Regression analysis and resampling methods

FYS-STK4155: Project 1

Morten Ledum & Håkon Kristiansen

 github.com/mortele/FYS-STK4155

October 8, 2018

Abstract

We parameterize digital terrain data using linear regression analysis algorithms: Ordinary least squares (OLS), Ridge regression, and Lasso regression. The bootstrap resampling technique is used to gauge the bias and variance of the models. We use basis sets of homogeneous monomials in two variables, up to and including total degree 5. We find that the ordinary least squares scheme produces the best fit overall, with the Lasso regression method coming in a distand second. Visualizing the bias-variance tradeoff reveals that we are most likely in a region of bias dominated MSE, with the variance having essentially no impact on the mean error. In order for the Ridge and Lasso regularizations to become more effective than the OLS scheme, we postulate that much higher model complexities than those achieved currently are needed (polynomials on the order of $p > 20$).

For initial validation of our models, we employ the test function of R. Franke.[1]

Contents

I. INTRODUCTION

Linear regression methods are fundamental in the field of statistical modelling. In the simplest variant of regression analysis—modelling a response dependent on some predictor variables—a linear ansatz is assumed for the relationship between response and predictors. The ansatz is linear in the sense of a linear combination of (possibly non-linear) functions of a single predictor. This linearity allows the matrix-vector equations resulting from minimizing the error between the observed data and the predicted model data to be swiftly solved, which is the main advantage of the method (and the main reason why linear regression is extensively used today).

In the modern society, data is ubiquitous and abundant. However, with an ever increasing volume of data available, drawing sensible conclusions about the relasionships between variables is highly non-trivial. Suppose you were given a complete set of statistics about every school district in Norway: The grade point average (and indeed individual grades) of every single student, the size and composition (in terms of nationalities, gender, height, weight, etc.) of every class, a full description of the teaching load of every teacher, as well as their personal data, educational history, and work history. Such a data set might be readily available, but drawing conclusions from it is not easy. Say e.g. that you wanted to know if increasing the number of teachers per pupil would result in overall better grades. Or—a more complex questions—what is the most efficient way to increase the overall grades of students?

Such questions pertaining to relationships between data of varying types are possible to answer using regression analysis, and bound on the statistical significance of the answers are possible to find. In the following, we will consider a much simpler toy problem as a simple introduction to the topic: Fitting a real valued function of two real variables. We will start out by considering the theory behind linear regression.

II. THEORY

In the following we briefly introduce the theory underlying the technical aspects of the present work. We begin by considering linear regression in general, and the ordinary least squares (OLS) method.

A. Linear regression

In order to introduce the least squares methods, we consider a case in which p characteristics of n samples are measured. The outcome, or the *response*, is denoted $\mathbf{y}$: a vector of size n. The measured characteristics, denoted the predictors, are organized in a matrix $\mathbf{X}$ of size $n \times p$. This is called the *design matrix*.

In regression analysis, we aim to explain the response in terms of the predictors, i.e. construct a function $\mathbf{y}(\mathbf{X})$. Assuming a linear relationship between $\mathbf{X}$ and $\mathbf{y}$ gives rise to *linear regression*, in which the response can be written as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{1}$$

where $\boldsymbol{\varepsilon}$ denotes the deviation of the linear model $\mathbf{X}\boldsymbol{\beta}$ and the response $\mathbf{y}$ and $\boldsymbol{\beta}$ is a parameter vector containing the linear regression coefficients β_i. The β_i variables are the unknowns in the linear regression problem, and they represent the partial derivative of the *modelled* response w.r.t. the descriptors.

In any non-trivial case, the error terms ε_i in the error vector $\boldsymbol{\varepsilon}$ will be non-zero. In this case, we regard our linear anzats as a *model* of the true response, the observed values y_i. We denote our model by $\tilde{\mathbf{y}}$, and define

$$\begin{aligned} \tilde{\mathbf{y}} &= \mathbf{X}\boldsymbol{\beta} \tag{2} \\ &= \mathbf{y} - \boldsymbol{\varepsilon}. \end{aligned}$$

The objective of linear regression thus emerges: Determine $\boldsymbol{\beta}$ in such a way that $\boldsymbol{\varepsilon}$ is minimized, thus giving a best possible linear fit of the response (minimizing the deviation $|\mathbf{y} - \tilde{\mathbf{y}}|$).

B. Ordinary least squares

In order to *minimize* the error $\boldsymbol{\varepsilon}$, we must define exactly what that means. We require a functional expression—commonly referred to as the *cost function*—and a metric in which to calculate it's size. Choosing the Euclidean L^2 norm ($\|\mathbf{v}\|_2 = \sqrt{\sum_i v_i^2}$) and the absolute value of $\boldsymbol{\varepsilon}$ as the metric and cost function, respectively, leads to the *ordinary least squares* (OLS) method. Defining the cost function,

$$\begin{aligned} C(\boldsymbol{\beta}) &= \|\mathbf{y} - \tilde{\mathbf{y}}\|_2^2 \\ &= \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \\ &= \sum_{i=1}^{n} \left| y_i - \beta_0 - \sum_{j=1}^{p} X_{ip}\beta_p \right|^2, \tag{3} \end{aligned}$$

we can formulate the OLS method as computing $\boldsymbol{\beta}_{\text{optimal}}$ by

$$\boldsymbol{\beta}_{\text{optimal}} = \arg\min_{\boldsymbol{\beta}}\big\{C(\boldsymbol{\beta})\big\}. \qquad (4)$$

In order to find $\boldsymbol{\beta}_{\text{optimal}}$, we may simply differentiate $C(\boldsymbol{\beta})$ w.r.t. $\boldsymbol{\beta}$ and enforce $\partial C(\boldsymbol{\beta})/\partial \boldsymbol{\beta} = 0$. Following Hastie, Tibshirani & Friedman,[2] we find that

$$\begin{aligned} \frac{\partial C(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \frac{\partial}{\partial \boldsymbol{\beta}}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \\ &= -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \overset{!}{=} 0 \\ \Rightarrow \mathbf{X}^T\mathbf{y} &= \mathbf{X}^T\mathbf{X}\boldsymbol{\beta} \\ \boldsymbol{\beta}_{\text{optimal}} &= \big(\mathbf{X}^T\mathbf{X}\big)^{-1}\mathbf{X}^T\mathbf{y}, \qquad (5) \end{aligned}$$

where we have written $\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$ as $(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$. We note that even though $\mathbf{X} \in \mathbb{R}^{n\times p}$ is a "large" matrix (assuming the number of observations $n \gg p$ the number of predictors per observation), the product $\mathbf{X}^T\mathbf{X} \in \mathbb{R}^{p\times p}$ is "small". Thus explicitly inverting $\mathbf{X}^T\mathbf{X}$ is not a problem on a modern computer.

We note that the *model prediction* may now be calculated simply as $\tilde{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta}_{\text{optimal}}$. This represents the optimal linear model subject under the Euclidean norm of the cost function as given in Eq. (3). This method was first rigorously described by Legendre in 1805.[3]

The design matrix

The design matrix, $\mathbf{X}$, can in principle contain any set of linearly independent functions of the predictors[1]. Every column in the design matrix corresponds to a mapping of the predictors, with elements $\mathbf{p}_i \mapsto \mathbf{X}_{ij}$. We will now consider an example of such a design matrix. We use two predictors—we will denote them x and y—with the response y. We introduce our model using the mappings $(x, y) \mapsto x$, $(x, y) \mapsto y$, and $(x, y) \mapsto xy$. Including also what is commonly referred to as the intercept, this gives rise to the design matrix

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_2 & y_2 & x_2y_2 \\ & & \vdots & \\ 1 & x_{n-1} & y_{n-1} & x_{n-1}y_{n-1} \\ 1 & x_n & y_n & x_ny_n \end{bmatrix}. \qquad (6)$$

For inputs (x_i, y_i) our model $\tilde{\mathbf{y}}$ now returns

$$\begin{aligned} \tilde{\mathbf{y}}_i &= \mathbf{x}_i^T\boldsymbol{\beta} \\ &= \beta_0 + \beta_1 x_i + \beta_2 y_i + \beta_3 x_iy_i, \qquad (7) \end{aligned}$$

where we used the shorthand notation $\mathbf{x}_i^T$ to denote $\text{Row}_i(\mathbf{X})$.

