$

When trying to find the optimal $\boldsymbol{\beta}$, a useful construct is the so-called Design Matrix, which for a polynomial basis takes the form

$$\mathrm{X} = \begin{bmatrix} 1 & x_1 & \dots & x_1^n \\ 1 & x_2 & \dots & x_2^n \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & \dots & x_N^n \end{bmatrix} \tag{3}$$

where we see that entry X_{ij} contains the x_i data point evaluated in the j-th basis function. This matrix has the particularly useful property that we may then multiply it with $\boldsymbol{\beta}$ to obtain the corresponding set of predictions $\tilde{\boldsymbol{y}}$ like

$$\tilde{\boldsymbol{y}} = \mathrm{X}\boldsymbol{\beta} \tag{4}$$

leaving us with a linear algebra problem to solve in finding the $\boldsymbol{\beta}$ which optimizes $\tilde{\boldsymbol{y}}$.

This also extends directly to multivariate cases where we have more than one independent variable. In particular for two dimensional polynomials, our choice of basis for this article, we have that an n-th degree polynomial is constructed by all permutations of $x^p y^q$ where $p + q \leq n$ which means that the total number of predictors for degree n is given by

$$\binom{2+n}{n} = \frac{(2+n)!}{n!(2+n-n)!} = \frac{(n+1)(n+2)}{2} \tag{5}$$

2. *Model assessment*

Once we have constructed a model, we a also need a way to quantitatively evaluate its performance. Here, we will focus on two statistical measures; First, we have the *mean squared error* (MSE) defined as

$$\mathrm{MSE}(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \frac{1}{n}\sum_i (y_i - \tilde{y}_i)^2 \tag{6}$$

which as the name suggests is a measure of the mean square distance the model $\tilde{\boldsymbol{y}}$ has from the data set $\boldsymbol{y}$. We also have the *coefficient of determination*, defined as

$$\mathrm{R}^2(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = 1 - \frac{\sum_i (y_i - \tilde{y}_i)^2}{\sum_i (y_i - \mathbb{E}[\boldsymbol{y}])^2} \tag{7}$$

which loosely speaking is a measure of well the proposed model predicts the variance of the data set, where $\mathrm{R}^2 = 1$ corresponds to a perfect fit and $\mathrm{R}^2 < 1$ an increasingly worse fit. It is worth to note that the MSE and R^2 are essentially equivalent when measuring the performance of a model on a particular sample, which is ultimately determined by the $\sum_i (y_i - \tilde{y}_i)^2$ term in both cases. Thus they differ mainly in scaling & centering. However, R^2 will notably yield results which translates somewhat more directly across different models.

A particularly important point to make when considering these measures is that they will both generally favor increasingly higher degree polynomials when looking at some particular data set. Whilst yielding good MSE & R^2 scores this leads to over-fitting, where the model becomes too specialized to one particular sample which means that the overall predictive capabilities of the model actually decreases. We will look closer at the details of this in a later section. For now, we simply note that this problem motivates us to split our data sets into training and test sets. Where we train our model using the training set, and subsequently validate the model by computing the MSE & R^2 scores on the test sets. Thus, if our model becomes too specialized to the training set, this will lower MSE & R^2 scores in the test set, giving us a much better indication on the predictive capabilities of our model.

3. *Ordinary Least Squares*

In ordinary least squares (OLS), we aim to find an optimal set of parameters $\hat{\boldsymbol{\beta}} = [\hat{\beta}_0, \dots, \hat{\beta}_n]^T$ such that the L^2 norm $\|\boldsymbol{y} - \mathrm{X}\boldsymbol{\beta}\|_2$ is minimal, with the L^2 norm being induced by the inner product

$$\|\boldsymbol{u}\|_2^2 = \sum_i u_i^2 = \boldsymbol{u}^T\boldsymbol{u} \tag{8}$$

This defines the cost function for OLS, which may be written as

$$C_{OLS}(\boldsymbol{\beta}) = (\boldsymbol{y} - \mathrm{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \mathrm{X}\boldsymbol{\beta}) \tag{9}$$

In order to find the minima, we differentiate wrt to $\boldsymbol{\beta}$ and assert that $\partial_{\boldsymbol{\beta}} C_{OLS} = 0$ for the optimal predictor. Taking the partial derivative yields

$$\frac{\partial}{\partial \boldsymbol{\beta}} C_{OLS}(\boldsymbol{\beta}) = -2\mathrm{X}^T(\boldsymbol{y} - \mathrm{X}\boldsymbol{\beta}) \tag{10}$$

We assert this is zero at the minima, which yields

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

then taking the inverse of $\mathrm{X}^T\mathrm{X}$ on both sides then gives the optimal $\boldsymbol{\beta}$ as

$$\hat{\boldsymbol{\beta}}_{\mathrm{OLS}} = (\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\boldsymbol{y} \tag{12}$$

which mathematically is the best projection of the datapoints to our model. It can also be shown that the variance of these optimal predictors are given by [1]

$$\mathrm{Var}\left(\hat{\boldsymbol{\beta}}_{\mathrm{OLS}}\right) = \sigma^2\left(\mathrm{X}^T\mathrm{X}\right)^{-1} \tag{13}$$

Where σ^2 denotes the variance of the underlying noise of the sample.

While OLS yields the mathematically best model for some particular sample, it does not necessarily yield the best predictive model. OLS may suffer from over-fitting to the particular sample, and will therefore not always perform very well on new data points for which the model has not been trained. As such, we instead look at alternate regression methods with additional tuning parameters which allow us to account for the variance between different sets of data as to yield a model with better predictive abilities compared to the OLS.

