Regression analysis and resampling methods

Johan Mylius Kroken[1,2], and Nanna Bryne[1,2]

[1] Institute of Theoretical Astrophysics, University of Oslo, Norway
[2] Center for Computing in Science Education (CCSE), University of Oslo, Norway

October 7, 2022

ABSTRACT

We perform linear regression on two different data sets in an attempt to fit a twodimensional polynomial onto them. The first data set is a 20×20 grid for which we compute values using the Franke function. We investigate the effect of adding normally distributed noise $\varepsilon \sim \mathcal{N}(0,1)$ to the function. The whole data set is z-score normalised. We perform OLS regression, Ridge regression and Lasso regression on the data, and we optimise and in turn assess the validity using (5-10)-fold cross-validation and bootstrap resampling techniques. For the OLS regression we find an optimal model with polynomial degree $d^{\text{OLS}} = 5$, and by using the same polynomial degree for the Ridge regression we also find an optimal penalty parameter $\lambda^{\text{Ridge}} = 7.58 \cdot 10^{-5}$. For the Lasso regression we allow the model to find a new optimal polynomial degree. It finds the end degree, for a $d \times \lambda$ grid so we cannot determine the validity of this. However, the MSE values found from Lasso and Ridge regression are not sufficiently smaller than that for OLS. The conclusion is thus that OLS with $d^{\text{OLS}} = 5$ yields the best model for the Franke function data.
We repeat the analysis for the terrain data, which is a part from the Grand Canyon in the United States, and find the Lasso regression to be very computationally expensive, especially for small λ, and it does not yield a significantly better result than OLS and Ridge. We therefore conclude for the terrain data that we have to viable models: The first found with OLS giving a model with $d^{\text{OLS}} = 6$ and the second with Ridge giving a model with $d^{\text{Ridge}} = 18$ and $\lambda^{\text{Ridge}} = 1.23 \cdot 10^{-4}$. The Ridge model is arguably favourable since it is able to reproduce the canyon's plateauing.

Contents

1. Introduction

Linear regression is the gateway to statistical analysis, and in this investigation we will perform three types of linear regression: ordinary least squares (OLS), Ridge, and Lasso regression, on two different data sets: The Franke function and terrain data of parts of the Grand Canyon in the United States. Common for all is, in addition to minimise error, that we will fit a two-dimensional polynomial onto the two data sets by finding an optimal set of parameters $\hat{\boldsymbol{\beta}}$, which in our case will be the coefficients of the different terms in the polynomial. How we determine these parameters depend on the regression method of choice. The way of measuring error (deviation from data), the cost function, varies between the different regression methods. Ridge and Lasso regression are so-called regularisation methods, allowing us to deal with more complex models, reducing the chance of over-fitting. Determining the best model is very dependant on the data set at hand. In section 2.1 we state the singular value decomposition from linear algebra. This comes in handy when explaining the regression models in section 2.2, especially the penalty term involved in the regularisation of the Ridge and Lasso schemes. We also explain resampling methods in section 2.3, which are useful when assessing the accuracy of our model. The main analysis is given in section 3 with an introduction to the data and noise of the Franke function in section 3.1, description of how and why we split the data in section 3.2, how we set up the model and design matrix in section 3.3, and how and why we scale the data in section 3.4. We then carry out the analysis of the Franke function in section 3.5, and of the terrain data in section 3.6. We draw conclusions in section 4. At last, we link to the code and list all figures, as well as catpions.

2. Theory

Throughout this project we concern ourselves with some observed values $\mathbf{y}$ for which we seek to obtain an approximation $\tilde{\mathbf{y}}$ which predicts the true value. Once we have created a model $\tilde{\mathbf{y}}$ we need to determine its accuracy somehow. There are numerous way of doing this, we will mostly use the mean squared error (MSE),

$$\mathrm{MSE}(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 \tag{1}$$

and the R2-score,

$$R^2(\mathbf{y}, \tilde{\mathbf{y}}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2} \tag{2}$$

where the mean of the observed values $\mathbf{y}$ is given by:

$$\bar{y} = \frac{1}{n}\sum_{i=0}^{n-1} y_i.$$

Before we delve into the various methods, let us have a look at some mathematical concepts that will be of great use in the further discussion.

2.1. Singular value decomposition

The singular value decomposition is a result from linear algebra that states that an arbitrary matrix A of rank r and size $n \times p$ can be decomposed into the following (David C. Lay 2016):

$$A = U\Sigma V^\intercal, \tag{3}$$

where Σ is a $n \times p$ diagonal matrix with the singular values of A as diagonal elements in descending order: $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \cdots \geq \sigma_r \geq 0$. That is:

$$\Sigma = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}, \qquad D = \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_r \end{bmatrix}$$

where D is a diagonal matrix of size $r \times r$ and $r \leq \min(n, p)$. Further is U a $n \times n$ matrix whose first r columns is an orthornormal basis of Col$A$. The remaining columns span Nul$A^\intercal$. Alltogether, U forms an orthonormal basis set spanning $\mathbb{R}^n$. Likwise, V is a $p \times p$ square matrix whose columns are an orthonormal basis spanning $\mathbb{R}^p$. The first r columns of V form an orthonormal basis of Row$A$. The ramaining columns span Nul$A$. As a result of the orthogonality of U and V we have that $U^\intercal U = VV^\intercal = \mathbb{1}$

2.2. Linear regression

We will attempt to create a model $\tilde{\mathbf{y}}$ by the means of linear regression. There are several possible estimation techniques when fitting a linear regression model. We will discuss three common approaches, one least squares estimation (section 2.2.2) and two forms of penalised estimation (section 2.2.3 and section 2.2.4).

We assume the vector $\mathbf{y} \in \mathbb{R}^n$ consisting of n observed values y_i to take the form:

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

where $f(\mathbf{x}) \in \mathbb{R}^n$ is a continous function and $\boldsymbol{\varepsilon} = \eta\mathcal{N}(\mu, \sigma) \in \mathbb{R}^n$ is a normally distributed noise of mean $\mu = 0$ and standard deviation σ and with an amplitude tuning parameter η.

We approximate f by $\tilde{\mathbf{y}} = X\boldsymbol{\beta}$, where $X \in \mathbb{R}^{n\times p}$ is a design matrix of n row vectors $\mathbf{x}_i \in \mathbb{R}^p$, and $\boldsymbol{\beta} \in \mathbb{R}^p$ are the unknown parameters to be determined. That is, we assume a *linear* relationship between X and $\mathbf{y}$. The integers n and p then represent the number of data points and features, respectively.

For an observed value y_i we have $y_i = \mathbf{x}_i^\intercal\boldsymbol{\beta} + \varepsilon_i = (X\boldsymbol{\beta})_i + \varepsilon_i$. The inner product $(X\boldsymbol{\beta})_i$ is non-stochastic, hence its expectation value is:

$$\mathbb{E}\big[(X\boldsymbol{\beta})_i\big] = (X\boldsymbol{\beta})_i$$

and since

$$\mathbb{E}\big[\varepsilon_i\big] \overset{\text{per def.}}{=} 0,$$

we have the expectation value of the response variable as:

$$\begin{aligned}\mathbb{E}\big[y_i\big] &= \mathbb{E}\big[(X\boldsymbol{\beta})_i + \varepsilon_i\big] \\ &= \mathbb{E}\big[(X\boldsymbol{\beta})_i\big] + \mathbb{E}\big[\varepsilon_i\big] \\ &= (X\boldsymbol{\beta})_i.\end{aligned}$$

To find the variance of this dependent variable, we need the expetation value of the outer product $\mathbf{y}\mathbf{y}^\intercal$,

$$\begin{aligned}\mathbb{E}\big[\mathbf{y}\mathbf{y}^\intercal\big] &= \mathbb{E}\big[(X\boldsymbol{\beta} + \boldsymbol{\varepsilon})(X\boldsymbol{\beta} + \boldsymbol{\varepsilon})^\intercal\big] \\ &= \mathbb{E}\big[X\boldsymbol{\beta}\boldsymbol{\beta}^\intercal X^\intercal + X\boldsymbol{\beta}\boldsymbol{\varepsilon}^\intercal + \boldsymbol{\varepsilon}\boldsymbol{\beta}^\intercal X^\intercal + \boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\intercal\big] \\ &= X\boldsymbol{\beta}\boldsymbol{\beta}^\intercal X^\intercal + \mathbb{1}\sigma^2.\end{aligned} \tag{4}$$

The variance now becomes

$$\begin{aligned}\mathrm{Var}\big[y_i\big] &= \mathbb{E}\big[(\mathbf{y}\mathbf{y}^\intercal)_{ii}\big] - \Big(\mathbb{E}\big[y_i\big]\Big)^2 \\ &= (X\boldsymbol{\beta})_i(X\boldsymbol{\beta})_i + \sigma^2 - (X\boldsymbol{\beta})_i(X\boldsymbol{\beta})_i \\ &= \sigma^2.\end{aligned}$$

The optimal estimator of the coefficients $\boldsymbol{\beta}_j$, call it $\hat{\boldsymbol{\beta}}$, is in principle obtained by minimising the cost function $C(\boldsymbol{\beta})$. The cost function is a measure of how badly our model deviates from the observed values, and the method we choose is defined from its cost function. By minimising it we obtain $\hat{\boldsymbol{\beta}}$, that is:

$$\left.\frac{\partial C(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}\right|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}} = 0. \tag{5}$$

2.2.1. Bias-variance tradeoff

In order to understand the error of our function, we often divide the error into different classifications. This is commonly referred to as bias and variance. When deriving it, we consider When considering the bias-variance trad off we take into account the mean squared error of the cost function: .