Before we continue describing the Ridge and Lasso regression schemes, we briefly introduce the basis sets used in this project.

C. Polynomial basis sets

Throughout the present work we employ a basis set of homogeneous monomials[2]. We will be working with 2D terrain data, and thus will need to consider monomials of up to and including two variables—x and y—in all possible homogeneous combinations. Disregarding the zero degree monomial, there are two possible such terms of degree up to and including one. These are simply x and y. Moving up to degree two, we must include x^2, y^2, and xy, for a total of five terms up to and including degree 2. Degree three adds an additional four terms: x^3, x^2y, xy^2, and y^3, and so on. In general, there are $n + 1$ such terms for monomials of degree n, namely

$$x^n,\ x^{n-1}y,\ x^{n-2}y^2,\ \ldots,\ xy^{n-1},\ \text{and}\ y^n.$$

The total basis sets of all such monomials of degree *up to and including* degree n—$\mathcal{B}_n$—thus contains

$$\text{size}(\mathcal{B}_n) \sum_{k=2}^{n+1} k = \frac{n(n+3)}{2} \qquad (8)$$

terms.

D. Ridge regression

As mentioned in section B, defining exactly what we mean by *minimizing the error* requires a cost function and a metric. The previous choice of $C(\boldsymbol{\beta}) = \|\mathbf{y} - \tilde{\mathbf{y}}\|_2^2$ is obviously not the only possible one. In fact, a possibly superior method may be devised by keeping the Euclidean L^2 norm, but including a term in the cost function which penalizes large values of β_i.

[1]We require linear independence to ensure the normal equations have a unique solution

[2]A homogeneous polynomial is a polynomial in which all terms have the same total degree, e.g. $xy+y^2+x^2$ is a homogeneous polynomial, x is a homogeneous monomial, while $xy + x^3$ is *not*. Monomials are simply polynomials with only a single term.

Such an approach was first used in statistics by Hoerl & Kennard,[4] but was proposed already in the 1940s by Andrey Tikhonov.[5] Taking the cost function to be

$$C_{\mathrm{T}}(\boldsymbol{\beta}) = \|\mathbf{y} - \tilde{\mathbf{y}}\|_2^2 + \|\boldsymbol{\Gamma}\boldsymbol{\beta}\|_2^2 \qquad (9)$$

gives rise to a regressions scheme known as Tikhonov regularization. The Tikhonov matrix $\boldsymbol{\Gamma}$ governs the form of the regularization term. A simple case of $\boldsymbol{\Gamma} = \sqrt{\lambda}\mathbb{1}$, favoring solutions with small (in the L^2 norm sense) values of the parameters $\boldsymbol{\beta}$, results in the *Ridge regression* of Hoerl & Kennard. The $\lambda \geq 0$ parameter here represents a tuneable penalty for large $\boldsymbol{\beta}$ values. We note that $\lambda = 0$ recovers the OLS method of section B.

Writing out the cost function of Ridge regression, we find that

$$\begin{aligned}
C_{\mathrm{R}}(\boldsymbol{\beta}) &= \|\mathbf{y} - \tilde{\mathbf{y}}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 \\
&= \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 \\
&= \sum_{i=1}^{n} \left| y_i - \beta_0 - \sum_{j=1}^{p} X_{ip}\beta_p \right|^2 + \lambda \sum_{j=1}^{p} \beta_j^2,
\end{aligned}$$
$$(10)$$

where we have left out the intercept β_0 from the regularization term. This is done to ensure the solutions do not explicitly depend on the zero point chosen for $\mathbf{y}$, i.e. adding a constant to each response value y_i would not result in a simple shift of the predictions by the same amount.[2]

In the same way as before, we may differentiate the cost function and enforce $C_{\mathrm{R}}(\boldsymbol{\beta}) = 0$ in order to find the optimal $\boldsymbol{\beta}$. We obtain

$$\begin{aligned}
\frac{\partial C_{\mathrm{R}}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \frac{\partial}{\partial \boldsymbol{\beta}}\Big[(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\mathbb{1}\boldsymbol{\beta}^T\boldsymbol{\beta}\Big] \\
&= -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + 2\lambda\mathbb{1}\boldsymbol{\beta} \overset{!}{=} 0
\end{aligned}$$
$$\Rightarrow \mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{X}\boldsymbol{\beta} + \lambda\mathbb{1}\boldsymbol{\beta}$$
$$\boldsymbol{\beta}_{\mathrm{optimal}}^{\mathrm{R}} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbb{1}\right)^{-1}\mathbf{X}^T\mathbf{y}. \qquad (11)$$

Singular-value decomposition

Before we continue, we introduce briefly the singular-value decomposition (SVD). Note carefully that *any* $m \times n$ matrix $\mathbf{A}$ can be decomposed into a product like this.

Let $\mathbf{A}$ be an $m \times n$ matrix with rank r. Then there exists an $m \times n$ matrix $\boldsymbol{\Sigma}$ in which the first r diagonal entries are the singlar values of $\mathbf{A}$, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r >$ 0, and there exist and $m \times m$ orthogonal matrix $\mathbf{U}$ and an $n \times n$ orthogonal matrix $\mathbf{V}$ such that

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T. \qquad (12)$$

All entries in $\boldsymbol{\Sigma}$ outside of the first r diagonal elements are zero. The singular values of $\mathbf{A}$ denote the square roots of the eigenvalues of $\mathbf{A}^T\mathbf{A}$.[6]

The (first r columns of the) matrix $\mathbf{U}$ contains the eigenvectors of $\mathbf{A}^T\mathbf{A}$ and represents an orthonormal basis for the column space of $\mathbf{A}$, $\mathrm{Col}\,\mathbf{A}$. The (first r columns of the) matrix $\mathbf{V}$ contains the eigenvectors of $\mathbf{A}\mathbf{A}^T$ and represents an orthonormal basis for the row space of $\mathbf{A}$, $\mathrm{Row}\,\mathbf{A}$. The remaining $m - r$ and $n - r$ columns of $\mathbf{U}$ and $\mathbf{V}$ form orthonormal bases for $\mathrm{Nul}\,\mathbf{A}$ and $\mathrm{Nul}\,\mathbf{A}^T$.

Thus we can interpret the SVD loosely as finding the orthonormal bases of $\mathrm{Col}\,\mathbf{A}$ and $\mathrm{Row}\,\mathbf{A}$ such that application of $\mathbf{A}$ maps $\mathbf{v}_i \mapsto \sigma_i\mathbf{u}_i$.

Ridge regression using the SVD

If we consider the SVD of $\mathbf{X}$, $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$, and compute $\mathbf{X}^T\mathbf{X}$ we find that

$$\begin{aligned}
\mathbf{X}^T\mathbf{X} &= \left(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\right)^T\left(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\right) \\
&= \mathbf{V}\boldsymbol{\Sigma}^T\mathbf{U}^T\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T \\
&= \mathbf{V}\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}\mathbf{V}^T = \mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^T, \qquad (13)
\end{aligned}$$

where the orthogonality of $\mathbf{U}$ made $\mathbf{U}^T\mathbf{U} = \mathbb{1}$. Inserting this into the expression for the optimal $\boldsymbol{\beta}_{\mathrm{optimal}}^{\mathrm{R}}$ (Eq. (11)) yields[2]

$$\begin{aligned}
\boldsymbol{\beta}_{\mathrm{optimal}}^{\mathrm{R}} &= \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbb{1}\right)^{-1}\mathbf{X}^T\mathbf{y} \\
&= \left(\mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^T + \lambda\mathbb{1}\right)^{-1}\mathbf{X}^T\mathbf{y} \\
&= \left(\mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^T + \lambda\mathbb{1}\mathbf{V}\mathbf{V}^T\right)^{-1}\mathbf{X}^T\mathbf{y} \\
&= \left(\mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^T + \lambda\mathbf{V}\mathbb{1}\mathbf{V}^T\right)^{-1}\mathbf{X}^T\mathbf{y},
\end{aligned}$$

where we multiplied by $\mathbb{1} = \mathbf{V}\mathbf{V}^T$ (recall that $\mathbf{V}^{-1} = \mathbf{V}^T$ due to orthogonality) and used the fact that the identity matrix commutes with any