4. Ridge Regression

One such method is the Ridge regression where we instead try to minimize $\|\boldsymbol{y} - \mathrm{X}\boldsymbol{\beta}\|_2 + \lambda\|\boldsymbol{\beta}\|_2$, from which we define our cost function as

$$C_R(\boldsymbol{\beta}) = (\boldsymbol{y} - \mathrm{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \mathrm{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta} \tag{14}$$

Similar for what we did for OLS, we take the partial derivative wrt. $\boldsymbol{\beta}$, where the only difference in finding the minima resides in the additional term

$$\frac{\partial}{\partial\boldsymbol{\beta}}\lambda\boldsymbol{\beta}^T\boldsymbol{\beta} = 2\lambda\boldsymbol{\beta} \tag{15}$$

if we again assert that the optimal $\boldsymbol{\beta}$ is given by $\partial_{\boldsymbol{\beta}}C_R(\boldsymbol{\beta})$ and rearrange we get

$$\hat{\boldsymbol{\beta}}_R = \left(\mathrm{X}^T\mathrm{X} + \lambda\right)^{-1}\mathrm{X}^T\tilde{\boldsymbol{y}} \tag{16}$$

As the optimal β for Ridge regression. So we see that the only difference between OLS and Ridge regression is the addition of the "penalty" parameter λ to the L^2 norm of the coefficients $\boldsymbol{\beta}$. The main point of introducing this parameter is to reduce the variance of the regression coefficients $\boldsymbol{\beta}$. It introduces a constraint on the allowable values of $\boldsymbol{\beta}$, which means that the method no longer is unbiased. The aim is that the reduction of the variance outweighs the increase of the method's bias, leading to an overall lower test error, cf. the later discussion of the Bias-Variance Trade-off. As a technical note, during our computations we center our response-variable and scale plus center our predictors. This has the effect of removing the constant column in the design matrix, so that penalty term doesn't apply to β_0.

5. LASSO Regression

The cost function for the *least absolute shrinkage and selection operator* (LASSO) method is defined as

$$C_L(\boldsymbol{\beta}) = \|\boldsymbol{y} - \mathrm{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 \tag{17}$$

Which as opposed to OLS and Ridge has no analytical solution for finding the optimal $\boldsymbol{\beta}$, which is instead found by optimization methods such as gradient descent. The specifics of this is beyond the scope of this article, and we will simply use the existing libraries provided by scikit-learn [2] to perform the regression.

Qualitatively the LASSO method is distinguished from the Ridge regression by how the norm of $\boldsymbol{\beta}$ is penalized. While the Ridge regression applies the penalty to the L^2 norm, the Lasso aims to shrink the L^1 norm. Two important consequences arise from this: firstly, it is no longer possible to obtain a closed-form solution to the minimization problem; secondly, the regression coefficients may be shrunk far more unevenly. In contrast to the Ridge regression, LASSO regression may shrink individual β_i to zero. This is an extremely useful property of the method, since it allows unimportant predictors to be identified and discarded. The drawback, of course, is that the bias of the Lasso method is expected to exceed the bias of the Ridge method. Furthermore, for highly correlated prediction variables, like our two-dimensional polynomials, the LASSO method might struggle with which of those prediction variables to keep. Once again a technical note; by centering we remove the constant column from the design matrix to avoid penalizing the constant term through β_0.

B. Singular Value Decomposition

Consider an $m \times n$ matrix X of rank r. The singular values of X are then defined as the root of the eigenvalues of the diagonal matrix $\mathrm{X}^T\mathrm{X}$. We may also express X in the so-called singular value decomposition (SVD)

$$\mathrm{X} = \mathrm{U}\Sigma\mathrm{V}^T \tag{18}$$

where U, V are unitary matrices and

$$\Sigma = \begin{bmatrix} D & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} \tag{19}$$

an $m \times n$ matrix, where the matrix D is a diagonal $r \times r$ matrix containing the singular values of $\mathrm{X}^T\mathrm{X}$

$$D = \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n \end{bmatrix} \tag{20}$$

which by convention is ordered such that $\sigma_1 \geq \cdots \geq \sigma_n$.

1. *Application to OLS*

We may use the SVD to re-express the expression for $\hat{\boldsymbol{\beta}}$ in OLS given by Eqn. 12 by writing

$$\mathrm{X}^T\mathrm{X} = \left(\mathrm{U}\Sigma\mathrm{V}^T\right)^T\left(\mathrm{U}\Sigma\mathrm{V}^T\right) = \mathrm{V}\Sigma^T\mathrm{U}^T\mathrm{U}\Sigma\mathrm{V}^T \tag{21}$$

Since U is unitary, it follows that $\mathrm{U}^T\mathrm{U} = \mathbb{I}$, further we have that $\Sigma^T = \Sigma$ since it is a diagonal matrix. We therefore end up with

$$\mathrm{X}^T\mathrm{X} = \mathrm{V}\Sigma^2\mathrm{V}^T \tag{22}$$

inserting this into Eqn. 12 gives us

$$\hat{\boldsymbol{\beta}}_{\mathrm{OLS}} = \left(\mathrm{V}\Sigma^2\mathrm{V}^T\right)^{-1}\mathrm{V}\Sigma\mathrm{U}^T\boldsymbol{y} \tag{23}$$

This gives an alternate way of solving the OLS problem, which becomes particularly useful in situations where we have a large amount of data sampled from a relatively small domain, increasing the chance of X having linearly dependent columns which in turns leads to X, and by extension $\mathrm{X}^T\mathrm{X}$ no longer being invertible.

C. Re-sampling

Re-sampling methods are ways in which we can generate new statistics from our existing data, which as the name suggests implies sampling new data sets from our already existing data. By doing so, we may gain new insights about our data which may not be available through regular analysis, particularly in situations where we are limited by the number of data points.

Here, we will focus on two of many such techniques.

1. *k-fold Cross Validation*

In the k-fold cross-validation re-sampling method, we split our data set S into k equally sized subsets $s_1, \ldots, s_k$. We then for each $i = 1, \ldots, k$ assign the i-th subset as the test set and the remaining $k-1$ subsets as the training set and compute the statistics in the usual way. Then at the end, we compute the mean value of the k sets of statistics. A visual representation of this process can be seen in Fig. 1. When doing cross-validation, typical choices of k are 5 and 10 [1]. Which one is better will depend on how the error scales with the size of the training set, as such, choosing a suitable k requires some analysis. The cross-validation re-sampling provides a good estimate for the mean error of our estimates.