$$C(X,\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = \mathbb{E}\big[(\mathbf{y}-\tilde{\mathbf{y}})^2\big],$$

which is a measure of the expected error in our model. Having a model like $\mathbf{y} = \boldsymbol{f} + \boldsymbol{\varepsilon}$ where $\boldsymbol{f} = f(\mathbf{x})$ yields:

$$\begin{aligned} C(X,\boldsymbol{\beta}) &= \mathbb{E}\big[(\boldsymbol{f}+\boldsymbol{\varepsilon}-\tilde{\mathbf{y}})^2\big] = \mathbb{E}\big[\varepsilon^2 + 2\varepsilon(\boldsymbol{f}-\tilde{\mathbf{y}}) + (\boldsymbol{f}-\tilde{\mathbf{y}})^2\big] \\ &= \boldsymbol{\sigma}^2 + \mathbb{E}\big[(\boldsymbol{f}-\tilde{\mathbf{y}})^2\big] \end{aligned}$$

where we have used that $\mathbb{E}\big[\varepsilon^2\big] = \boldsymbol{\sigma}^2$ and $\mathbb{E}\big[\boldsymbol{\varepsilon}\big] = 0$ We then add and subtract $\mathbb{E}\big[\tilde{\mathbf{y}}\big]$ to the last term and obtain:

$$\begin{aligned} \mathbb{E}\big[(\boldsymbol{f}-\tilde{\mathbf{y}})^2\big] &= \mathbb{E}\big[\,\big(\boldsymbol{f}-\tilde{\mathbf{y}}+\mathbb{E}\big[\tilde{\mathbf{y}}\big]-\mathbb{E}\big[\tilde{\mathbf{y}}\big]\big)^2\,\big] \\ &= \mathbb{E}\big[\,\big(\boldsymbol{f}-\mathbb{E}\big[\tilde{\mathbf{y}}\big]\big)^2 + \big(\mathbb{E}\big[\tilde{\mathbf{y}}\big]-\tilde{\mathbf{y}}\big)^2\,\big] \\ &+ \mathbb{E}\big[2\,\big(\boldsymbol{f}-\mathbb{E}\big[\tilde{\mathbf{y}}\big]\big)\,\big(\mathbb{E}\big[\tilde{\mathbf{y}}\big]-\tilde{\mathbf{y}}\big)\,\big] \\ &= \big(\boldsymbol{f}-\mathbb{E}\big[\tilde{\mathbf{y}}\big]\big)^2 + \mathbb{E}\big[\,\big(\tilde{\mathbf{y}}-\mathbb{E}\big[\tilde{\mathbf{y}}\big]\big)^2\,\big] \end{aligned}$$

where $\mathbb{E}\big[\mathbb{E}\big[\tilde{\mathbf{y}}\big]-\tilde{\mathbf{y}}\big] = 0$ and $\mathbb{E}\big[\,\big(\boldsymbol{f}-\mathbb{E}\big[\tilde{\mathbf{y}}\big]\big)^2\,\big] = \big(\boldsymbol{f}-\mathbb{E}\big[\tilde{\mathbf{y}}\big]\big)^2$

We are then left with the following expression:

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

where the bias of our model $\tilde{\mathbf{y}}$ compared to the continuous function we try to replicate $\boldsymbol{f}$ is given by:

$$(\mathrm{Bias}[\tilde{\mathbf{y}}])^2 = \big(\boldsymbol{f}-\mathbb{E}\big[\tilde{\mathbf{y}}\big]\big)^2$$

and the variance of $\tilde{\mathbf{y}}$ is:

$$\mathrm{Var}[\tilde{\mathbf{y}}] = \mathbb{E}\big[\,\big(\tilde{\mathbf{y}}-\mathbb{E}\big[\tilde{\mathbf{y}}\big]\big)^2\,\big]$$

and the irreductible error that arises as a result of stochastic noise is $\boldsymbol{\sigma}^2$ Having a high bias means that our model predicts the wrong results, even if there is a small spread (small variance) in the predictions. Having a low bias, but a high variance means that the predictions made by our model vary a lot, but are centred around the true value. An exercise when creating a viable model is to perform a bias-variance trade off, as too much of either will make the predictions less trustworthy.

2.2.2. ordinary least squares (OLS)

The ordinary least squares (OLS) method assumes the cost function

$$C^{\mathrm{OLS}}(\boldsymbol{\beta}) = \sum_{i=0}^{n-1}(y_i-\tilde{y}_i)^2 = \|\mathbf{y}-\tilde{\mathbf{y}}\|_2^2 = \|\mathbf{y}-X\boldsymbol{\beta}\|_2^2,$$

where the subscript "2" implies the ℓ^2-norm[1]. Solving eq. (5) for $C = C^{\mathrm{OLS}}$ yields the OLS expression for the optimal parameter.

$$\hat{\boldsymbol{\beta}}^{\mathrm{OLS}} = (X^\intercal X)^{-1}X^\intercal\mathbf{y} = H^{-1}X^\intercal\mathbf{y}, \tag{6}$$

where $H = X^\intercal X$ is the Hessian matrix. Letting $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}^{\mathrm{OLS}}$ we get the expected value

$$\begin{aligned} \mathbb{E}\big[\hat{\boldsymbol{\beta}}\big] &= \mathbb{E}\big[(X^\intercal X)^{-1}X^\intercal\mathbf{y}\big] \\ &= (X^\intercal X)^{-1}X^\intercal\mathbb{E}\big[\mathbf{y}\big] \\ &= (X^\intercal X)^{-1}X^\intercal X\boldsymbol{\beta} \\ &= \boldsymbol{\beta}. \end{aligned}$$

The variance is then

$$\begin{aligned} \mathrm{Var}\big[\hat{\boldsymbol{\beta}}\big] &= \mathbb{E}\big[\hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^\intercal\big] - \mathbb{E}\big[\hat{\boldsymbol{\beta}}\big]\mathbb{E}\big[\hat{\boldsymbol{\beta}}^\intercal\big] \\ &= \mathbb{E}\big[(X^\intercal X)^{-1}X^\intercal\mathbf{y}\mathbf{y}^\intercal X((X^\intercal X)^{-1})^\intercal\big] - \boldsymbol{\beta}\boldsymbol{\beta}^\intercal \\ &= (X^\intercal X)^{-1}X^\intercal\mathbb{E}\big[\mathbf{y}\mathbf{y}^\intercal\big]X(X^\intercal X)^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^\intercal \\ &\overset{(4)}{=} (X^\intercal X)^{-1}X^\intercal(X\boldsymbol{\beta}\boldsymbol{\beta}^\intercal X^\intercal + \mathbb{1}\sigma^2)X(X^\intercal X)^{-1} \\ &= \boldsymbol{\beta}\boldsymbol{\beta}^\intercal + (X^\intercal X)^{-1}X^\intercal\sigma^2X(X^\intercal X)^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^\intercal \\ &= \sigma^2(X^\intercal X)^{-1}. \end{aligned} \tag{7}$$

If we perform the singular value decomposition from section 2.1, and rewrite X using eq. (3) we obtain the following:

$$\begin{aligned} \tilde{\mathbf{y}} &= X\hat{\boldsymbol{\beta}} = X(X^\intercal X)^{-1}X^\intercal\mathbf{y} \\ &= U\Sigma V^\intercal(V\Sigma^\intercal U^\intercal U\Sigma V^\intercal)^{-1}V\Sigma^\intercal U^\intercal\mathbf{y} \\ &= U\Sigma V^\intercal(V\Sigma^2V^\intercal)^{-1}V\Sigma^\intercal U^\intercal\mathbf{y} \\ &= U\Sigma V^\intercal V(\Sigma^2)^{-1}V^\intercal V\Sigma^\intercal U^\intercal\mathbf{y} \\ &= U\Sigma^2(\Sigma^2)^{-1}V^\intercal VV^\intercal VU^\intercal\mathbf{y} \\ &= U\,\mathrm{diag}\left(\frac{\sigma_i^2}{\sigma_i^2}\right)U^\intercal\mathbf{y} = UU^\intercal\mathbf{y} = \sum_{i=1}^{p}\mathbf{u}_i\mathbf{u}_i^\intercal\mathbf{y}, \end{aligned} \tag{8}$$

where we have used that $U^\intercal U = VV^\intercal = \mathbb{1}$ and $(V^\intercal V)^{-1} = V^\intercal V \implies V^\intercal VV^\intercal V = \mathbb{1}$.

2.2.3. Ridge regression

Let $\lambda \in \mathbb{R}$ be some small number such that $\lambda > 0$. If we add a penalty term $\lambda\|\boldsymbol{\beta}\|_2^2$ to the OLS cost function, we get the cost function of Ridge regression,

$$\begin{aligned} C^{\mathrm{Ridge}}(\boldsymbol{\beta}) &= C^{\mathrm{OLS}}(\boldsymbol{\beta}) + \lambda\|\boldsymbol{\beta}\|_2^2 \\ &= \|\mathbf{y}-\tilde{\mathbf{y}}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 \\ &= \|\mathbf{y}-X\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 \end{aligned}$$

ADD SOME FILLER TEXT HERE

$$\hat{\boldsymbol{\beta}}^{\mathrm{Ridge}} = \big(X^\intercal X + \lambda\mathbb{1}\big)^{-1}X^\intercal\mathbf{y}$$

[1] Euclidian norm (ℓ^2-norm) is defined as $\|\mathbf{a}\|_2 = \sqrt{\sum_i a_i^2}$

We use the singular value decomposition to obtain a similar result as for OLS. We notice the only difference is the following term:

$$\begin{aligned}\left(X^\intercal X+\lambda\mathbb{1}\right)^{-1} &= \left(V\Sigma^\intercal U^\intercal U\Sigma V^\intercal+\lambda\mathbb{1}\right)^{-1}\\ &= \left(V\Sigma^\intercal\Sigma V^\intercal+\lambda\mathbb{1}\right)^{-1}\\ &= \left(V\Sigma^2V^\intercal+\lambda\mathbb{1}\right)^{-1}\\ &= \left(V\left[\Sigma^2+\lambda\mathbb{1}\right]V^\intercal\right)^{-1}\end{aligned}$$

We follow the same argument as with eq. (8) and obtain:

$$\begin{aligned}\tilde{\mathbf{y}}^{\text{Ridge}} &= X\hat{\boldsymbol{\beta}}^{\text{Ridge}}\\ &= U\Sigma V^\intercal\left(V\left[\Sigma^2+\lambda\mathbb{1}\right]V^\intercal\right)^{-1}V\Sigma^\intercal U^\intercal\mathbf{y}\\ &= U\Sigma^2\left[\Sigma^2+\lambda\mathbb{1}\right]^{-1}V^\intercal VV^\intercal VU^\intercal\mathbf{y}\\ &= U\,\text{diag}\left(\frac{\sigma_i^2}{\sigma_i^2+\lambda}\right)U^\intercal\mathbf{y}\\ &= \sum_{i=1}^{p}\mathbf{u}_i\frac{\sigma_i^2}{\sigma_i^2+\lambda}\mathbf{u}_i^\intercal\mathbf{y}.\end{aligned}\tag{9}$$

If we now compare eq. (8) to eq. (9) we see that they are fairly similar but the prediction $\tilde{\mathbf{y}}^{\text{Ridge}}$ contains a factor $\sigma_i^2/(\sigma_i^2+\lambda)$ where σ_i are the singular values of the design matrix X (follows from the singular value decomposition, section section 2.1), and λ is the penalty term which effectively shrinks the predicted values. The shrinkage is large when σ_i^2 is small, and thus adding this penalty term means that we are emphasizing the parts of X (and thereby the prediction), whose corresponding singular values are the largest. This is called *principal component analysis*.