other matter. Furthermore, we find

$$\beta_{\text{optimal}}^{\text{R}} = \left(\mathbf{V}\left[\mathbf{\Sigma}^2 + \lambda\mathbb{1}\right]\mathbf{V}^T\right)^{-1}\mathbf{X}^T\mathbf{y}$$
$$= \mathbf{V}\left[\mathbf{\Sigma}^2 + \lambda\mathbb{1}\right]^{-1}\mathbf{V}^T\mathbf{X}^T\mathbf{y} \qquad (14)$$
$$= \mathbf{V}\left[\mathbf{\Sigma}^2 + \lambda\mathbb{1}\right]^{-1}\mathbf{V}^T\left(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\right)^T\mathbf{y}$$
$$= \mathbf{V}\left[\mathbf{\Sigma}^2 + \lambda\mathbb{1}\right]^{-1}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}$$
$$= \mathbf{V}\left[\mathbf{\Sigma}^2 + \lambda\mathbb{1}\right]^{-1}\mathbf{\Sigma}\mathbf{U}^T\mathbf{y}, \qquad (15)$$

where we used that $\mathbf{\Sigma}$ is diagonal, so $\mathbf{\Sigma}^T = \mathbf{\Sigma}$. Note that since $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$, we can rewrite the inverse product

$$\left(\mathbf{V}\left[\mathbf{\Sigma}^2 + \lambda\mathbb{1}\right]\mathbf{V}^T\right)^{-1} = \left(\mathbf{V}^T\right)^{-1}\left[\mathbf{\Sigma}^2 + \lambda\mathbb{1}\right]^{-1}\mathbf{V}^{-1}$$
$$= \mathbf{V}\left[\mathbf{\Sigma}^2 + \lambda\mathbb{1}\right]^{-1}\mathbf{V}^T,$$

as was done in Eq. (14). Since we are now taking the inverse of a diagonal matrix, we can simply write out the terms. Note that the inverse will itself be diagonal, and given by

$$\left(\left[\mathbf{\Sigma}^2 + \lambda\mathbb{1}\right]^{-1}\right)_{ii} = \frac{1}{\sigma_{ii}^2 + \lambda}. \qquad (16)$$

Rewriting the OLS scheme in terms of the SVD yields a very similar

$$\beta_{\text{optimal}} = \mathbf{V}\left(\mathbf{\Sigma}^2\right)^{-1}\mathbf{V}^T\mathbf{\Sigma}^T\mathbf{U}^T y. \qquad (17)$$

E. Lasso regression

As mentioned repeatedly, we are free to choose what we mean by *minimizing the error* w.r.t. what metric and what cost function we use. Whereas the Ridge regression of section D used L^2 regularization by adding a $\lambda\|\boldsymbol{\beta}\|_2^2$ term to $C(\boldsymbol{\beta})$, we may instead try a L^1 regularization. This constitutes setting up the cost function as

$$C_{\text{L}}(\boldsymbol{\beta}) = \|\mathbf{y} - \tilde{\mathbf{y}}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1$$
$$= \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1$$
$$= \sum_{i=1}^{n}\left|y_i - \beta_0 - \sum_{j=1}^{p}X_{ip}\beta_p\right|^2 + \lambda\sum_{j=1}^{p}|\beta_j|^2. \qquad (18)$$

Originally popularized by Tibshirani,[7] the *Lasso regression* has certain potential advantages over the OLS and Ridge regression schemes. Most notably, the Lasso can perform *variable selection*, i.e. some β_js may be identically zero as a result of the minimization. The name Lasso is short for "least absolute shrinkage and selection operator".

Computing the derivative of the Lasso cost function yields

$$\frac{\partial C_{\text{L}}(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_j} = \frac{\partial C_{\text{OLS}}(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_j} + \lambda\sum_{j=1}^{p}\frac{\partial}{\partial\beta_j}|\beta_j|$$
$$= \frac{\partial C_{\text{OLS}}(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_j} + \lambda\frac{\beta_j}{\sqrt{\beta_j^2}}$$
$$= \frac{\partial C_{\text{OLS}}(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_j} + \lambda\,\text{sgn}(\beta_j) \stackrel{!}{=} 0. \qquad (19)$$

In general, this can not be directly solved for $\beta_{\text{optimal}}^{\text{L}}$ as in the case of OLS or the Ridge scheme. Under the assumption that $\mathbf{X}$ is orthogonal, an explicit solution exists and is given by[8]

$$\left(\beta_{\text{optimal}}^{\text{L}}(\lambda)\right)_j = \text{sgn}(\beta_j^{\text{OLS}})\left(|\beta_j^{\text{OLS}}| - \lambda\right)_+, \qquad (20)$$

where $(\cdot)_+$ represents the positive part of $\cdot$. In the general case, an iterative solver must be used to compute $\beta_{\text{optimal}}^{\text{L}}$.

F. Principal components

The following section follows Hastie, Tibshirani & Friedman.[2]

Recall from section that the SVD matrix $\mathbf{U}$ represents an orthonormal basis for the column space of the decomposed matrix $\mathbf{A}$. If we consider the prediction resulting from OLS, and perform a SVD of $\mathbf{X}$ we find that

$$\tilde{\mathbf{y}}_{\text{OLS}} = \mathbf{X}\beta_{\text{OLS}}$$
$$= \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$
$$= \mathbf{X}\left[\left(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\right)^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\right]^{-1}\mathbf{X}^T\mathbf{y}$$
$$= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\left[\mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T\right]^{-1}\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}$$
$$= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\left[\mathbf{\Sigma}^2\right]^{-1}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}$$
$$= \mathbf{U}\mathbf{\Sigma}\left[\mathbf{\Sigma}^2\right]^{-1}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}$$
$$= \mathbf{U}\mathbf{\Sigma}\left[\mathbf{\Sigma}^2\right]^{-1}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}$$
$$= \mathbf{U}\mathbf{U}^T\mathbf{y}. \qquad (21)$$

A similar derivation for $\tilde{\mathbf{y}}_{\text{R}}$ for the Ridge regres-

sion yields

$$\tilde{\mathbf{y}}_{\mathrm{R}} = \mathbf{X}\boldsymbol{\beta}_{\mathrm{R}}$$
$$= \mathbf{X}\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbb{1}\right)^{-1}\mathbf{X}^T\mathbf{y}$$
$$= \mathbf{U}\boldsymbol{\Sigma}\left(\boldsymbol{\Sigma}^2 + \lambda\mathbb{1}\right)^{-1}\boldsymbol{\Sigma}\mathbf{U}^T\mathbf{y}$$
$$= \sum_{j=1}^{p}\mathbf{u}_j\frac{\sigma_j^2}{\sigma_j^2 + \lambda}\mathbf{u}_j^T\mathbf{y}, \qquad (22)$$

where σ_j denotes the diagonal elements of the diagonal matrix $\boldsymbol{\Sigma}$, i.e. $(\boldsymbol{\Sigma})_{jj} = \sigma_j$. Note now that $\mathbf{U}^T\mathbf{y}$ are the coordinates of $\mathbf{y}$ w.r.t. the orthonormal basis $\mathbf{U}$. Comparing Eqs. (21) and (22), we note that the coordinates of $\mathbf{y}$ in both cases are computed in the orthonormal basis of Col $\mathbf{X}$ (as specified by the SVD matrix $\mathbf{U}$), but the Ridge scheme also *shrinks* the coordinates. The shrinkage is large whenever σ_j^2 is small. Recalling that $\mathbf{X}^T\mathbf{X} = \mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^T$ is an eigendecomposition of $\mathbf{X}^T\mathbf{X}$, with $\mathbf{V}$ containing the eigenvectors, we denote these eigenvectors $\mathbf{v}_j$ to be the *principal components* of $\mathbf{X}$. The eigenvalues contained in $\boldsymbol{\Sigma}^2$ are precicely the proportionality factors, σ_j^2, involved in the shrinkage.

The diagonalization of $\mathbf{X}^T\mathbf{X}$ constitutes a co-ordinate transform into a coordinate system in which $\mathbf{X}^T\mathbf{X}$ itself is diagonal. The matrix $\mathbf{X}^T\mathbf{X}$ is the covariance matrix (apart from a constant factor $1/N$). In the orthonormal basis of $\mathbf{V}$, the covariance matrix is diagonal and contains the variances σ_j^2 of these linear combinations of the columns of $\mathbf{X}$.