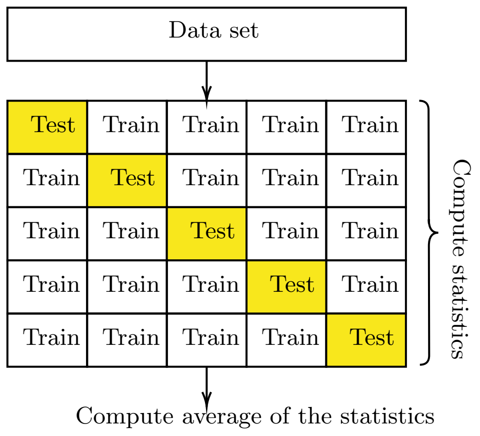


FIG. 1: Visual representation of k-fold cross-sampling for $k = 5$

2. *Bootstrap*

In the bootstrap re-sampling method, we sample our data set $S = \{s_1, \ldots s_N\}$ N-number of times, in particular, we allow sampling the same s_i multiple times. In this way, we generate new datasets in which some points are under-weighted and others overweighted with respect to the original dataset S. Loosely speaking, the concept corresponds to evenly sampling the ensemble of all possible input data sets to our regression method, in order to estimate the variance and expectation values of our model predictions over the ensemble. The idea is that if our training set representatively covers the domain of inputs, each re-sampled regression computation will give a certain prediction $\tilde{\boldsymbol{y}}_{\boldsymbol{B}}$ and these $\tilde{\boldsymbol{y}}_{\boldsymbol{B}}$ will appear with a frequency corresponding to their underlying probability distribution. When we later try to compute statistics of the predictions, we will then be able to use point-estimators on the computed $\tilde{\boldsymbol{y}}_{\boldsymbol{B}}$ to obtain estimates of, for instance, the variance of the predictions or the mean value of the predictions over the ensemble of all possible inputs in the domain spanned by our data.

D. The Bias-Variance Trade-off

A key concept in much of machine learning is the so-called Bias-Variance Trade-off. Simply put the test error of a learned prediction method can be viewed as being partly the sum of two distinct quantities, the bias of the method and the variance of the method. The bias can be thought of as errors due to inflexible assumptions and constraints on the model, while the variance, quite opposingly, is the actual statistical variance of the procedure. Crucially, the variance is connected with the model having too much flexibility, over-fitting on the training data while giving very spread results on the test data. Naturally, the relative sizes of the errors due to bias and the errors due to variance change with the flexibility/-complexity of the model. For an inflexible model, e.g. a polynomial regression model of low degree, the bias is expected to be the dominant contribution to the test error. On the other side of the spectrum, for a very flexible model, e.g. a polynomial regression model of high degree, the variance would be anticipated to be the dominating source of the test error. Importantly, the relative effects of the bias and the variance won't necessarily change at the same rate. Thus, one could hope to find an optimally flexible model which minimizes the combined error due to variance and bias. This is the essence of the Bias-Variance Trade-off; one could, for instance, try to slightly increase the bias of a method in the hopes of reducing the variance far more.

Concretely for the problem at hand in this article, we use the expected MSE of our models on the test data to assess our models. This expected MSE can be explicitly decomposed into a bias term, a variance term and an irreducible error [1]. We assume that our response values $\boldsymbol{y}$ are given by a "true" function $\boldsymbol{f}$ with the addition of normally distributed noise $\boldsymbol{\varepsilon}$ with zero mean:

$$\boldsymbol{y} = \boldsymbol{f} + \boldsymbol{\varepsilon} \tag{24}$$

Denoting our model predictions by $\tilde{\boldsymbol{y}}$ and keeping our "true" test values $\boldsymbol{f}$ fixed, we can then decompose the expected squared test error, the expectations being taken over the ensemble of possible predictions and noise values, as:

$$\begin{aligned}
&\mathbb{E}[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2] \\
&= \mathbb{E}[((\boldsymbol{y}-\mathbb{E}[\tilde{\boldsymbol{y}}])-(\tilde{\boldsymbol{y}}-\mathbb{E}[\tilde{\boldsymbol{y}}]))^2] \\
&= \mathbb{E}[(\boldsymbol{y}-\mathbb{E}[\tilde{\boldsymbol{y}}])^2]+\mathbb{E}[(\tilde{\boldsymbol{y}}-\mathbb{E}[\tilde{\boldsymbol{y}}])^2] \\
&\quad - \mathbb{E}[2((\boldsymbol{y}-\mathbb{E}[\tilde{\boldsymbol{y}}])(\tilde{\boldsymbol{y}}-\mathbb{E}[\tilde{\boldsymbol{y}}]))] \\
&= \mathbb{E}[(\boldsymbol{y}-\mathbb{E}[\tilde{\boldsymbol{y}}])^2]+\mathrm{Var}[\tilde{\boldsymbol{y}}]-2(\boldsymbol{y}-\mathbb{E}[\tilde{\boldsymbol{y}}])\mathbb{E}[(\tilde{\boldsymbol{y}}-\mathbb{E}[\tilde{\boldsymbol{y}}])] \\
&= \mathbb{E}[(\boldsymbol{y}-\mathbb{E}[\tilde{\boldsymbol{y}}])^2]+\mathrm{Var}[\tilde{\boldsymbol{y}}]-2(\boldsymbol{y}-\mathbb{E}[\tilde{\boldsymbol{y}}])(\mathbb{E}[\tilde{\boldsymbol{y}}]-\mathbb{E}[\tilde{\boldsymbol{y}}]) \\
&= \mathbb{E}[(\boldsymbol{y}-\mathbb{E}[\tilde{\boldsymbol{y}}])^2]+\mathrm{Var}[\tilde{\boldsymbol{y}}] = \mathrm{Bias}^2[\boldsymbol{y},\tilde{\boldsymbol{y}}]+\mathrm{Var}[\tilde{\boldsymbol{y}}] \\
&= \mathbb{E}[(\boldsymbol{f}+\boldsymbol{\varepsilon}-\mathbb{E}[\tilde{\boldsymbol{y}}])^2]+\mathrm{Var}[\tilde{\boldsymbol{y}}] \\
&= \mathbb{E}[(\boldsymbol{f}-\mathbb{E}[\tilde{\boldsymbol{y}}])^2]+\mathrm{Var}[\tilde{\boldsymbol{y}}]+\mathrm{Var}[\boldsymbol{\varepsilon}] \\
&= \mathrm{Bias}^2[\boldsymbol{f},\tilde{\boldsymbol{y}}]+\mathrm{Var}[\tilde{\boldsymbol{y}}]+\boldsymbol{\sigma}^2
\end{aligned} \tag{25}$$