We state here, without derivation, that the variance of $\boldsymbol{\beta}^{\text{Ridge}}$ is

$$\text{Var}\left[\boldsymbol{\beta}^{\text{Ridge}}\right]=\sigma^2(X^\intercal X+\lambda\mathbb{1})^{-1}X^\intercal X\left(X^\intercal X+\lambda\mathbb{1}\right)^\intercal,$$

which is generally smaller than for OLS, $\text{Var}\left[\boldsymbol{\beta}^{\text{Ridge}}\right]<\text{Var}\left[\boldsymbol{\beta}^{\text{OLS}}\right]$ (Hjorth-Jensen 2021).

2.2.4. Lasso regression

If we add the penalty term $\lambda\|\boldsymbol{\beta}\|_1$, now using the ℓ^1-norm[2], to the OLS cost function, we are left with the Lasso regression's cost function,

$$\begin{aligned}C^{\text{Lasso}}(\boldsymbol{\beta}) &= C^{\text{OLS}}(\boldsymbol{\beta})+\lambda\|\boldsymbol{\beta}\|_1\\ &= \|\mathbf{y}-\tilde{\mathbf{y}}\|_2^2+\lambda\|\boldsymbol{\beta}\|_1\\ &= \|\mathbf{y}-X\boldsymbol{\beta}\|_2^2+\lambda\|\boldsymbol{\beta}\|_1\end{aligned}$$

The analytical expression for $\hat{\boldsymbol{\beta}}^{\text{Lasso}}$ is not trivial to derive, and when performing the analysis we will use the `scikit-learn` package in python. Hence, we do not derive this analytical expression.

[2] Manhatten norm (ℓ^1-norm) is defined as $\|\mathbf{a}\|_1=\sum_i|a_i|$

2.3. Resampling

Having obtained some optimal parameters $\hat{\boldsymbol{\beta}}$ from either OLS, Ridge regression or Lasso regression it is of interest to determine how good of a prediction $\hat{\boldsymbol{\beta}}$ yields. Data is often limited and thus we resample the data in clever ways in order to test it for larger samples. We will consider two ways of resampling data, the Bootstrap and Cross Validation.

2.3.1. Bootstrap method

Suppose we have some set of data $\mathbf{y}$ from which we have estimated $\hat{\boldsymbol{\beta}}$. We think of $\boldsymbol{\beta}$ as a random variable (since $\boldsymbol{\beta}=\boldsymbol{\beta}(X)$) with an unknown probability distribution $p(\boldsymbol{\beta})$, that we want to estimate. We then have that $\hat{\boldsymbol{\beta}}$ is the $\boldsymbol{\beta}$ that has the highest probability. We do the following:

1. From the data $\mathbf{y}$ we draw with replacement as many numbers as there are in $\mathbf{y}$ and create a new dataset $\mathbf{y}^*$.
2. We then estimate $\boldsymbol{\beta}^*$ by using the data in $\mathbf{y}^*$.
3. Repeat this k times and we are left with a set of vectors $B=(\boldsymbol{\beta}_1^*,\boldsymbol{\beta}_2^*,\ldots,\boldsymbol{\beta}_k^*)$ The relative frequency of vectors $\boldsymbol{\beta}^*$ in B is our approximation of $p(\boldsymbol{\beta})$.

We now have a collection of k $\boldsymbol{\beta}$ parameters. If we assume y to be independent and identically distributed variables, the central limit theorem tells us that the distribution of $\boldsymbol{\beta}$ parameters should approach a normal distribution when k is sufficiently large. Thus, $\hat{\boldsymbol{\beta}}$, which is the beta with the highest probability should approach the expectation value of the above distribution, which for a normal distribution is just the mean values. We therefore write:

$$\hat{\boldsymbol{\beta}}^*=\mathbb{E}\left[\boldsymbol{\beta}^*\right]=\bar{B}$$

which is our estimate of the optimal parameter $\hat{\boldsymbol{\beta}}^*$ after bootstrapping. From the set of vectors B we can estimate the variance and standard error of $\boldsymbol{\beta}$, both of which will be vector quantities, with entries that corresponds to each feature in our model.

2.3.2. Cross-validation

Another resampling technique is the cross-validation. Suppose we have the data set $\mathbf{y}$ which we split into k smaller datasets equal in size. Then:

1. Decide on one (or more) of the sets to be the testing test. The remaining sets will be considered the training set.
2. Fit some model to the training set. Evaluate this model by finding the desired test scores. This could be the MSE and/or R^2 scores. Save these values on discard the model.
3. Repeat k times, or until all the data have been used as test data.

We use the retained scores for all the testing sets in our assessment of the model.

3. Analysis

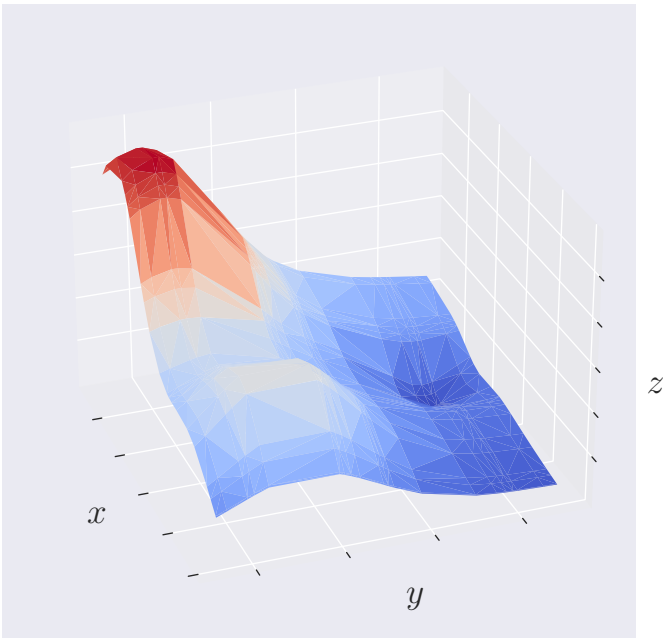

Fig. 1. The Franke function plotted on a grid where $N = 20$ and $\eta = 0$. Since we scale the data, the z becomes arbitrary and we choose to leave it out. The important information is the shape of the graph which is smooth when there is no noise.

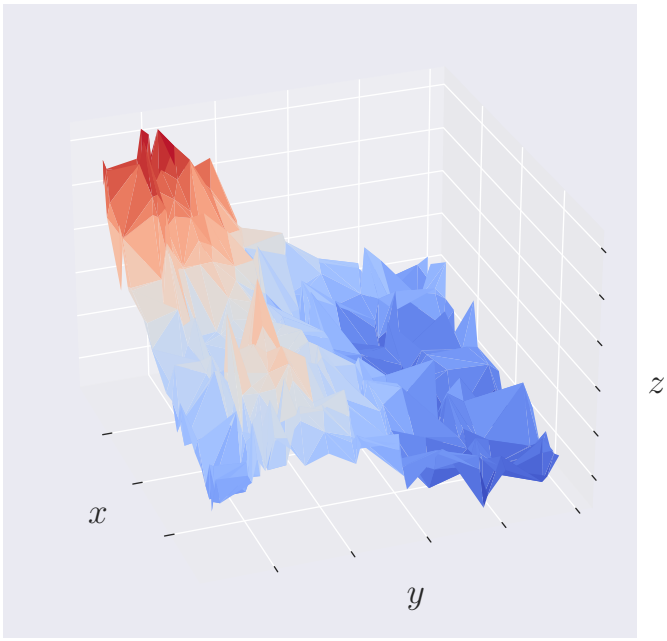

Fig. 2. The Franke function with added noise plotted on a grid where $N = 20$ and $\eta = 0.1$. Since we scale the data, the z becomes arbitrary and we choose to leave it out. The important information is the shape of the graph which becomes quite ragged even for this small level of noise

3.1. Data and noise

The function, onto which we are trying to fit a model, is the Franke Function(cite this). It is defined as follows:

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

In order to generate a dataset we will use N uniformly distributed values of $x, y \in [0,1]$. We will also add some normally distributed noise $\varepsilon = \eta\mathcal{N}(\mu,\sigma^2) = \eta\mathcal{N}(0,1)$ to $f(x,y)$, where η is a strength parameter controlling the amplitude of the added noise. The full description of our data then become:

$$\begin{aligned} \mathbf{y} &= f(x,y) + \eta\mathcal{N}(0,1) \\ &= f(\mathbf{x}) + \varepsilon \end{aligned} \tag{11}$$

where $\mathbf{x} = (x,y)$. From section section 2.2 we have a model for this data: $\tilde{\mathbf{y}} = X\hat{\boldsymbol{\beta}}$ where X is the design matrix and $\hat{\boldsymbol{\beta}}$ are the optimal parameters which we are trying to determine.

We visualise the data both with and without noise. Figure Figure 1 shows the Franke function without any noise ($\eta = 0$), plotted for uniformly distributed x and y, where $N = 20$. We could use more data points, but for the sake computational efficiency we use a 20×20 grid throughout this analysis. Figure Figure 2 show the same function, for the same points but now with an added noise of $\eta = 0.1$.

3.2. Data splitting

In order to test our estimate of $\hat{\boldsymbol{\beta}}$ we reserve some of the data for testing. We thus divide our data set into a part for testing a part for training. We select 80 % of the data for training and the remaining 20 % for testing the data. The data is split at random.