The first principal component direction has the property that $\mathbf{z}_1 = \mathbf{X}\mathbf{v}_1$ has the largest variance of all the linear combinations of the columns of $\mathbf{X}$. In general, the principal components are ordered such that $\mathrm{Var}\,\mathbf{z}_1 \geq \mathrm{Var}\,\mathbf{z}_2 \geq \cdots \geq \mathrm{Var}\,\mathbf{z}_N$.

In essence, the first principal component represent the direction in Col $\mathbf{X}$ in which the variance is highest, the second principal component represents the corresponding direction in which the variance is highest, apart from the first, and so on. It is clear that the Ridge regression scheme simply rotates the OLS solution into the principal components of the design matrix, and then shrinks the coefficients corresponding to low-variance components.

G. Assessing model accuracy

The goal in machine learning is to use a chosen model to make predictions. In particular, we need to quantify how close a predicted response value for a given observation is to the true response value. In the regression setting, a common measure is the *mean squared error* (MSE), given by

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2, \qquad (23)$$

where $\hat{f}(x_i)$ is the prediction for the ith observation y_i. Computing the MSE on the training set result in what we could call the training MSE and gives a measure of how well the model fits the training data. However, we are mainly interested in how good predictions are on previously unseen data.

Suppose that we have set of test data, $\{x_i, y_i\}$, which is not used during training. Then we could compute the MSE on the test set, giving a measure on how well the method works on unseen data. We would like to select the method which has the lowest test MSE.

The challenge now is how do we estimate the test MSE? Of course, if test data are available we just compute. The problem is that test data are not available in general. So if do not have test data at hand, a natural approach seems to choose the method that minimizes the training MSE. However, there is no guarantee that low training MSE results in the lowest test MSE. As an extreme example suppose you draw a curve that passes through every point in a data set, this would result in perfect train MSE. If you try to use this model on another data set where I move one datapoint significantly it would result in a high test MSE.

One can show that the expected test MSE, for a given value x_0, can always be decomposed as

$$E[(y_0 - \hat{f}(x_0))]^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + \mathrm{Var}(\epsilon). \qquad (24)$$

Here $E[(y_0 - \hat{f}(x_0))]^2$ defines the expected test MSE, and refers to the average test MSE that we would obtain if we repeatedly estimated f using a large number of training sets, and tested each at x_0.

The variance, $\mathrm{Var}(\hat{f}(x_0))$, measures how much $\hat{f}$ would change if we trained the model on a different data set. Ideally $\hat{f}$ should not vary to much between training sets. Generally more flexible methods have higher variance, which can give large deviations in $\hat{f}$ when changes in the training set is small.

Bias measures the the error introduced by assuming for example a linear relationship between the response $\mathbf{y}$ and the predictors $\mathbf{X}$. Thus, if the true relation is non-linear this will result in bias. In particular, if the data is highly non-linear it would be impossible to produce accurate estimates by assuming a linear relationship between the response and predictors. Generally more flexible methods will have less bias.

Eq. (24) suggests that we want both low variance and low bias in order to minimize the expected test error. The above discussion implies that these are conflicting interests, in the sense that increasing flexibility of a model reduces bias, but increases variance. This phenomena is referred to as the *bias-variance trade-off*.

In practice, with test data unavailable, it is generally not possible to compute the test MSE, bias or variance for a specific method. Thus we need tools to estimate the test MSE (or other quantities of interest) from training data. Two such methods are cross-validation and the *bootstrap*, which belongs to a class of methods called *resampling* methods.

H. Resampling methods

Resampling methods is an important tool in statistics. The idea is to repeatedly draw samples from a training set and refit the model on each sample in order to gain additional information about the fitted model.

One drawback of resampling methods is that they can be computationally expensive, since they have to fit the same model multiple times using different subsets of the training data. In particular, if there is a large number of samples or if each instance of training is expensive this can be time consuming. However, the computational requirement of resampling methods are in general not prohibitive. Two of the most commonly used resampling methods are cross-validation and the *bootstrap*.

Cross-validation can be used to estimate the test error in order to assess the performance of a particular model, or to select the suitable level of flexibility. Evaluating a model's performance is known as *model assessment*, while selecting the proper level of flexibility for a model is known as *model selection*. The bootstrap is commonly used to provide a measure of accuracy of a parameter estimate.

I. The *bootstrap*

The bootstrap method is a statistical technique for estimating quantities about a population by taking averages over estimates from several small data samples. Samples are constructed by drawing observations from a large data sample -in our case the training set- with replacement. The bootstrap method estimates a quantity of a population by taking small samples repeatedly, calculating the statistic, and taking the average of the calculated statistics. In particular we use the bootstrap to estimate the MSE, bias and variance which will give an indication of the appropriate order of complexity for a given model.

III. DATA SETS

We are chiefly interested in parametrizing digital terrain data. However, in order to test and validate our implementation of the regression model and the resampling technique, we employ the Franke function[1] as a test case before considering real data.

J. The Franke function

The test function of Franke—originally developed to test and rate different surface interpolation techniques—is "a surface with a variety of behaviour" which consists of "two Gaussian peaks and a sharper Gaussian dip superimposed on a surface sloping towards the first quadrant."[1] It is noted by Franke in the his original paper that the slope was introduced mainly as a visual aid and presumably had little impact on the actual interpolations performed.

More specifically, the Franke function $f_{\mathrm{F}}(x, y)$ takes the full form

$$
\begin{aligned}
f_{\mathrm{F}}(x, y) =\ & \frac{3}{4} \exp\left\{\frac{-1}{4}\left[(9x-2)^2 + (9y-2)^2\right]\right\} \\
& + \frac{3}{4} \exp\left\{\frac{-1}{49}(9x+1)^2 + \frac{1}{10}(9y+1)^2\right\} \\
& + \frac{1}{2} \exp\left\{\frac{-1}{4}\left[(9x-7)^2 + (9y-3)^2\right]\right\} \\
& - \frac{1}{5} \exp\left\{\frac{-1}{4}\left[(9x+4)^2 + (9y-7)^2\right]\right\}.
\end{aligned}
\tag{25}
$$

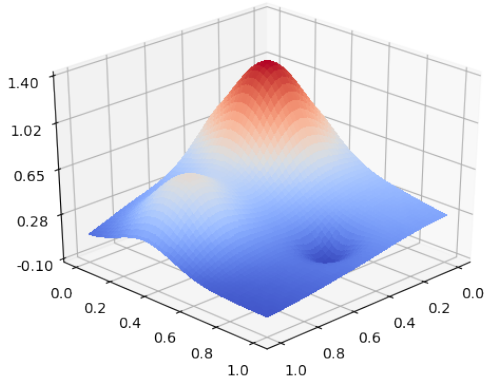A plot of the $f_{\mathrm{F}}(x, y)$ surface can be seen in Fig. 1.

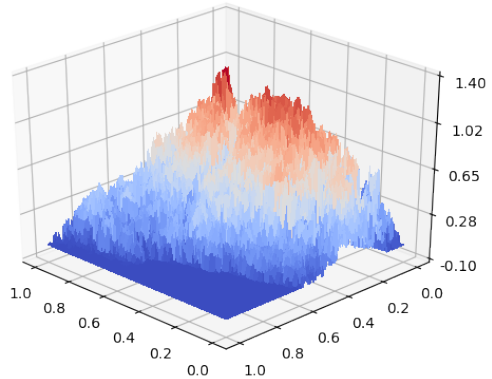FIG. 1. The Franke test function plotted for $0 \leq x, y \leq 1$.

FIG. 2. The terrain data in use in the present work, taken from the Møsvatn Austfjell area in the municipality of Tinn in Telemark county, Norway. Retrieved using the USGS EarthExplorer website. The height data is scaled to fit in $0 \leq z \leq 1$, and the reference zero point is set to zero.

K. Terrain data

The terrain data used is taken from the U.S. Department of the Interior U.S. Geological Survey's (USGS) EarthExplorer[3] website. The USGS stores data from the Shuttle Radar Topography Mission (SRTM) which maps the earth's land surface topology with a resolution of 1 arc-second (about 30 m). We will use SRTM data taken from the EarthExplorer website as the basis for our terrain parametrization.

The specific terrain data we will use in the present project is taken from the Møsvatn Austfjell area in the municipality of Tinn in Telemark county, Norway. A visual representation of the data is shown in Fig. 2.