In a convenient abuse of notation, the above calculation is meant to be performed element-wise. What has been computed above is the expected squared test error for each data point in the test set separately; for notational convenience stacked up as vectors. To get a total MSE for the whole test set, one simply must take the mean over the vector elements afterwards. This yields:

$$\begin{aligned}
\mathrm{MSE} &= \frac{1}{n}\sum_i^n \mathbb{E}[(y_i-\tilde{y}_i)^2] \\
&= \frac{1}{n}\sum_i^n (\mathbb{E}[(y_i-\mathbb{E}[\tilde{y}_i])^2]+\mathrm{Var}[\tilde{y}_i]) \\
&= \frac{1}{n}\sum_i^n (\mathbb{E}[(f_i+e_i-\mathbb{E}[\tilde{y}_i])^2]+\mathrm{Var}[\tilde{y}_i]) \\
&= \frac{1}{n}\sum_i^n (\mathbb{E}[(f_i-\mathbb{E}[\tilde{y}_i])^2]+\mathrm{Var}[\tilde{y}_i]+\mathrm{Var}[e_i] \\
&= \frac{1}{n}\sum_i^n ((\mathbb{E}[(f_i-\mathbb{E}[\tilde{y}_i])^2]+\mathrm{Var}[\tilde{y}_i]+\sigma_i^2) \\
&= \frac{1}{n}\sum_i^n (\mathrm{Bias}^2[f_i,\tilde{y}_i]+\mathrm{Var}[\tilde{y}_i])+\sigma^2
\end{aligned} \tag{26}$$

Here σ_i^2 is the variance of the noise for each test point. Since all test points are assumed to have the same noise distribution, their variances will be equal. To be perfectly clear, the random variables for both of the preceding calculations are the model predictions $\tilde{y}_i$ and the test point noises ε_i, which are considered to be independent of each other. As an annotation aside; similar derivations may be found in most, if indeed not all, textbooks on machine learning, though they are often less than explicit in defining what domains the expectation values are taken over. For a refreshingly clear presentation, complete with pointing arrows and textboxes, the reader is referred to the set of lecture notes by Professor W. Cohen [3].

In practical terms, we use the bootstrap re-sampling to estimate the bias and variance of the model. This is achieved by replacing $\tilde{y}_i$ with $\tilde{y}_{i,B}$ in Eqn. 26. The expectation values will then be computed over all the bootstrap iterations. Furthermore, we are not able to extract the noise from the response values, and must then use the bias estimate $\mathrm{Bias}^2[y_i, \tilde{y}_{i,B}]$. Explicitly, the MSE, the bias and the variance estimates obtained from the bootstrap re-sampling become, for a test set with size n and a number M bootstrap iterations:

$$\mathrm{MSE}_B = \frac{1}{n}\sum_i^n \left(\frac{1}{M}\sum_B^M (y_i-\tilde{y}_{i,B})^2\right) \tag{27}$$

$$\mathrm{Bias}_B^2 = \frac{1}{n}\sum_i^n \left(y_i - \frac{1}{M}\sum_B^M \tilde{y}_{i,B}\right)^2 \tag{28}$$

$$\mathrm{Var}_B = \frac{1}{n}\sum_i^n \left(\frac{1}{M}\sum_B^M \left(\tilde{y}_{i,B}-\frac{1}{M}\sum_B^M \tilde{y}_{i,B}\right)^2\right) \tag{29}$$

E. The Franke Function

The Franke function is defined as

$$
\begin{aligned}
f(x,y) &= \frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)}{4}\right) \\
&+ \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y-21}{10}\right) \\
&+ \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)}{4}\right) \\
&- \frac{1}{5}\exp\left(-(9x-2)^2 - (9y-2)\right)
\end{aligned}
\tag{30}
$$

and will serve as function on which we test the performance of our models and build an understanding before we tackle the real terrain data, which we do not have direct control over in the same way. For reference sake, we have plotted the Franke function for values $[0,1] \times [0,1]$ which can be seen in Fig. 2.

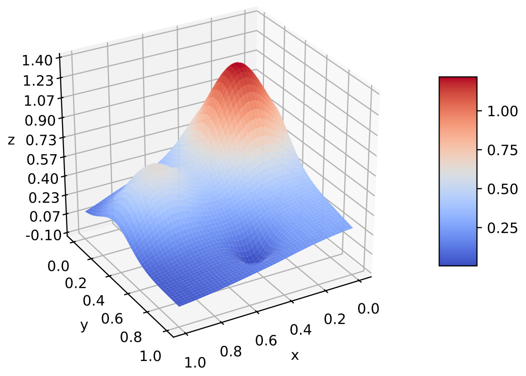


FIG. 2: The Franke Function

III. RESULTS & DISCUSSION

We aim to model both the Franke function and a set of terrain data as sums of polynomials in x and y. Our overarching goal is to determine which regression method performs the best (i.e. most faithfully reproduces the full data-set on which they are sampled) in both cases, and for which degree of the polynomial model and which value of the penalty parameter λ.

The terrain data, which has been downloaded from [4], is a collection of measured elevation heights at a discrete set of 3601×1801 pixel coordinates. For ease of interpretation and computation, we parameterize the pixel indices as uniformly spaced $x, y \in [0,1]$, and restrict ourselves to the first square of the terrain map, i.e. the first $[1801, 1801]$ pixels.

We first consider random samples of the Franke function where we analyze the performance of the different regression methods for various models (polynomial degrees) and penalty parameters. We then investigate the bias-variance behaviour of the methods, and the effect of the sampling size. Finally we compare various 'best predictions' for the different regression methods, in order to visualize the significance of their different approaches. These predictions are all trained on the same randomly sampled set of points in $x, y \in [0,1]$, and applied to a grid of 2000 points in each direction, with their resulting predictions being compared amongst the methods and with the ground truth of the Franke function evaluated on that grid.

Armed with insights from the investigation of the Franke function, we will subsequently do much of the same analysis for the terrain data. Ultimately we will compare the best predictions for each method (learned on the same down-sampled data set) with each other and with the ground truth of the full $[1801, 1801]$ terrain map we down-sampled from.

A. The Franke function

Throughout our analysis of the Franke function our focus was primarily on a data set consisting of $N = 500$ uniformly distributed random points, though we also made a brief analysis of the effects of varying the sample size. In particular, we present a limited analysis of a $N = 200$ sample in order to infer about the effects on which N has on the various regression methods.