3.3. Model and design matrix

The design matrix X has dimensionality $(n \times p)$ where n represents the data points and p the features of the model. We have already split the data set into training and testing, and must therefore create two design matrices: X^{train} and X^{test}.
We want our model to be a two-dimensional polynomial of order d:

$$P_d(x,y) = \beta_0 + \sum_{l=1}^{d}\sum_{k=0}^{l}\beta_j x^{l-k}y^k$$

where $j \in [1,p]$ and $p = (d+1)\cdot[(d+2)/2]$ is the number of features, or number of terms in a two dimensional polynomial. β_0 is our intercept (constant term). The design matrix is set up without an intercept (constant term).

3.4. Scaling

We want to scale both our data and the design matrices because the data obtained from either the Franke function, or from some other source often vary a great deal in magnitude. Since the design matrices are set up in order to fit data to a polynomial of degree d in two dimensions, we want to be sure that no term is given more emphasis than the others. In addition, when working with multi-dimensional data

and design matrices we want to standardise the features as much as possible to ensure equal (relative) emphasise and treatment of the dimensions. We use the scaling technique *standardization* or *z-score normalization* which makes the mean of each feature equal to zero, and the their variances to be of unit length. For our observed data $\mathbf{y}$ this mean:

$$\mathbf{y}' = \frac{\mathbf{y} - \mu_{\mathbf{y}}}{\sigma_{\mathbf{y}}}$$

where $\mathbf{y}'$ is now our scaled data. For the design matrices, this must be done for each feature, i.e. for each column. Mathematically, if X_j is column j of the original design matrix X, $j \in [1, p]$:

$$X'_j = \frac{X_j - \mu_{X_j}}{\sigma_{X_j}}.$$

We do this for all the columns and end up with the scaled design matrix X'. Since $\hat{\boldsymbol{\beta}}$ is a function of the design matrix of the training data: X^{train} and the data $\mathbf{y}$, we need to scale both the training and test data with respect to the mean and standard deviation of each column of the train data:

$$X'^{\,\text{train}}_j = \frac{X^{\text{train}}_j - \mu_{X^{\text{train}}_j}}{\sigma_{X^{\text{train}}_j}}$$

$$X'^{\,\text{test}}_j = \frac{X^{\text{test}}_j - \mu_{X^{\text{train}}_j}}{\sigma_{X^{\text{train}}_j}}$$

This scaling will ensure that we treat the data in the same way, no matter the actual numerical value of the data.

3.5. Regression analysis for Franke function

3.5.1. OLS

We start by performing an Ordinary Least Square regression analysis, as outlined in section section 2.2.2, where we find $\hat{\boldsymbol{\beta}}^{\text{OLS}}$ from eq. (6) and $\text{Var}[\hat{\boldsymbol{\beta}}^{\text{OLS}}]$ from eq. (7). The error can be estimated with the standard deviation $\sigma_\beta = \text{Var}[\hat{\boldsymbol{\beta}}^{\text{OLS}}]^{1/2}$.

We have scaled the data and removed the intercept, thus we do not concern ourselves with β_0, so $\boldsymbol{\beta}^{\text{OLS}} = (\beta_1, \beta_2, \beta_3, \ldots, \beta_{p-1})$. We train models, i.e. we find $\hat{\boldsymbol{\beta}}^{\text{OLS}}$ for polynomials up to order 5. The values of these optimal parameters are plotted in Figure 3, with error bars of one standard deviation. We notice that the variation of the feature parameters increase as the polynomial degree of the model increase. This is expected because when the complexity of the model increase, it want to traverse more points exactly. We further see that the coefficients of the same features more or less have the same sign for the different polynomial degrees. This is a sign of a stable model, as we do not want these coefficient to change much when we use different polynomial degrees for our model. WHY??

Having found the optimal parameter $\hat{\boldsymbol{\beta}}^{\text{OLS}}$ we can make predictions by computing $\tilde{\mathbf{y}} = X\hat{\boldsymbol{\beta}}^{\text{OLS}}$ on both the training data and the test data. In order to say something about the quality of these predictions we compute the mean square error and the R^2 score from equations eq. (1) and eq. (2) respectively. The result of this is shown in Figure 4. We see that the MSE decreases and the R^2 score increases (towards 1) as the polynomial degree increase. This signifies that the data we are trying to model (the Franke function) shows a complex behaviour, and is not very well modelled by a low-degree polynomial. From this plot a lone it is fair to deduce that the higher the polynomial degree the better the model. However, we will see that this is not necessarily the case.

Fig. 3. Numerical value of the feature parameters β, with 1σ error bars, for polynomials up to order $d = 5$ for the OLS analysis of the Franke function ($N = 20$, $\eta = 0.1$). We note that the parameters for similar features tend to have the same sign across models, however there is an increase in parameter magnitude. This is clearly visible for $d = 5$.

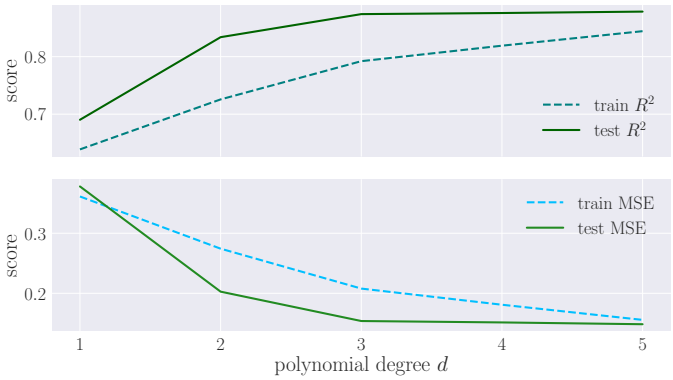

Fig. 4. MSE and R^2 scores for train and test data for polynomials up to order $d = 5$ for the OLS analysis of the Franke function ($N = 20$, $\eta = 0.1$). We see that for the polynomial degrees present, the MSE decrease, while the R^2 increase towards 1 for increasing polynomial degree.

We want to investigate the effect of the noise parameter η on the MSE for our model. In Figure 5 we have plotted this for $\eta = 10^\gamma, \gamma \in [-4, -3, -2, -1, 0]$. We observe that for large noise ($\eta = 1$) the MSE is high, which is expected. For lower noise the MSE is lower (pay attention to the logarithmic scale for the MSE). The MSE also decreases with polynomial degree for low noise, emphasizing what we found in Figure 4. In order to represent something slightly more physical than a smooth curve (e.g. terrain data), we will use $\eta = 0.1$ for our further analysis.

From Figure 4 we got the impression the higher the polynomial degree, the better the model. We want to investigate this further and look at how the (MEAN) MSE behaves for polynomials of order up to 20. The model is trained on the training data, but the MSE is evaluated for both the training and test data. The result is shown in Figure 6. The MSE of the training data decrease as the polynomial degree in-

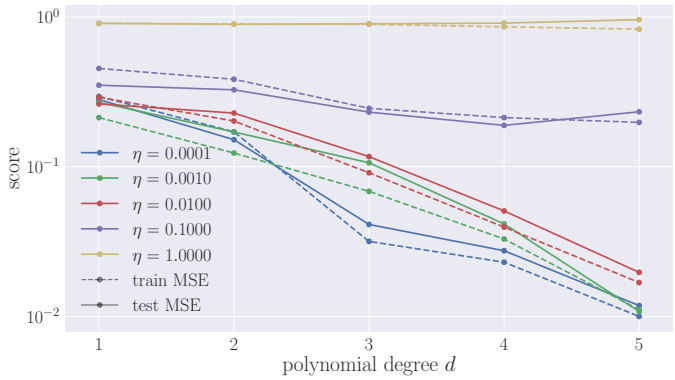

Fig. 5. MSE for train and test data for polynomials up to order $d = 5$ for the OLS analysis of the Franke function ($N = 20$, $\eta = 0.1$), for different noise parameters $\eta = 10^{\gamma}, \gamma \in [-4, -3, -2, -1, 0]$. The MSE is large for noisy function (large η), but decreasing with polynomial degree.

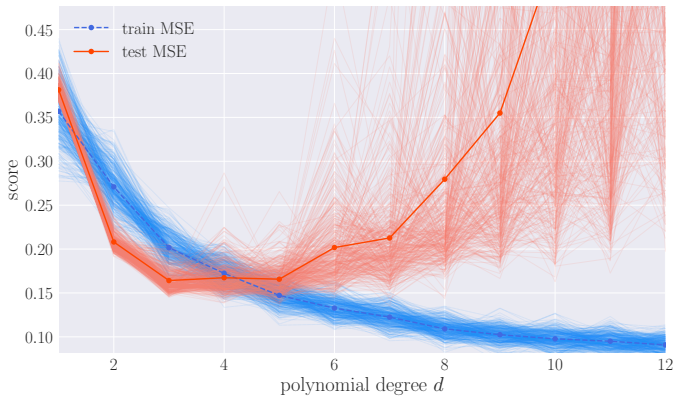

Fig. 6. MSE for train and test data for polynomials up to order $d = 12$ for the OLS analysis of the Franke function ($N = 20$, $\eta = 0.1$) generated with a $k = 400$ bootstrap. When we increase the model complexity up to order $d = 12$ we easily spot that the MSE for train data starts to diverge, which could be a sign of over-fitting.

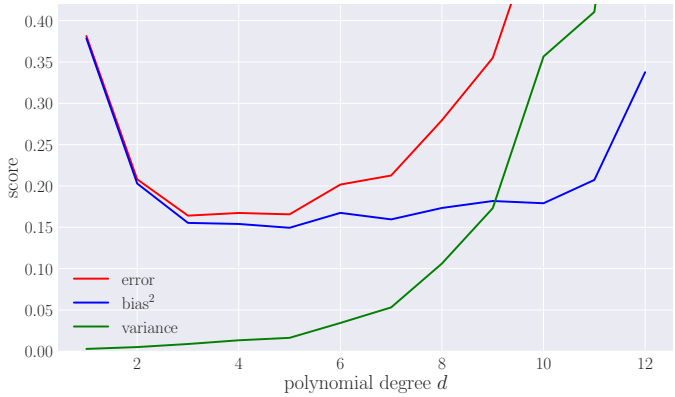

Fig. 7. Bias, variance and residual error for polynomials up to order $d = 12$ for the OLS analysis of the Franke function ($N = 20$, $\eta = 0.1$), generated with a $k = 400$ bootstrap. We see that for low polynomial degrees there is a high bias and low variance, the opposite for larger degrees, and we find a trade off at around $d \approx 5$.