IV. RESULTS AND DISCUSSION

L. Verification of the models: The Franke function

Our first tests comprise simple OLS fitting of the Franke function described in section J. We perform the fits using the scheme described in section B. Basic testing is done automatically using the `pytest` unit test framework.[4] These ensure e.g. that the fits performed by the manual matrix inversion are identical to the ones performed automatically by sci-kit learn, that intercepts and polynomial coefficients are fitted correctly when the underlying functional form is itself a polynomial of known order, etc. In ad-

dition to the small scale automatic testing, we perform more thorough, extensive testing on the results of the fittings. First off we visualize and compare the performed OLS fits to the underlying Franke function. These are shown in Fig. 3 (the Franke function, for reference, is shown in Fig. 1). We note that, as expected, higher degree polynomials appear to recreate the underlying exponentials better than lower degree polynomials.

The fact that higher order polynomials are better able to reproduce the Franke function is in itself not surprising. Remember that Franke formed his function from a sum of exponentials. The power series expansion of this function is thus just a sum over all possible monomials in x and y with prefactors proportional to $1/n!$.

The resampling scheme described in section H, we compute the variance of the computed β coefficients. These are shown in Table I for all the different polynomial degrees used, ranging from $p = 2$ to $p = 5$. We note that increasing the polynomial degrees apparently tends to increase the variance of the computed β coefficients. This is in line with the statement of the bias-variance tradeoff: Higher model complexity leads to an increased sensitivity to small fluctuations in the training set, and the variance goes up. This is illustrated in Fig. 5, where the mean values of the variances of all β parameters corre-

---

[3]EarthExplorer website: https://earthexplorer.usgs.gov/.

[4]All tests can be run automatically by calling `pytest -v` from anywhere within the Github resository

TABLE I. Parameters β and their bootstrap computed variance $\sigma^2(\beta)$ for the OLS fits of the Franke function, shown in Fig. 3. Pair wise columns represent β and $\sigma^2(\beta)$ for each polynomial degree p used. Each row shows the β_j coefficient and $\sigma^2(\beta)$ for the corresponding monomial$_j$ term.

| p	2		3		4		5	
	β	$\sigma^2(\beta)$	β	$\sigma^2(\beta)$	β	$\sigma^2(\beta)$	β	$\sigma^2(\beta)$
1	1.176 06	0.000 07	0.994 32	0.000 14	0.607 14	0.000 06	0.379 20	0.000 52
x	−1.065 33	0.000 47	−0.603 90	0.003 14	4.231 95	0.004 78	8.057 87	0.039 89
y	−0.741 96	0.000 41	1.367 10	0.002 06	3.188 95	0.003 99	3.773 85	0.033 82
x^2	0.120 32	0.000 26	−1.401 95	0.008 85	−19.382 16	0.055 41	−35.011 95	0.520 45
xy	0.875 01	0.000 25	2.043 06	0.005 97	−2.283 49	0.028 25	−15.642 35	0.500 91
y^2	−0.367 20	0.000 29	−6.659 28	0.005 85	−12.536 72	0.044 34	−8.308 69	0.440 47
x^3			0.891 77	0.002 88	25.256 28	0.107 94	49.218 54	1.617 46
x^2y			0.362 61	0.002 19	8.104 44	0.056 90	45.875 08	1.703 97
xy^2			−1.506 62	0.001 88	1.405 54	0.052 42	21.251 27	1.474 19
y^3			4.696 44	0.002 07	12.615 20	0.084 71	−9.030 64	1.506 07
x^4					−10.841 36	0.025 48	−24.135 09	1.257 84
x^3y					−5.151 14	0.019 48	−54.812 83	1.391 24
x^2y^2					0.007 64	0.014 37	−8.235 81	1.141 34
xy^3					−1.905 24	0.020 13	−30.191 86	1.123 19
y^4					−3.511 79	0.020 32	30.201 66	1.282 60
x^5							1.538 82	0.160 41
x^4y							19.535 04	0.185 67
x^3y^2							10.839 52	0.170 47
x^2y^3							−5.299 38	0.159 35
xy^4							16.912 43	0.150 35
y^5							−16.843 69	0.169 20
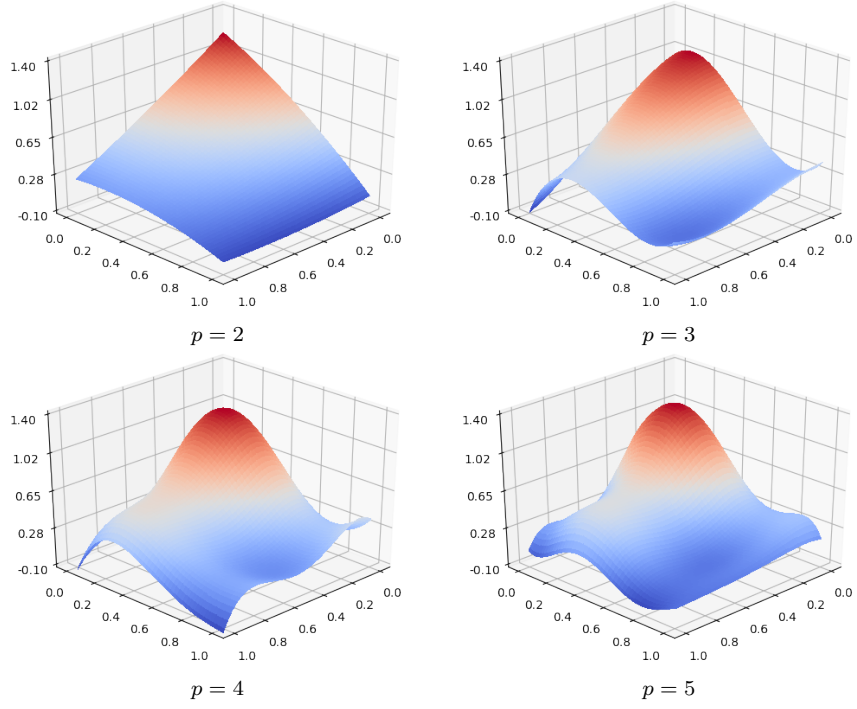
FIG. 3. Ordinary least squares fits, using data from the Franke function with polynomials of degree 2, 3, 4, and 5. The p parameter indicates what order of polynomials are used. The target function of Franke can be seen in Fig. 1.
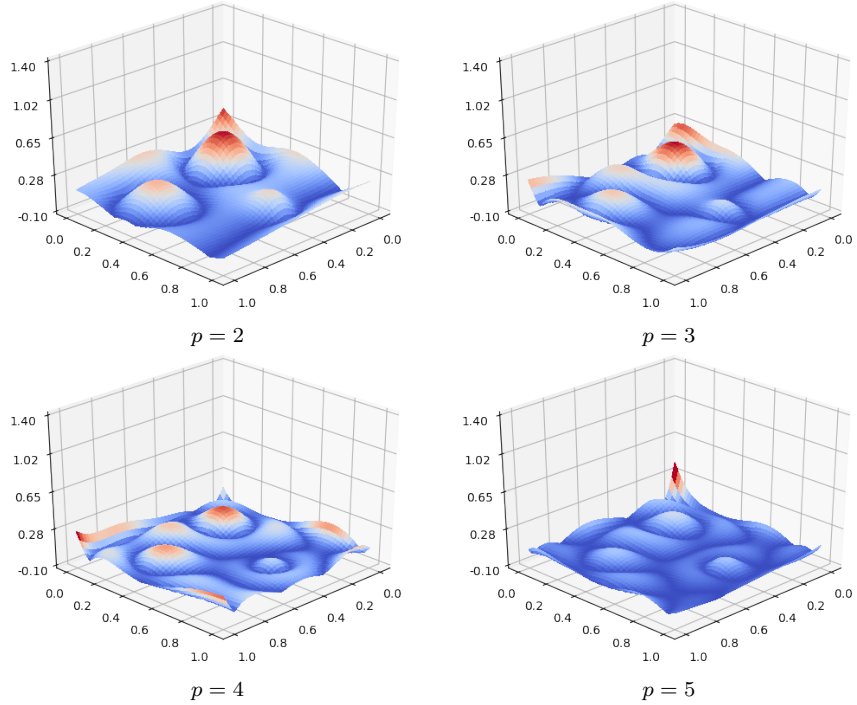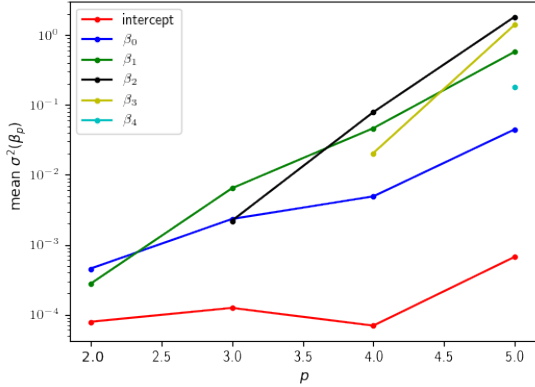

FIG. 4. The absolute difference between the ordinary least squares fits of Fig. 3 and the true data, the Franke function. Polynomials of degree 2, 3, 4, and 5 have been used in the fitting. The p parameter indicates what order of polynomials are used.