1. Noise & confidence intervals

We start by looking at the behaviour of OLS limited to relatively modest complexities of polynomials up to the 5th order modeled from data sets consisting of $N = 500$ random samples of the Franke function with added random noise in the form $0.2\mathcal{N}(0,1)$. We also chose our confidence interval for the predictors as $\beta_i \pm 2\sigma_i$ in adherence with the literature as a standard choice [1], which corresponds to 95% of similarly sampled models falling within our range of uncertainty. Here, σ_i denotes the standard deviation of β_i computed as the root of the variance given by Eqn. 13. Furthermore, we also scale this and all subsequent datasets by first subtracting the mean of our response variable, then applying sci-kit learns [2] StandardScaler() functionality.

In Fig. 5 we see the size of each predictor β_i for a 5th order polynomial. Further, we also see the full size of confidence interval for each term for all the polynomial degrees $p = 1, \ldots, 5$.

We observe that the confidence intervals markedly increases as a function of the models complexity, as shown more explicitly in Fig. 6, which seems to suggest a $\sim$ quadratic trend. This implies that as we try to model our data on higher order polynomials, we will at some point run into the issue of the confidence interval of our predictors becoming an increasingly significant problem for the regression. We observe the manifestation of this problem more concretely as the complexity grows large in Fig.3, where we observe a rather significant rise in the MSE of the test set compared to the training set.

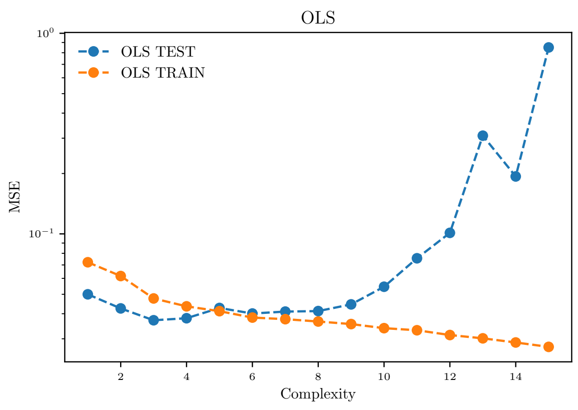


FIG. 3: The MSE for both the testing and the training data performed with OLS regression on $N = 500$ random samples of the Franke Function

We then investigated the effects that different magnitudes of noise had on OLS. In particular, we added random noise sampled from a weighted normal distribution $\delta \cdot \mathcal{N}(0, 1)$ for different weights δ. We also split our data set into a training and testing set with an 80/20 split using Sci-kit learn, where the R^2 score for each of the test sets over the different complexities are shown in Fig 4. We observe here that the regression becomes increasingly worse as δ approaches 0.5, and the noise begins to dominate.

2. *Resampling*

We then move on to our comparison of the various regression methods; OLS, Ridge and LASSO. In order to infer about their relative performance, we first computed their MSE using standard[1] 5-fold cross-validation for complexities $p = 1, \dots 15$ and in the case of Ridge & LASSO, 30 evenly spaced $\lambda \in [10^{-5}, 10^0]$. Then, for the optimal λ in each case we performed $M = 100$ bootstrap re-samplings for each of the models over all the complexities which we used to compute the MSE, as well as the MSE decomposed into the squared bias and variance. To specify, performed the bootstrap with a 80/20 train/test split, where the resamplings where performed on the training data and tested against the test data.

[1] As in; we did utilize a separate test/validation set in addition to the k-fold split

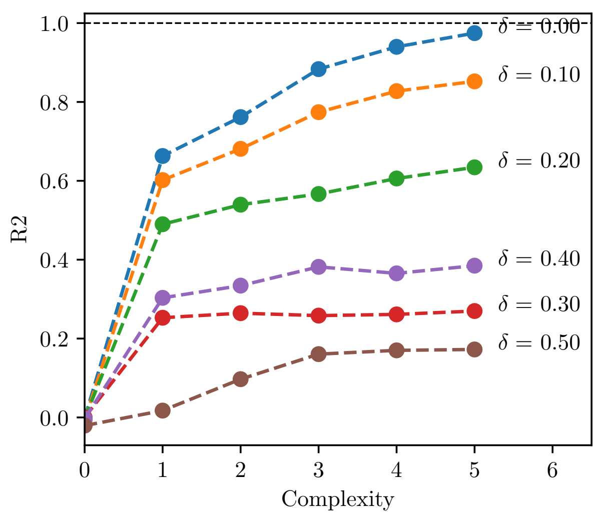


FIG. 4: R^2 measured for $N = 500$ random samples of the Franke function with added noise in the form $\delta \cdot \mathcal{N}(0, 1)$ modeled with complexities 0 to 5.

Here, we present the above analysis performed for two datasets consisting of $N = 200$ and $N = 500$ random samples of the Franke function. Additionally, we again added Gaussian noise sampled from $0.2 \cdot \mathcal{N}(0, 1)$ in both of the data sets.

As we saw in Fig. 3, once the complexity of our model reaches approximately the 8^{th} order, the MSE of our test set begins to grow rapidly. Looking then at Fig. 8d, we observe at the same complexity that the variance overtakes the square bias, and thus becomes the dominating term. Hence, the rapid growth of the MSE that we observed in Fig. 3 is an indication of the model being over-fitted to the data sample. This problem motivates the usage of Ridge and LASSO regression, which as we can see in Fig. 8d and Fig. 8f successfully suppresses the variance from becoming dominant for higher complexities. Furthermore, in both cases this is accomplished without affecting the bias of the model to a significant extent thus keeping the MSE relatively stable.

We turn then to Fig. 8a, 8b, 8c for the same analysis performed in a situation where the number of data samples has been limited to $N = 200$. Here, we observe the same problem but shifted to an lower complexity and to a much greater extent. Furthermore, the problem of overfitting is apparanlty much greater in this case. We may explain this by considering how the number of data points in our sample related to the total number of predictors in our model for each complexity. Using Eqn. 5, we may compute easily the number of predictors, which for convenience we list here in Table I for a range of complexities

Looking at Fig. 8a, we observe the variance becoming problematic at around the $5^{\text{th}} - 7^{\text{th}}$ order. Corresponding to 21-36 predictors respectively. Similarly, in the case of

TABLE I: Relationship between polynomial complexity and number of predictors for a 2D polynomial.