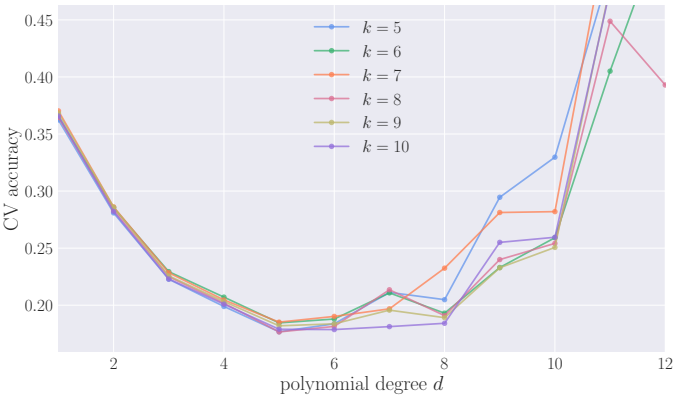

Fig. 8. MSE for test data for polynomials up to order $d = 12$ for the OLS analysis of the Franke function ($N = 20$, $\eta = 0.1$) generated with an $k \in [5, 10]$-fold cross-validation. We observe a similar behaviour with Figure 6 where the MSE for the test data reaches a minimum at $d = 5$ and then seem to diverge.

crease. This is not a surprise, since the model will traverse more points of the training data exactly as the complexity (d) increase. The MSE of the testing data however, seem to decrease at first, but then rapidly increase as the polynomial degree increase. This is similarly explained with the model being too tailored to the training data. When applied to the test data (which consist of completely different pieces of data), the model fails to make accurate predictions. This is an indicator of a phenomenon called *over-fitting*: when the model is so tailored to the training data it is unable to make accurate predictions on other, similar data. For a more reliable result, the data in figure Figure 6 is generated by bootstrapping the training data with $k = 400$ as explained in section 2.3.1. We consider 400 a suitable number of bootstraps in order to gain an accurate result without requiring too much computational time. Without showing it here, a larger number of bootstraps yield more or less the same results. NANNA CONFIRM THIS.

We take the analysis further and consider the bias-variance trade off, as explained in section 2.2.1. The result is shown in figure Figure 7 where we clearly see that for low polynomial degrees the error occurs mostly from bias, and for high degrees from variance. There is a trade off, where the total error switches main dependence from bias to variance, ultimately minimising it. In Figure 7 this trade off can be found around polynomial degree $d \approx 5$.

A similar reasoning can be made by applying cross-validation as a resampling technique, as explained in section 2.3.2. The result of this is shown in Figure 8 where we have used an 8-fold cross-validation of the training data and found the corresponding MSE for the training and test data. From the graph we see that MSE for the training data decrease with polynomial degree, which is expected w.r.t. to previous discussion. Similarly the MSE for the test data reaches a minimum and seem to diverge. The minimum point is found at $d \approx 5$.

It has become apparent that the polynomial degree that seem to fit our data the best is $d = 5$, and so we want to investigate this model further. Considering the bootstrapping resampling technique we make a histogram of the average MSE with samples from each iteration for $d = 5$. The result is shown in figure Figure 9. We may interpret these histograms (if they were to be normalised) as a probability distribution, where the expectation value of the MSE for the train and test data is the MSE which corresponds to the highest probability. As discussed in section section 2.3.1,

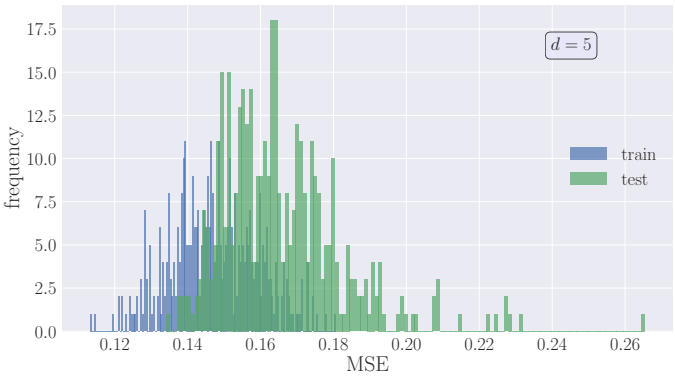

Fig. 9. Histogram of average MSE for train and test data for polynomial degree $d = 5$ for the OLS analysis of the Franke function ($N = 20$, $\eta = 0.1$), generated with a $k = 400$ bootstrap. The train data seem to have a slightly lower MSE than the test data. This is consistent with Figure 6 for $d = 5$.

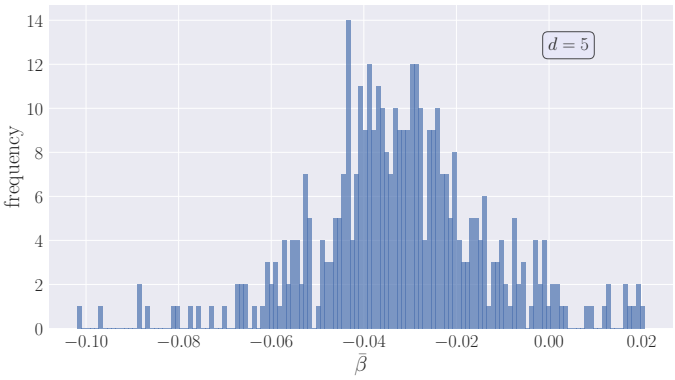

Fig. 10. Histogram of average feature parameters $\boldsymbol{\beta}$ for polynomial degree $d = 5$ for the OLS analysis of the Franke function ($N = 20$, $\eta = 0.1$), generated with a $k = 400$ bootstrap. Since all the features are scaled to mean $\mu = 0$, we expect the average features parameter to be close to 0, which is consistent both with this histogram, but also with Figure 3 for $d = 5$.

if $k \to \infty$ then this probability distribution approaches the normal distribution and the expectation value would be the mean. Translating this to the specific case at hand here, the most likely MSE value can interpreted as the MSE value with the highest frequency for both the train and test data. If we compare this plot to Figure 6 we see that the numerical values of MSEs coincide, with the test MSE being slightly higher than the test MSE.

We also investigate how the mean of the feature parameters $\boldsymbol{\beta}$ behave while bootstrapping for $d = 5$. Since both the design matrix X and the data $\mathbf{y}$ are scaled to have mean $\mu = 0$, we also expect the mean of the feature parameters to be centred around 0, as emphasised in figure Figure 3 where 0 appear to be the by-eye mean. The resulting histogram is shown in figure Figure 10, where we see a frequency distribution centred at around 0, as expected.

3.5.2. Ridge

We continue our analysis of the model using $d = 5$, but now for Ridge regression, where the free parameter will be the penalty term λ. The technicalities of this method is explained in section section 2.2.3. A natural place to start this

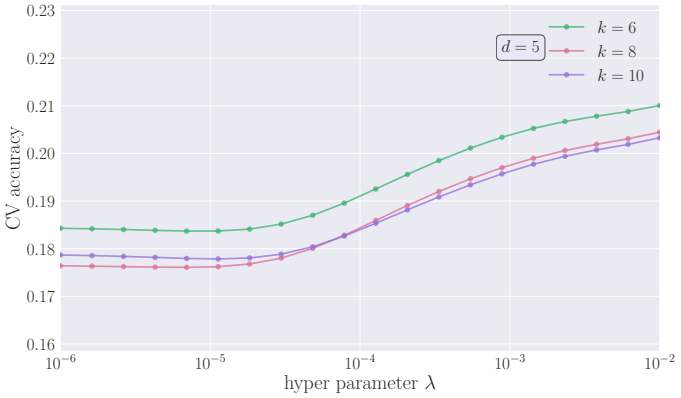

Fig. 11. MSE for train and test data for polynomial of degree $d = 5$, as function of the penalty parameter λ for the Ridge analysis of the Franke function ($N = 20, \eta = 0.1$) generated with an 8-fold cross-validation. For small λ the train MSE and test MSE are close the OLS values, but both increase when we increase λ.

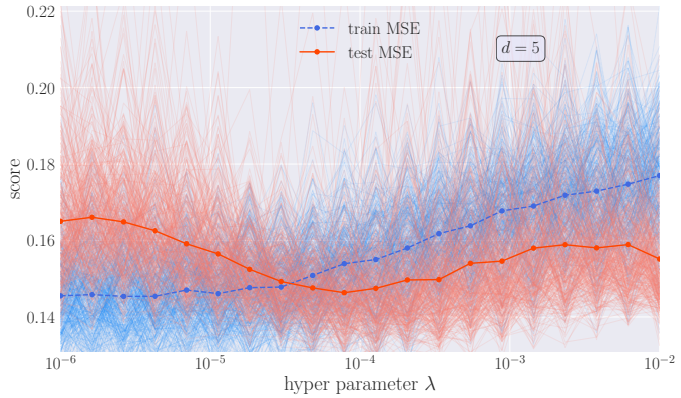

Fig. 12. MSE for train and test data for polynomial of degree $d = 5$, as a function of the penalty parameter λ for the Ridge analysis of the Franke function ($N = 20, \eta = 0.1$) generated using a $k = 400$ bootstrap. We see that the train MSE increase in a similar way as in Figure 11, while the test MSE decreases to a minimum and remain low, reaching a minimum at around $\lambda \approx 10^{-4}$

analysis is to see how the MSE change for different penalty parameters λ, and we to this by using the two resampling techniques previously discussed. In Figure 11 we see how the MSE for both the train and the test data change as a function of λ. This plot was generated using an 8-fold cross-validation. Since $\lambda = 0$ correspond to OLS analysis, we would expect the MSEs for low λ to be fairly similar to the OLS values. If we compare this with Figure 6 and Figure 8 we see that this is indeed the case when $d = 5$. We also observe that both the test and train MSEs increase when we increase λ.

However, when we repeat the analysis but generate the plot using a $k = 400$ bootstrap instead, as shown in Figure 12, we see that the test MSE decrease to a minimum and remain low when we increase λ. We also note that the λ which minimised the test MSE is found numerically to be $\lambda_{\text{min}}^{\text{Ridge}} = 7.85 \cdot 10^{-5}$ given to two significant figures.