FIG. 5. Mean values of the variances of the β coefficients resulting from OLS fitting, e.g. mean $\sigma^2(\beta_2)$ is the mean of all βs corresponding to monomials of total degree two, i.e. β_{x^2}, β_{xy}, and β_{y^2}. We note that as the overall complexity of the model increases (increasing p), the variance of each family of βs tends to increase.

sponding to a total monomial degree is plotted for each model.

The absolute difference between the fitted models and the Franke function is shown in Fig. 4. We note that the magnitude of the error decreases as the polynomial order increases, but the error for higher order polynomial fits exhibit faster oscillating behaviour.

The mean squared error, introduced in section G gives a measure of the accuracy of each model in a single number, the expectation value of the squared difference between each model prediction and the true underlying function value at each data point. We compute the MSE and the *coefficient of determination*, the R^2 score, as a function of the polynomial degrees used. This is shown in Fig. 6 (note that $1 - R^2$ is shown, not R^2 itself). As expected, both decrease as model complexity increases. We note that the MSE and the R^2 score are approximately proportional to $\propto 10^{-p/4}$ in the range plotted here.

Noisy Franke function

The next step in our validation scheme is to investigate how our OLS handles normally distributed noise added to the underlying functional values. Instead of $f_F(x,y)$ as we have fitted against previously, we now consider

$$\tilde{f}_F(x,y) = f_F(x,y) + \eta \mathcal{N}(0,1), \qquad (26)$$

where $\mathcal{N}(\mu, \sigma)$ is normally distributed stochastic noise with mean $\mu = 0$ and variance and standard deviation $\sigma = \sigma^2 = 1$. The parameter η determines the size of the noise, i.e. the signal to noise ratio (since the Franke function takes values on the order of $\propto 1$). As an illustrative example, we consider the case of $p = 2$, with only five β parameters and perform an OLS fit on the noisy $\tilde{f}_F(x,y)$ function. The resulting $\sigma^2(\beta_j)$ values are shown in Fig. 7, where we have used the Bootstrap scheme to extract the variances.

The variance increases as the noise overlying the Franke function increases. This is not unexpected, but it may be more interested to investigate how well the OLS model is able to fit the underlying function when the noise is included in the fitting. In order to visualize this, we plot R^2 and the mean squared error as functions of the noise scale factor η, *and* the corresponding R^2 and MSE values *relative to the plain Franke function with no noise*. For convenience we denote the R^2 score and the MSE values relative to the Franke function with no noise $R^2_{\eta=0}$ and $\text{MSE}_{\eta=0}$.

If the model is able to fit the underlying function despite the noise, then the values of $1 - R^2_{\eta=0}$ and $\text{MSE}_{\eta=0}$ will remain low even as $1 - R^2$ and the MSE relative to the noisy Franke function increases. Once $1 - R^2_{\eta=0}$ and $\text{MSE}_{\eta=0}$ start increasing, it means the noise is too much for the OLS scheme to handle and the fit is no longer representative of the underlying function. An example of this is shown in Fig. 8, where a $p = 10$ order polynomial was used to fit for different η values. We note that the dotted lines ($1 - R^2_{\eta=0}$ and $\text{MSE}_{\eta=0}$) remain essentially unchanged up to $\eta \approx 0.1$, meaning a signal to noise ratio of $\propto 10$ is easily handled by our model. As η increases further, the fit to the underlying Franke function becomes poorer, but surprisingly at $\eta = 1$—at which the size of the noise is on the order of the size of the function itself—the model still represents a good fit to $f_F(x,y)$ at $\text{MSE} \approx 10^{-3}$. Comparing to Fig. 6 we see that this is close to the MSE of $p = 6$ degree fit with $\eta = 0$.
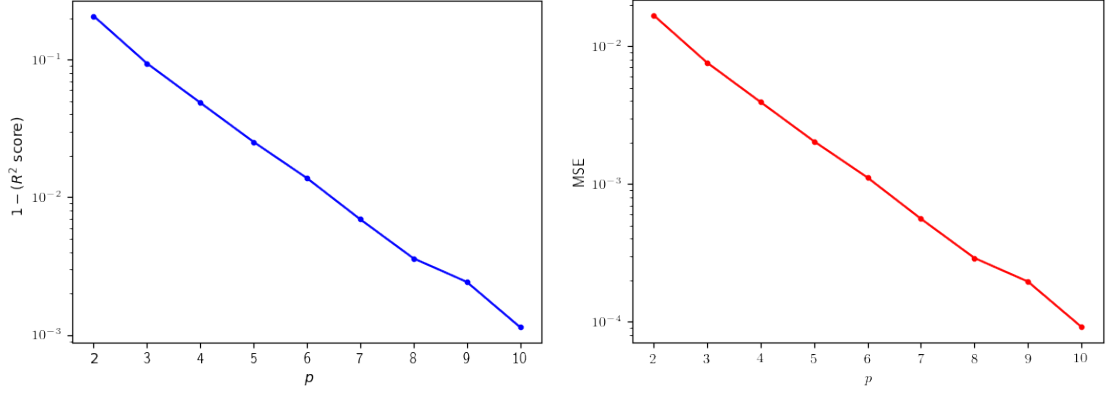
FIG. 6. Mean squared error and coefficient of determination, R^2, for the ordinary least squares fits on the Franke function. The parameter p indicates the order of polynomials used in the fitting.
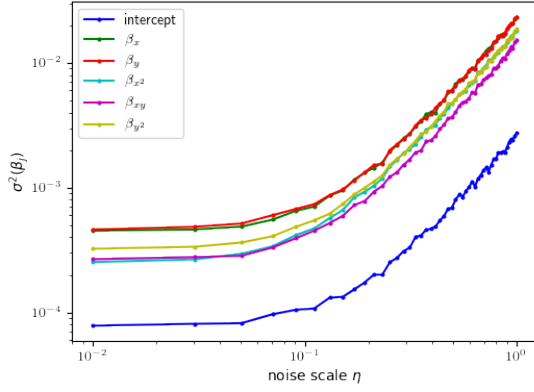

FIG. 7. The variance of the β_j coefficients of a $p = 2$ OLS fit on the noisy Franke function $\tilde{f}_{\mathrm{F}}(x, y)$ with signal to noise ratio η. A total of $k = 10\,000$ Bootstrap samples were used to calculate the variances for each value of η.
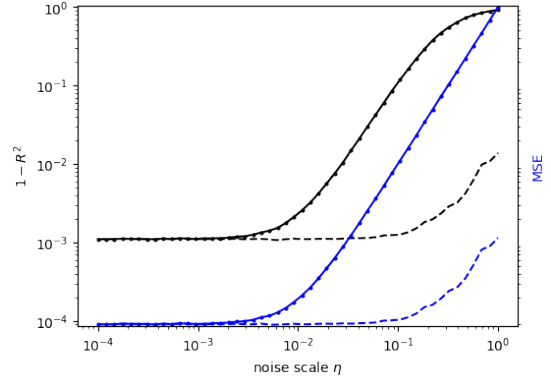

FIG. 8. The mean squared error and 1 minus the coefficient of determination R^2 as a function of the noise scale factor η. The dotted line represents the MSE and the R^2 score *relative to the underlying Franke function with no noise*. The full line with circles shows the corresponding values for the Franke function with the noise (the function the fit was performed on). A $p = 10$ order polynomial was used in this OLS fit.
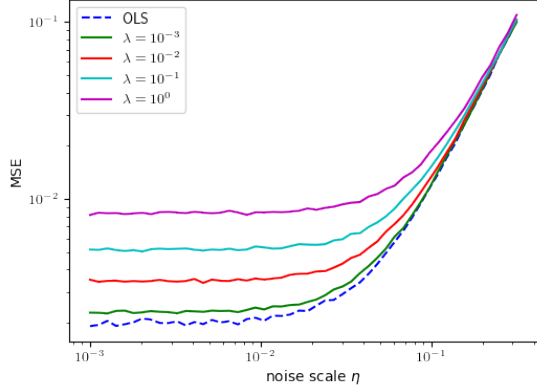
12