Complexity	3	4	5	6	7	8	9	10	11	12	13	14	15
No. Predictors	10	15	21	28	36	45	55	66	78	91	105	120	136

$N = 500$ the same problem begins to occur at the around the 8th order. Common to both of these cases are that the variance begins to increase rapidly once the number of predictors starts to approach $\sim 10\%$ the number of data points. This observation is supported by Harrell [5], in which limiting the complexity to $20\% - 10\%$ is proposed as a safe limit for the number of predictors, depending on the situation at hand.

We then move on to the $5-$fold cross-validation (CV), which is summarized for all the regression methods and both of the datasets in Fig. 7. Here, we note in particular for OLS that the CV yields a relatively low estimate of the MSE compared to the bootstrap. Looking more closely at the CV plots in Fig. 7, we observe the same phenomenon as we did with the bootstrap, where the MSE grows rapidly for OLS past a certain complexity. However, in this case we also get a much better insight of how Ridge and LASSO behaves wrt. λ. We observe that there is little change in the MSE for Ridge and Lasso over a wide range of λ values and polynomial degrees. Though not shown here, we have investigated and found that, with the exception of for very high λ-values, the bias and variance versus complexity for both ridge and lambda does not vary appreciably with λ, nor do the actual predictions resulting from those methods trained on our 500-point data set.

3. *Fitting the Franke Function*

B. The terrain data

Following our analysis of the Franke function, we performed a similar analysis of Terrain data found in [4]. In order to reduce the computational time required to process and analyze the data, we limited ourselves to a subsection of the full data set which was further down-sampled by only sampling every 40th pixel of the subsection. The full data set, as well as the down-sampled subset are both shown in Fig. 9.

Once again using k-fold cross-validation with $k = 5$ on our complete data set, we try to estimate what the optimal complexities and penalty parameters might be for the three methods. The results are shown in Fig. 11. We see once again that the MSE-scores remains quite consistent for Ridge and Lasso, and that there is little dependence on λ outside the very largest values. Here, the MSE for OLS is actually decreasing with complexity, until it blows up past a certain point.

A bootstrapped bias-variance analysis of a train-test split, with test/train ratio 0.2, is then performed, in order to see whether the exploding OLS MSE is due to variance and whether Ridge and Lasso manage to keep the variance down. Fig. 10 shows this to be the case. Ridge and LASSO do keep the variance in check, so that most of their errors are due to bias. For OLS we see that it performs quite well according to MSE, until the variance kicks in and leads to a rapid growth similar to what we observed for the Franke function.

IV. CONCLUSION

We have analyzed the performance of OLS, Ridge and LASSO regression on 'terrain-like' types of data, where the regression model was two-dimensional polynomials of varying degrees. We find that in order to qualitatively predict finer features of the terrain, only OLS is really useful; the other two methods smooth out the actual terrain variations in an effort to be stable against variance. The best qualitative description, for a given number of input sample points, is given by the highest degree OLS-solution where the variance remains insignificant. At higher degrees the variance becomes strong enough to completely mar the predictions, even though the MSE-score may remain deceptively low. Should the number of input data-points be so low that the highest "safe" OLS solution is such an exceedingly low degree as to be useless, then Ridge or Lasso at higher complexities may at least predict the main features of the terrain.

As a concluding observation, the authors conjecture that the use of a different type of basis functions might be appropriate for these kind of 'terrain-like' data sets. For instance orthogonal polynomials, or a Fourier-type basis set, could be global basis sets that allow Ridge and Lasso to be more discerning in their regularizations, than with the highly correlated polynomials that have been used in the present study. Or basis functions with compact support, which are only non-zero on limited domains, such as B-splines could provide more freedom to adapt to the actual variations in the terrains, without having over-fitting ramifications outside their support.

[1] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. (2009).

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

[3] W. W. Cohen, Bias-variance in machine learning.

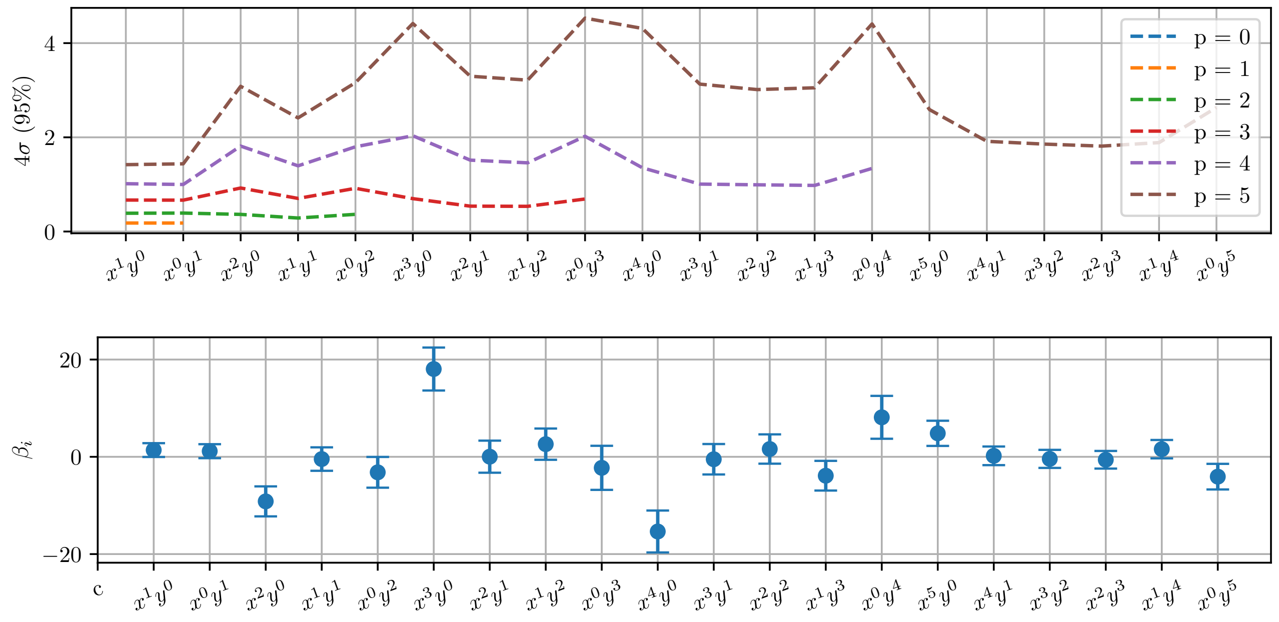


FIG. 5: Predictors βi for a 5th degree polynomial modeled with OLS on random samples of the Franke Function with noise $0.2 \cdot \mathcal{N}(0, 1)$