We perform the bias-variance analysis with a $k = 400$ bootstrap which results in the plot shown in figure Figure 13. We see that the variance remains low for this λ range, while the residual error follows the bias closely. They both

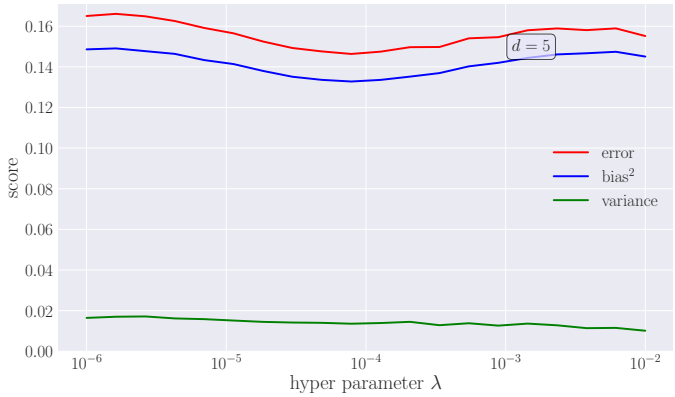

Fig. 13. Bias, variance and residual error for polynomial of degree $d = 5$, as a function of the penalty parameter λ for the Ridge analysis of the Franke function ($N = 20, \eta = 0.1$) generated with a $k = 400$ bootstrap. We see that the variance remain low for this rang, and the bias and error terms follow each other. They both seem to reach a local minimum at $\lambda \approx 10^{-4}$.

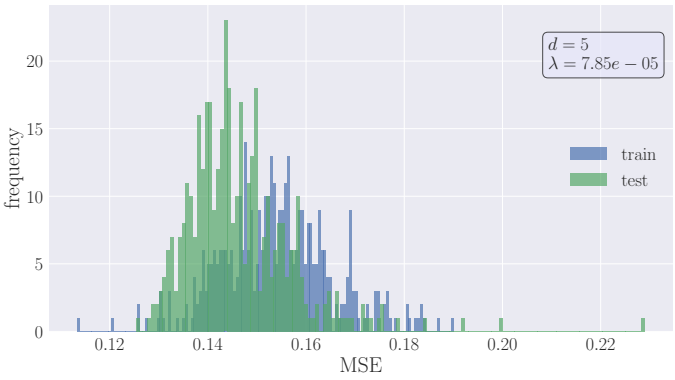

Fig. 14. Histogram of average MSE for train and test data for polynomial degree $d = 5$ for the Ridge analysis of the Franke function ($N = 20$, $\eta = 0.1$), generated with a $k = 400$ bootstrap and $\lambda = 8.86 \cdot 10^{-4}$. The test data has a lower MSE than the train data. This is consistent with Figure 12 for this value of λ.

seem to reach a minimum at around $\lambda \approx 10^{-4}$ which coincides very well with $\lambda_{\text{min}}^{\text{Ridge}}$. This is expected since Ridge regression is a biased model.

We investigate the model further by using $\lambda_{\text{min}}^{\text{Ridge}}$ and plot the frequencies of the average MSE for the train and test data as a histogram using a $k = 400$ bootstrap, shown in Figure 14. From this we see that the test MSE is slightly lower that the train MSE which is in accordance with Figure 12

We may also plot the plot the histogram of the average of the feature parameters $\boldsymbol{\beta}$. We would perhaps guess that these values should be centered around 0, using similar arguments as for Figure 10. However, the penalty parameters λ will shrink $\boldsymbol{\beta}$-values corresponding to small singular values in the design matrix X. It therefore give emphasise to the $\boldsymbol{\beta}$-values corresponding to large singular values of the design matrix. This may explain the skew towards negative parameter values which we observe in Figure 15 where we plot the abovementioned histogram.

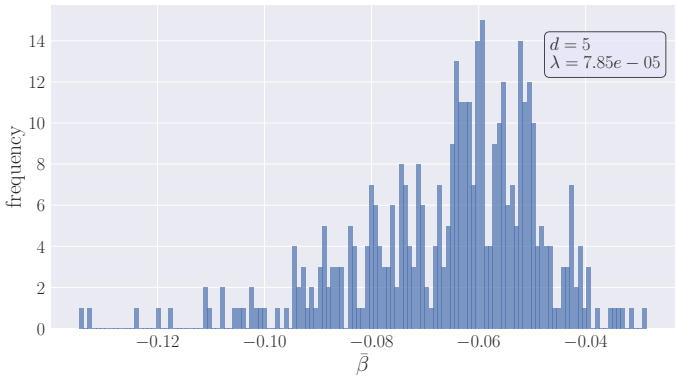

Fig. 15. Histogram of average feature parameters $\boldsymbol{\beta}$ for polynomial degree $d = 5$ for the Ridge analysis of the Franke function ($N = 20$, $\eta = 0.1$), generated with a $k = 400$ bootstrap and $\lambda = 8.86 \cdot 10^{-4}$. With the same reasoning as in Figure 10 we would expect these to be centred around 0. However, the penalty parameter shrinks certain beta values, which may explain the skew observed here.

3.5.3. Lasso

We now make a similar analysis, but we use the Lasso regression analysis instead, with a slightly different approach. Instead of looking for the optimal penalty parameter λ for a given polynomial degree d, we seek to find them at them simultaneously, by performing a grid search on a grid of MSE values, which are function of both λ and d. The result is shown in Figure 16, where we perform an 8-fold cross-validation and present the resulting MSE as a heatmap. What we draw from this heatmap is that the there seem to be an optimal band across polynomial degrees for the penalty parameter $\lambda = 2.15 \cdot 10^{-5}$. Our grid search yield the border value of the polynomial degree($d = 14$) which could indicate that the true best fit is of a higher polynomial degree. However, due to the fact that a polynomial of degree $d = 14$ is high for a $N = 20$ grid, and it becomes very computationally expensive to run the Lasso analysis on such models, we restrict ourselves to not go beyond this. On the other hand, higher polynomial degrees seem plausible with the Lasso regression scheme, as the penalty parameter is able to drive unimportant features to zero, reducing the dimensionality of the model. We thus take a closer look at the results of such a model ($d = 14$).

Figure 17 shows the MSE for the test data predictions for two different cross-validations: 6-fold and 8-fold. This plot confirms that there is an optimal value for $\lambda \in [10^{-5}, 10^{-4}]$ that minimise the MSE, and we have reason to believe it is the same λ that we found in Figure 16.

If we then consider the MSE found by a $k = 10$ bootstrap as seen in Figure 18 for larger λ we see that there is perhaps a more optimal λ around 10^{-3} which yield a lower MSE of ≈ 0.15.

Figure 19 show the bias variance trade off for the Lasso regression for $d = 14$ made with the same $k = 10$ bootstrap, for computational efficiency. We draw from this that the Lasso regression scheme has a low variance, but high bias for the λ values considered, with perhaps the same optimal λ at around 10^{-3}.

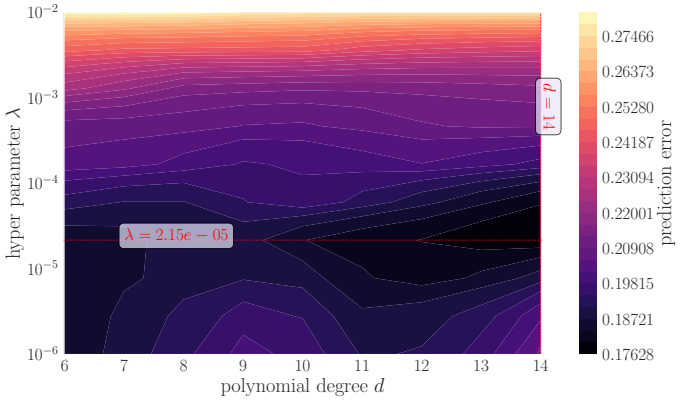

Fig. 16. Heatmap of the MSE as function of both polynomial degree $d \in [6, 14]$ and penalty parameter λ for the Lasso analysis of the Franke function ($N = 20, \eta = 0.1$), generated with and 8-fold cross-validation.

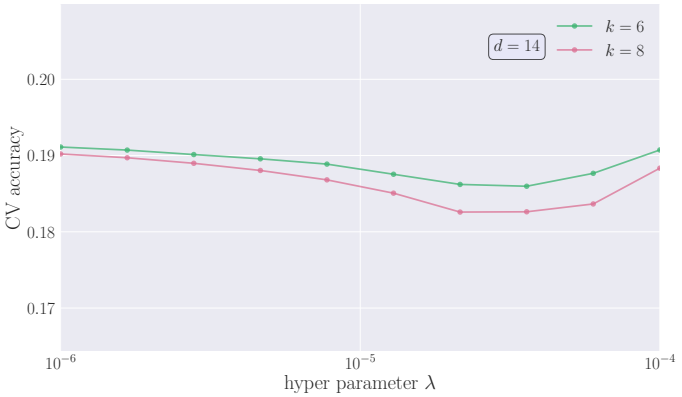

Fig. 17. MSE for test data for polynomial of degree $d = 14$, as function of the penalty parameter λ for the Lasso analysis of the Franke function ($N = 20, \eta = 0.1$) generated with both a 6-fold and 8-fold cross-validation. For small λ the train MSE and test MSE are close the OLS values.

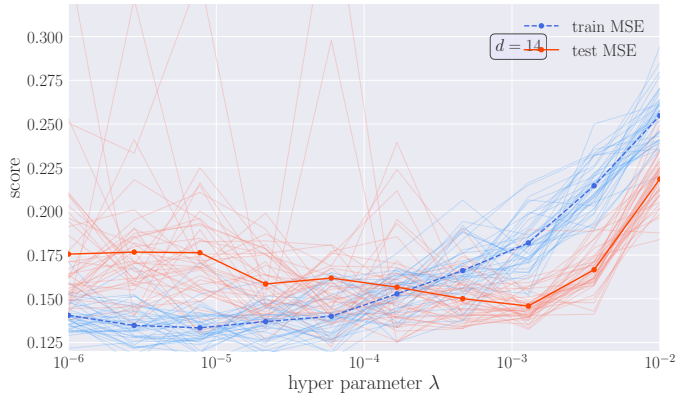

Fig. 18. MSE for train and test data for polynomial of degree $d = 9$, as a function of the penalty parameter λ for the Lasso analysis of the Franke function ($N = 20, \eta = 0.1$) generated using a $k = 10$ bootstrap.