FIG. 9. The mean squared error as a function of the signal-to-noise ratio η for Ridge regression at various different shrinkage values λ. The OLS result (corresponding to $\lambda = 0$) is inset for comparison. A $p = 5$ order polynomial was used in these fittings.
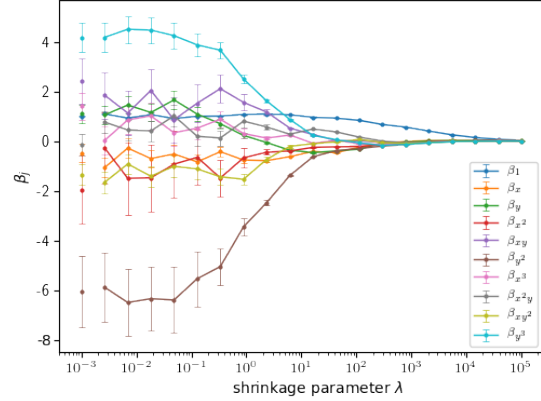

FIG. 10. The coefficients β_j for varying shrinkage parameter λ for Ridge regression fitting of the noisy Franke function with noise-to-signal ratio $\eta = 1.0$. The error bars represent the confidence intervals for each β_j, calculated as $\beta_j \pm 2\sigma^2(\beta_j)$, where $\sigma^2(\beta_j)$ is the Bootstrap computed variance for each β_j. Note carefully: The disconnected first point plotted at $\lambda = 10^{-3}$ represents the ordinary least squares β_js and their confidence intervals. The number of Bootstrap samples used to compute the variance in every case was $k = 10\,000$.

Ridge regression on the Franke function

Having investigated the behaviour of the OLS scheme in some detail, we now turn to the Ridge regression method. Since this introduces a shrinkage parameter λ it is natural to start with exploring how the MSE and the coefficient of determination depends on it. The MSE's dependence on the shrinkage parameter is shown in Fig. 9. We note—somewhat surprisingly—that for no value of λ is the computed MSE value lower than that of the ordinary least squares method ($\lambda = 0$). This is true over the entire spectrum of noise scales considered, $\eta = 10^{-3}$ to $\eta = 1$. We note that as the noise-to-signal ratio increases, the dependence on the shrinkage parameter λ lessens, and at $\eta = 1$ the model shows essentially no dependence on λ (i.e. the OLS and Ridge regressions for all λ values result in the same prediction).

The corresponding plot of the coefficient of determination, R^2, shows behaviour equal to that of Fig. 9 and is therefore omitted from the present work.

As discussed in section F, the Ridge regression method shrinks the β_j parameters corresponding to low-variance principal components of the OLS solutions. We can visualize the shrinkage done in the Ridge regression scheme by considering the β_j parameters as a function of λ for some fixed model. In order to keep the number of β_js relatively low, we consider in the follwing

a $p = 3$ polynomial model. We set the noise factor to $\eta = 1.0$ and consider the sizes of the parameters β_j for varying λ. This is shown in Fig. 10. Note carefully that the Ridge scheme is unable to set individual β_js to zero, but rather shrinks all β values. This is as expected, c.f. section E.

Lasso regression on the Franke function

Finally, we consider the Lasso regression scheme applied to the Franke function. As with the Ridge method, we first take a look at the MSE as a function of the shrinkage factor λ. This is shown in Fig. 11. As with the Ridge regression, we note that no value of λ gives better MSE results than OLS.

Whereas Ridge regression is unable to selectively put β_j parameters to zero, the Lasso regression can perform variable selection as well as regularization. Fig. 12 illustrates this. It is the corresponding Lasso regression version of Fig. 10, showing the evolution of β_j as λ increases.
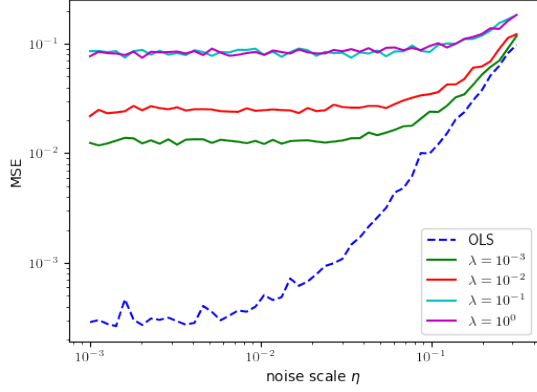
FIG. 11. The mean squared error as a function of the signal-to-noise ratio η for Lasso regression at various different shrinkage values λ. The OLS result (corresponding to $\lambda = 0$) is inset for comparison. A $p = 8$ order polynomial was used in these fittings.
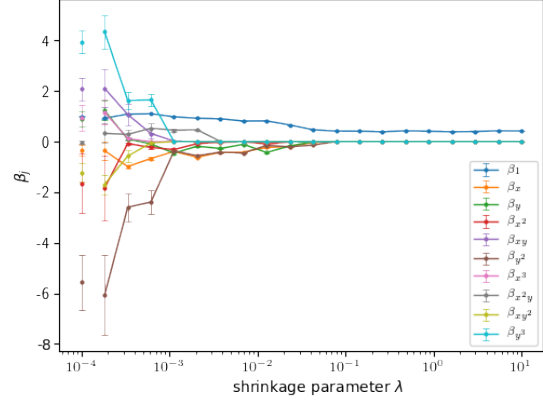
The same degree $p = 3$ basis as was used in the Ridge case is also used in Fig. 12.

Bias-variance tradeoff

As discussed in section G, increasing the flexibility of the model increases the variance but reduces the bias. Since the expected value of the error—the MSE—is the sum of the bias, the varaince, and the irreducible error, there must exist some sweet spot in which the total MSE is minimal. It is customary to plot the MSE, the bias, and the variance in a single plot and observe that as the bias initially decreases as complexity increases, so does the MSE. As complexity increases too much, and the variance begins to dominate the total MSE (overfitting), the error will start to increase. The bias-variance sweetspot is then found somewhere in the middle: The model complexity which minimizes the MSE. For the noisy Franke function we present such a plot in Fig. 13.

Note that, although the overall shape of the three lines are correct, the variance rises extremely slowly in comparison to the bias (note carefully the different vertical scales). Because of this, it appears that higher complexity is objectively better for the current problem example with the Franke function, up to and including polynomial degrees $p = 20$. Since we are computationally restricted to relatively low polyno-

FIG. 12. The coefficients β_j for varying shrinkage parameter λ for Lasso regression fitting of the noisy Franke function with noise-to-signal ratio $\eta = 1.0$. The error bars represent the confidence intervals for each β_j, calculated as $\beta_j \pm 2\sigma^2(\beta_j)$, where $\sigma^2(\beta_j)$ is the Bootstrap computed variance for each β_j. Note carefully: The disconnected first point plotted at $\lambda = 10^{-4}$ represents the ordinary least squares β_js and their confidence intervals. The number of Bootstrap samples used to compute the variance in every case was $k = 100$.
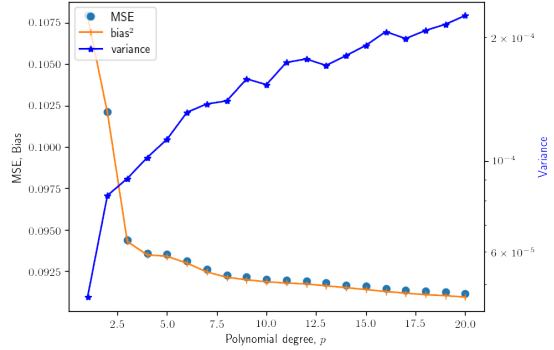

FIG. 13. The mean squared error, the bias, and the variance for Ridge regression with $\lambda = 1$ on the Franke function with noise scale $\eta = 0.3$. The number of Bootstrap samples used to calculate the variance is $k = 100$. Note carefully the different axes for the variance (right hand side of the figure).