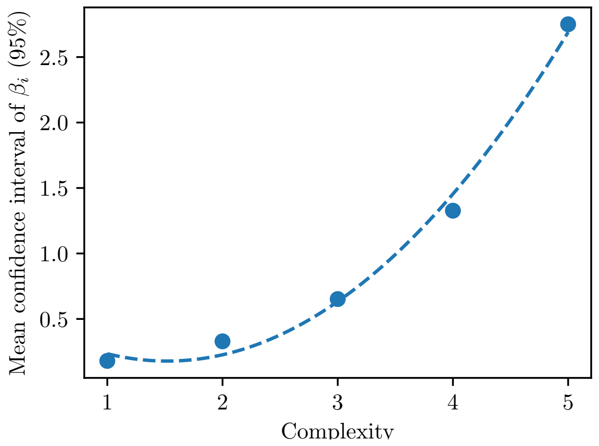
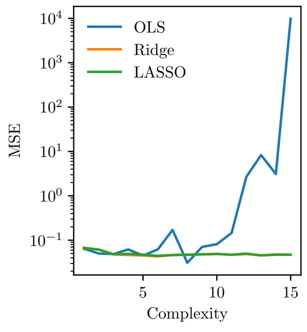
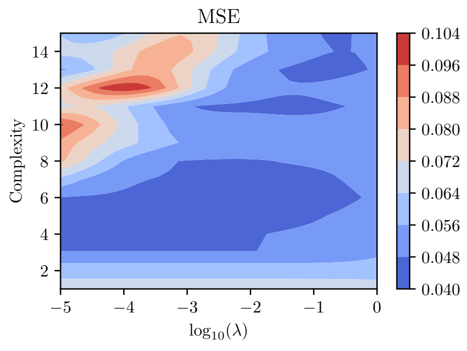


FIG. 6: Mean size of the 4σ (95%) confidence interval across all the predictors for polynomials up to the 5th order with OLS for $N = 500$ random samples of the Franke Function. Where the dashed line indicates 2nd order polynomial regression of the points

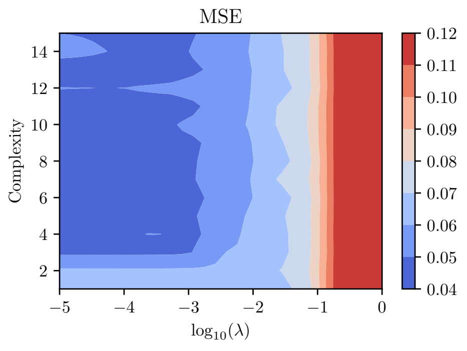
[4] Source of terrain data.
[5] F. Harrell, *Regression Modeling Strategies*, 2nd ed. (2015).



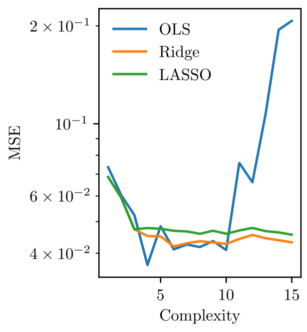
(a) OLS, N=200



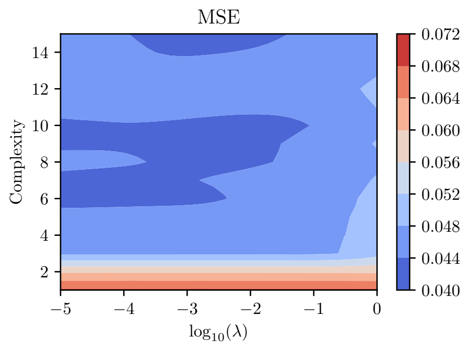
(b) Ridge, N=200



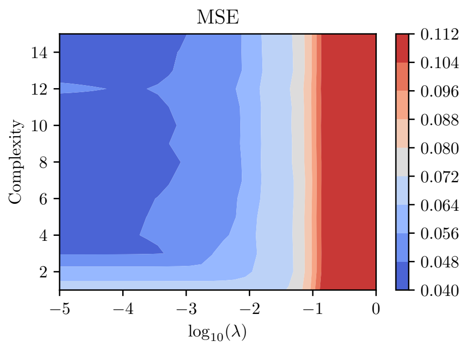
(c) LASSO, N=200



(d) OLS, N=500

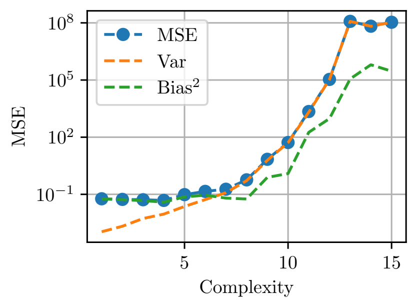


(e) Ridge, N=500

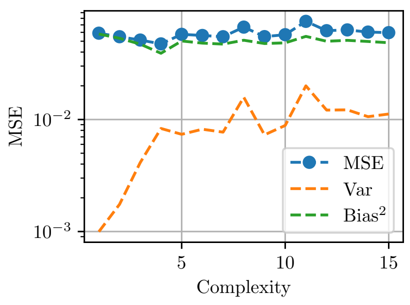


(f) LASSO, N=500

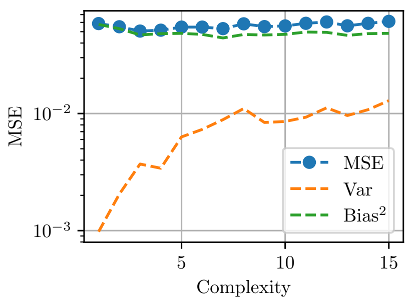
FIG. 7: MSE for $k = 5$ cross-validated Ridge & LASSO regression performed on N samples of the Franke Function with added random noise in the form $0.2 \cdot \mathcal{N}(0, 1)$