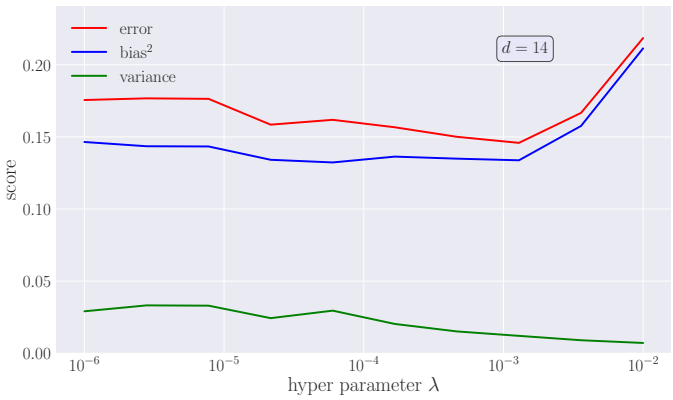

Fig. 19. Bias, variance and residual error for polynomial of degree $d = 14$, as a function of the penalty parameter λ for the Lasso analysis of the Franke function ($N = 20, \eta = 0.1$) generated with a $k = 10$ bootstrap.

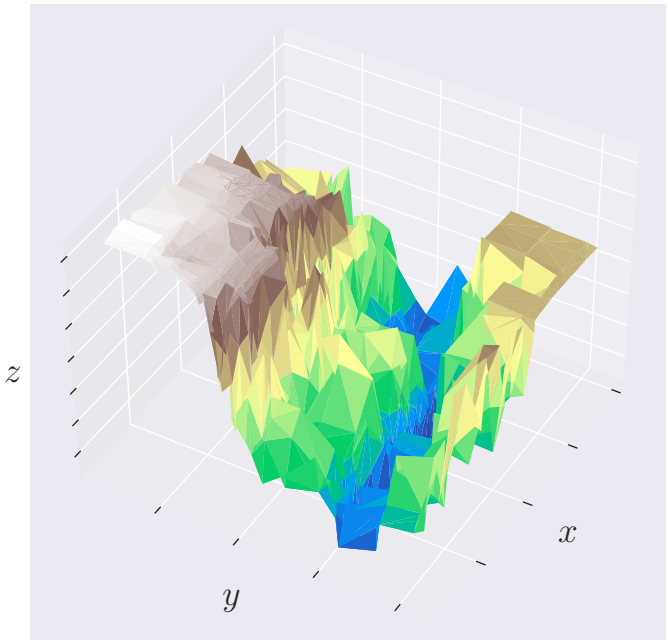

Fig. 20. Scaled terrain data representing a part of the Grand Canyon.

3.6. Regression analysis for real terrain data

By the courtesy of (Gelogigical Survey (U.S.) & EROS Data Center 2000), we were able to extract the terrain data of a part of the Crand Ganyon in Arizona, United States. To properly retrieve the geographical coordinates of the map is somewhat circumstantial and utterly unimportant to this task, we to not mind these in this project. We read the GeoTIFF and choose an area from which we extract a coarse grid. In particular, the original cutout area in question consists of $N = 900$ points in the x and y direction each. In order to speed up our codes, we use only the data from every 30^{th} x- and y-point, leaving us with $N = 30$ in each direction. We now have $N^2 = 900$ z-points representing relative altitude, of which 20% are saved for verification. After splitting randomly, we scale the data using the z-score normalisation explained in section 3.4. It would not make sense to concern ourselves with the vertical units (the actual altitude), as we already have omitted the horizontal units. The scaled data is what we aspire to work with and is presented in its completeness in Figure 20.

Moving forward, we essentially repeat the analysis in section 3.5 for the terrain data. Starting with the OLS method, we expect the need of a higher order polynomial than for the Franke funtion.

3.6.1. OLS

We perform the least squares fitting as elaborated in section 2.2.2 on the training data for polynomials of order $d = 1, 2, ..., 20$. For $d \leq 7$, the resulting $\hat{\boldsymbol{\beta}}^{\text{OLS}}$ is plotted in Figure 21, where we have grouped the parameters such that $\beta^{(d_i)}$ is the mean of the elements belonging to the same order d_i, i.e. $\beta^{(1)} = 1/2(\beta_1 + \beta_2)$, $\beta^{(2)} = 1/3(\beta_3 + \beta_4 + \beta_5)$, etc.

After performing a simple k-fold cross-validation to optimise the model, we see from Figure 22 that it clearly prefers order $d = 6$. We continue with the bootstrap resampling technique to verify that this complexity is preferable for our test data as well. In Figure 23 we see the expected behavoiur of the training and prediction error per increasing model complexity.

The choice $d = 6$ still seems sensible, but let us decompose the bootstrap prediction error into bias and variance as in section XXXX. The plot in Figure 24 points to a conclusion that OLS-estimations with $d \geq 7$ is over-fitting the data. However, that might not be the case if we are able to

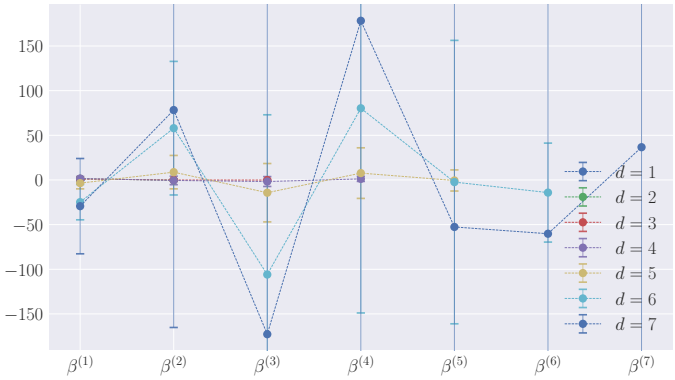

Fig. 21. Numerical value of the mean of the feature parameters β for each polynomial degree, with 1σ error bars, for polynomials up to order $d = 7$ for the OLS analysis of the terrain data. We note that the parameters for similar features tend to have the same sign across models, however there is an increase in parameter magnitude.

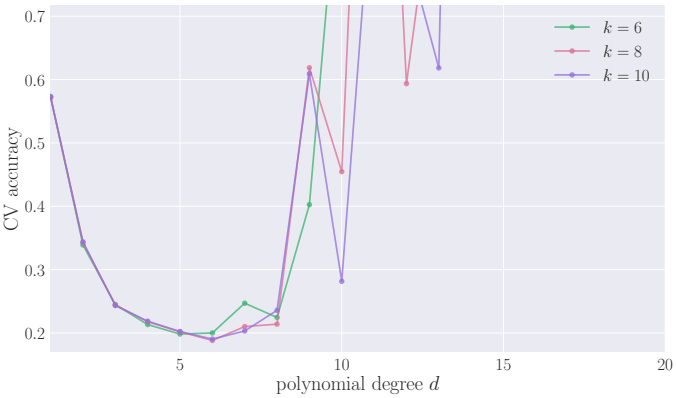

Fig. 22. CV accuracy FIXME, maybe remove this

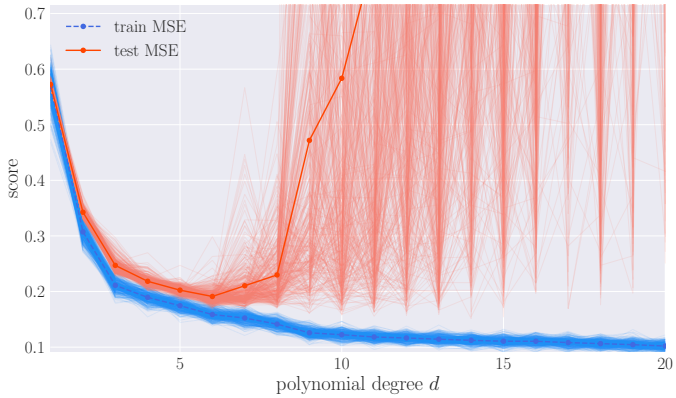

Fig. 23. Bootstrap training and prediction errors FIXME.

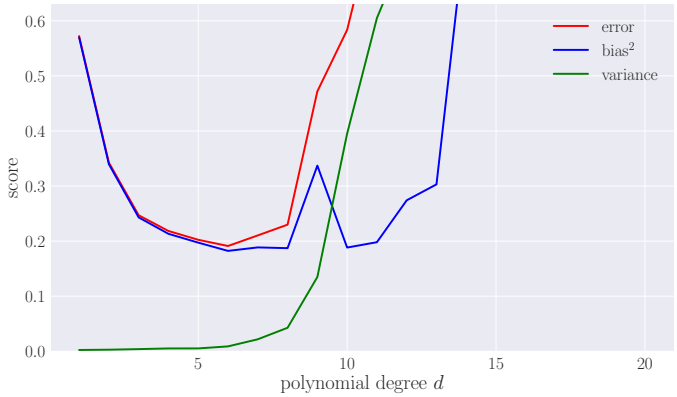

Fig. 24. FIXME.

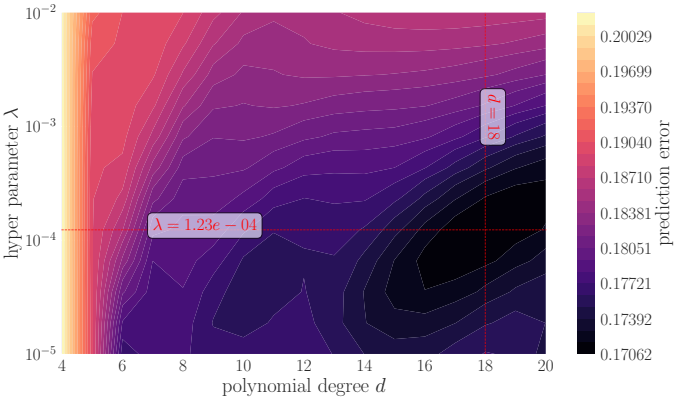

Fig. 25. Heatmap Ridge

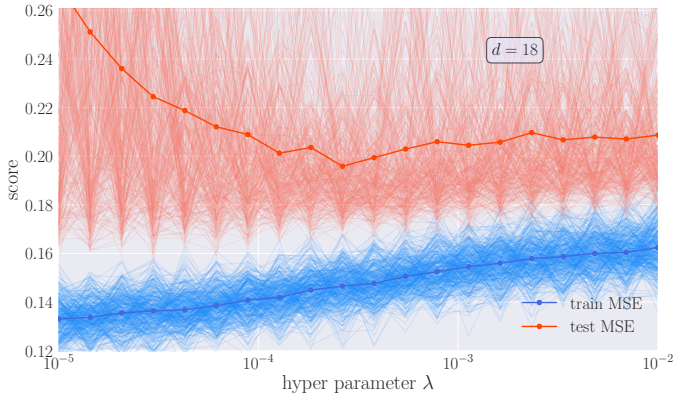

Fig. 26. MSE RIDGE BS

penalise the β_j's. We move on to the penalised estimation models.