14

mial degrees in the present work, we are not able to explore regions $p > 20$ for which the variance eventually begins to dominate the MSE expression.

This may be the reason why we find that the OLS scheme is strictly better (in terms of the MSE) than both the Ridge and Lasso methods in the present work: We are simply in a complexity region in which the variance is a non-issue for the MSE as a whole. Essentially, we are too far from overfitting our model, that L^1 and L^2 regularization (and even the variable selection of Lasso regression) does not benefit us much.

M. Terrain data parametrization

We turn now to terrain data parametrization, and use a subsection of the data shown in section K and Fig. 2. The raw data set is 3601×1801 points, of which we extract every 60 points in the first dimension and every 30 in the second. This results in a training set of size 60×60. For testing purposes, we extract a similar 60×60 testing data set by performing the same subsectioning only with indices shifted 30 points in each direction. This yields 3600 training points and 3600 testing points on which to evaluate our models.

In order to give our models the best possible chance of giving good fits, we prepare the data set by normalizing it and subtracting the minimum value, i.e. for every (x, y) point and height z we perform the following assignments:

$$z \leftarrow z - \min_{\text{all points}}(z) \qquad (27)$$

$$z \leftarrow \frac{z}{\max_{\text{all points}}(z)}. \qquad (28)$$

We consider the x and y coordinates to take values in $(0, 1]$ as for the Franke function of section J.

Having already developed and extensively tested our methods, all that remains is simply to apply OLS, Ridge, and Lasso regression to this data set. The resulting fits for $\lambda_{\text{R}} = 10^{-3}$ and $\lambda_{\text{L}} = 10^{-4}$ is shown in Fig. 15. These shrinkage factors are simply used as an example. A comparison of the MSE calculated on the same data set using different values of λ is shown in Fig. 14. We note that the OLS scheme is superior all around, with the Lasso method a distant second and the Ridge coming in third. The ordinary scheme produces the best MSE at $\text{MSE}_{\text{OLS}} \simeq 0.00928$.
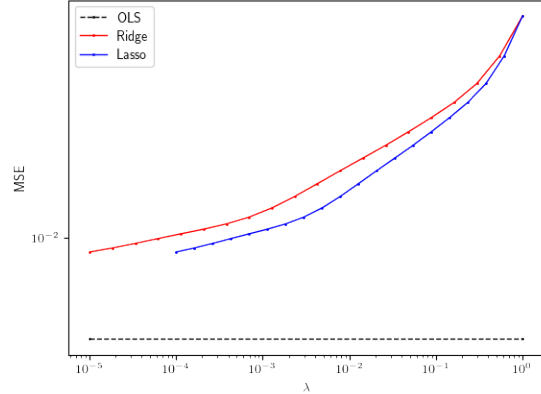

FIG. 14. The calculated mean squared error for the test data set described in section M for varying values of the shrinkage parameter λ. The OLS, the Ridge regression, and the Lasso scheme are all shown.

V. CONCLUSION

In the present work we have performed linear regression fits of real-valued functions of two variables. The Franke test function, as well as the terrain data considered in section M, both reveal that the ordinary least squares fits are superior to the regularized Ridge and Lasso fits for the present data sets. We propose that this is because we are in a bias-dominated region of the MSE, where the variance hardly matters at all for the overall error. This means that the variance-reducing regularization schemes do not benefit the MSE calculation much, giving OLS the edge.

An alternative explanation may be that we have some errors in our code (or more fundamentally, in our understanding of the subject) which skews our results in favour of the OLS. Being fresh in the field, our confidence is not exactly super high in this regard.
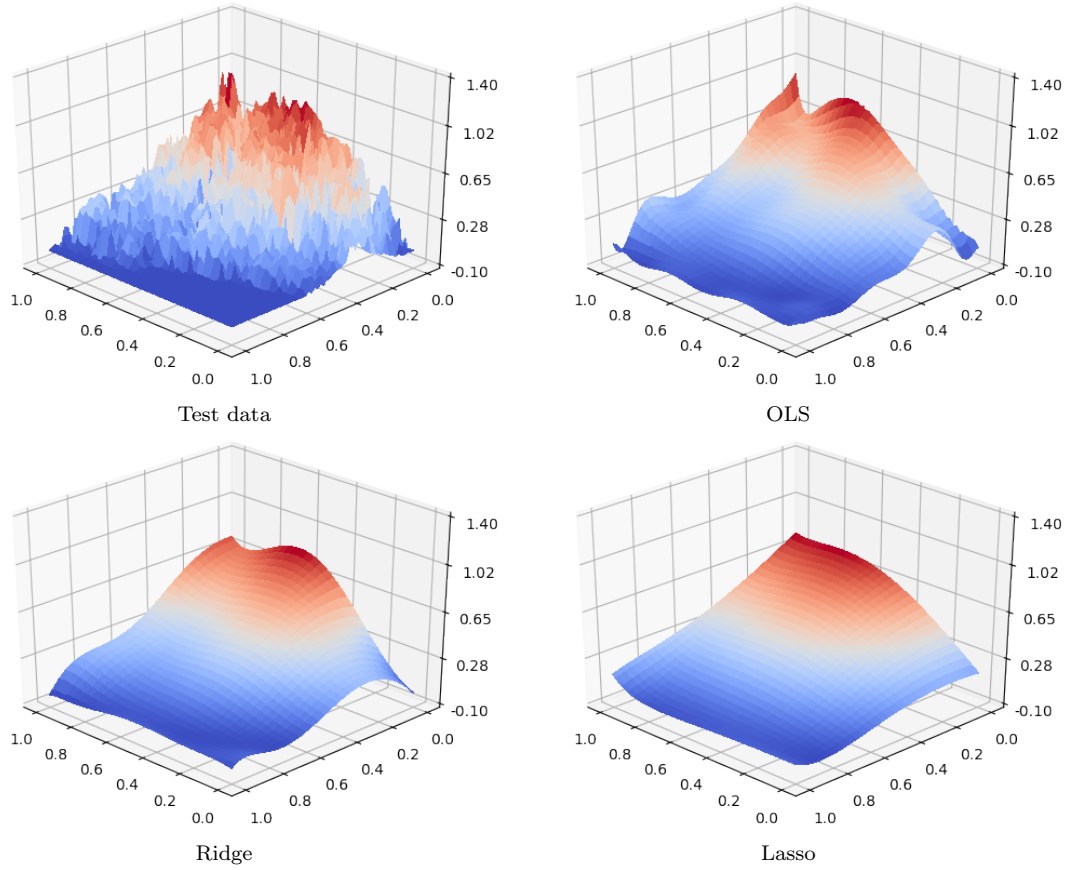
15

FIG. 15. Linear regression fits of a subset of the terrain data shown in Fig. 2. A 60×60 cutout of the original data is used as training data, while a separate equally large batch of points is used as test data for evaluating the fits. The test data itself is shown in the upper left hand corner. A shrinkage parameter $\lambda = 10^{-3}$ is used for the Ridge regression, while $\lambda = 10^{-4}$ is used for the Lasso scheme. We note that the OLS fit appears to best approximate the underlying data. This is also corroborated by the MSE calculations, which yields $\mathrm{MSE_{OLS}} < \mathrm{MSE_{Ridge}} < \mathrm{MSE_{Lasso}}$.

REFERENCES

[1] Richard Franke. *A critical comparison of some methods for interpolation of scattered data.* Tech. rep. Monterey, California: Naval Postgraduate School., 1979.

[2] Trevor Hastie, Robert Tibshirani, and JH Friedman. *The elements of statistical learning: data mining, inference, and prediction.* 2009.

[3] Adrien Marie Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes.* F. Didot, 1805.

[4] Arthur E Hoerl and Robert W Kennard. "Ridge regression: Biased estimation for nonorthogonal problems". In: *Technometrics* 12.1 (1970), pp. 55–67.

[5] Andrey Tikhonov. "On the stability of inverse problems". In: *Doklady Akademii Nauk SSSR* 39.5 (1943), pp. 195–198.

[6] David Lay. *Linear Algebra and Its Applications.* 4th ed. Pearson, 2012.

[7] Robert Tibshirani. "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288. ISSN: 00359246.

[8] P. Mehta et al. "A high-bias, low-variance introduction to Machine Learning for physicists". In: *ArXiv e-prints* (Mar. 2018). arXiv: 1803.08823 [physics.comp-ph].