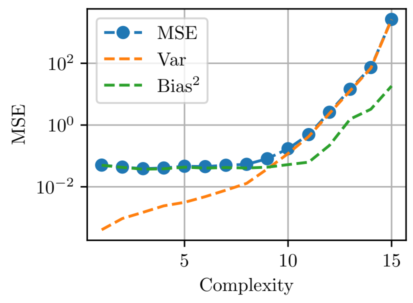
(a) OLS, N=200



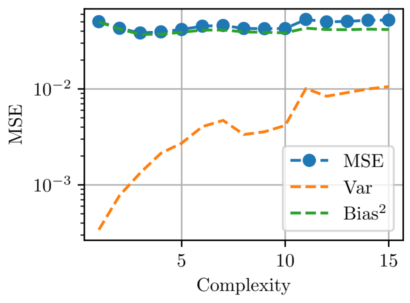
(b) Ridge, N=200



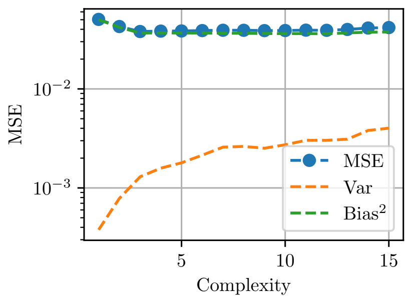
(c) LASSO, N=200



(d) OLS, N=500



(e) Ridge, N=500



(f) LASSO, N=500

FIG. 8: $M = 100$ Bootstrapped MSE, Bias^2 and Variance for OLS, Ridge and LASSO regression on N random samples of the Franke Function.

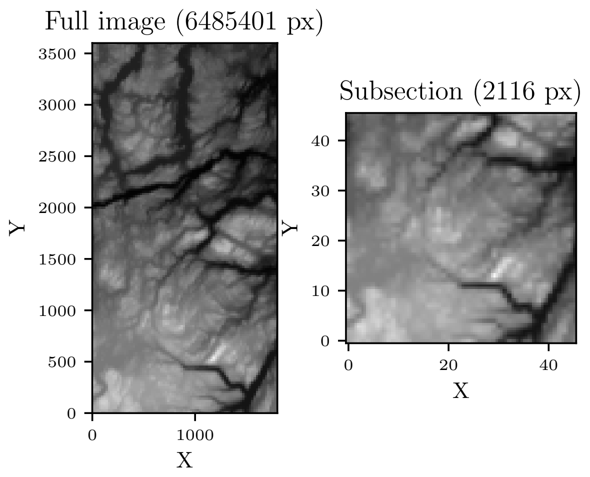


FIG. 9: The full terrain data as well as the down-sampled subsection which was used as the data set for our model.

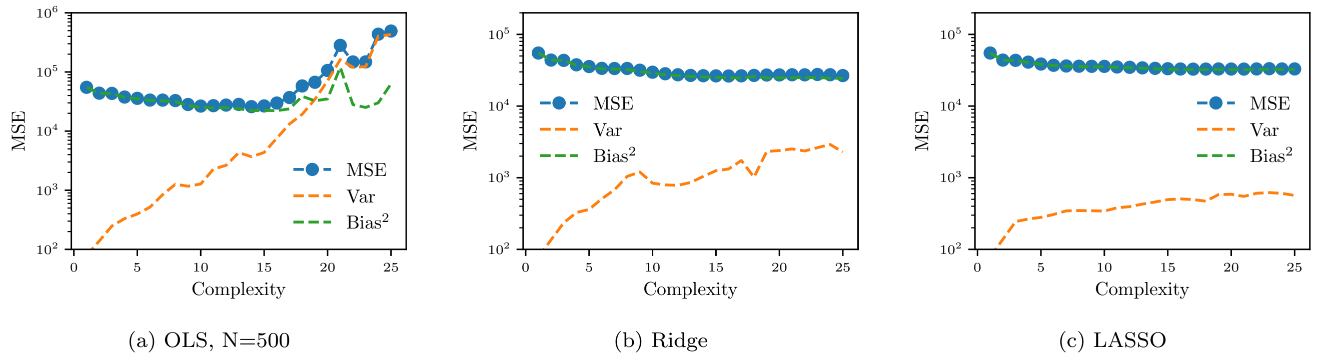


FIG. 10: $M = 100$ bootstrapped MSE for LASSO and Ridge regression on the Terrain data for the λ parameters yielding the best results for $k = 5$ fold cross-validation along with the $M = 100$ bootstrapped OLS.

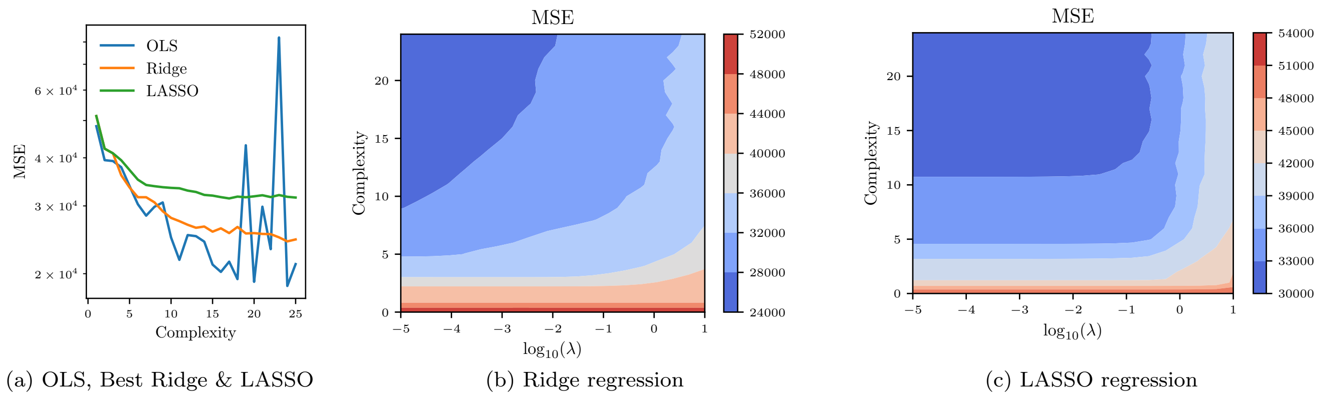


FIG. 11: MSE for $k = 5$ cross-validated Ridge & LASSO regression performed on the terrain data