3.6.2. Ridge

The search for an optimal model begins with a grid search where we limit ourselves to $d \in [4, 20]$ and $\lambda = 10^{-\gamma}, 5 \geq \gamma \geq 2$. The grid search is somewhat computationally expensive, at least for large d and small λ.

The error heatmap in Figure 25 favourises the combination $d = 18, \lambda = 1.23 \cdot 10^{-4}$. In an attempt to verify this model, we do the bootstrap for this polynomial degree and the same hyper parameters. The resulting train and test MSE is shown in Figure 26.

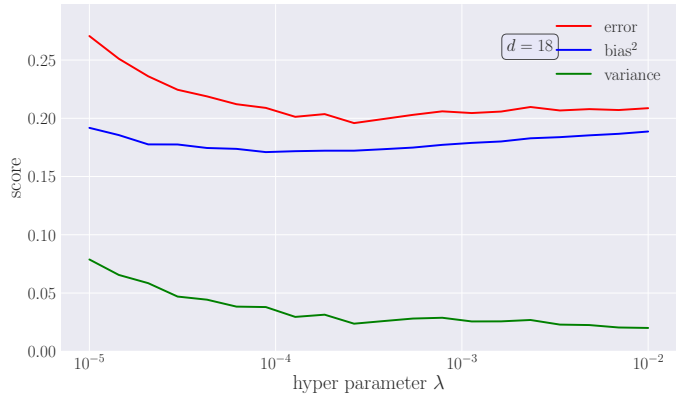

Fig. 27. Bias-variance trade off Ridge

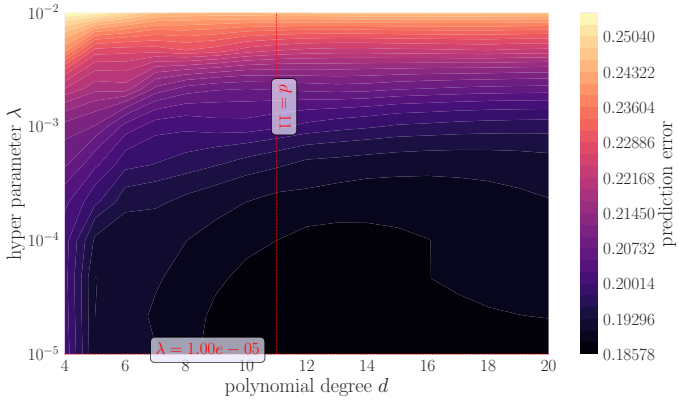

Fig. 28. Heatmap Lasso

We investigate the bias-variance tradeoff by the same bootstrap session. The tradeoff is shown in Figure 27. It shows nothing to indicate precisely which hyper parameter is best, but it does not contradict the model found by the grid search.

UNSURE ABOUT THIS:

There are actually models with $d = 15 \pm 1, \lambda \sim 10^{-8}$ which give cross-validation accuracy $\lesssim 0.16$ for various choices of k. However, when confirming these models using bootstrap, the prediciton errors grow huge, as do both the variance and squared bias, in the regime of $\lambda \sim 10^{-8}$. There is a sudden drop in error and variance for $\lambda \simeq 8.5 \cdot 10^{-9}$ for $d = 15$, but this now seems random.

3.6.3. Lasso

The same strategy is used with Lasso regression. This is much more frustrating as the computational expenses (?) of minimising the Lasso cost function are much higher than for the two others. A grid search indicates that we need $\lambda \geq 10^{-5}$ for $d = 11$ in order to get cross-validation generalisation errors down to 0.19. This is illustrated in Figure 28.

There is little point in continuing with validation of this model. In this paper, we omit the rest of the Lasso analysis due to

(i) the computational expense of the bootstrap and Lasso algorithm combined is very large, and even larger for smaller λ's, and

(ii) a wish to reduce the number of figures for the sake of a legibility.

4. Conclusion

We present our final models for the Franke function and the Grand Canyon terrain.

4.1. Franke function

After performing the OLS analysis on the Franke function, it quickly became clear that a two-dimensional polynomial of order $d = 5$ was the best model, i.e. yielding the lowest MSE. This can be seen most clearly from Figure 6 and Figure 7 we compare model of both higher and lower polynomial degree. A rough by-eye estimate of the MSE in these two cases is MSE ≈ 0.16 for the test data. This is confirmed in Figure 8 and Figure 9, where the former yield a slightly higher MSE value.

For the Ridge analysis we just assume that the polynomial degree found for OLS will yield the best result, and seek to find the optimal λ only. One could argue that this is naive, so we should perhaps have allowed for the polynomial degree to change when performing Ridge analysis. However, with $d = 5$, the optimal penalty parameter is found to be $\lambda^{\text{Ridge}} = 7.85 \cdot 10^{-5}$ given to two significant figures. We have from Figure 12 and Figure 14 that an approximate value of the MSE found for this model is MSE ≈ 0.14 for the test data.

For the Lasso analysis we try to find a new optimal polynomial degree, which result in the maximum value of our grid $d = 14$, which is not necessarily a bad model for a $N = 20$ grid, since the penalty parameter drives unimportant features to zero. However, according to Figure 17 and Figure 19 there seem to be an optimal $\lambda \in [10^{-5}, 10^{-4}]$. However, the MSE value of ≈ 0.15 of the same order for both OLS and Ridge.

One could argue that Ridge yield a slightly lower MSE than OLS. However, these values are highly dependant on the method by which they were found, and the k-number in both bootstrap and cross-validation. The only sensible conclusion is thus that they yield approximately the same result. While OLS is less computationally expensive, and gives a simpler model altogether we further conclude that OLS with a polynomial degree $d^{\text{OLS}} = 5$ is the model that fits the data best.

We check this and plot the prediction model alongside the scaled data points. This can be seen in Figure 29 and the model is indeed good.

4.2. Grand Canyon terrain

We got two models that in terms of MSE were almost equally good. The OLS model has $d^{\text{OLS}} = 6$ and the Ridge model has $d^{\text{Ridge}} = 18$ and $\lambda^{\text{Ridge}} = 1.23 \cdot 10^{-4}$. The analysis yielded very similar prediciton errors for the two, see Figure 23 and Figure 26. However, as introduced in section 2.2.3, the variance of the β_j's are generally lower for the Ridge scheme than for the OLS scheme, and this is why we choose the former. The resulting terrain prediction is shown in Figure 30.

The terrain is definitely recognisable to the Grand Canyon (Figure 20, mind the angle). There is a tendency to flatten out by the edges as opposed to continuing downwards, which was the typical case with the OLS-models.

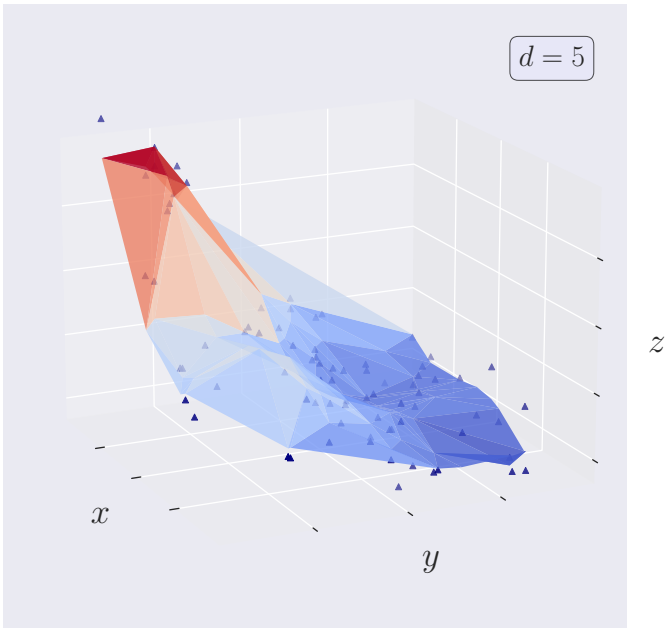

Fig. 29. FIXME.
xw

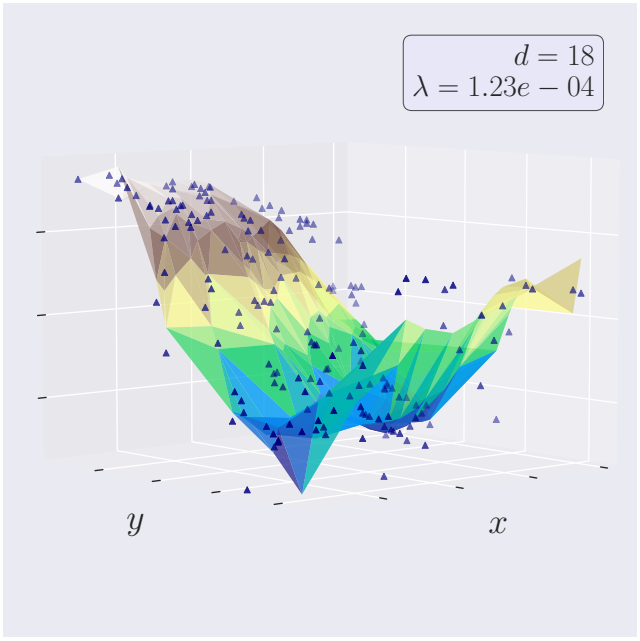

Fig. 30. The triangular points are the scaled test data points, whilst the surface represents the corresponding prediction we get from a Ridge estimation with $d^{\text{Ridge}} = 18$ and $\lambda^{\text{Ridge}} = 1.23 \cdot 10^{-4}$.

Code availability

The code is available on GitHub at https://github.com/Johanmkr/FYS-STK4155colab.

List of Figures

References

David C. Lay, Steven R. Lay, J. J. M. 2016, Linear Algebra and its Applications (Pearson Education Limited), 432–441
Gelogigical Survey (U.S.) & EROS Data Center. 2000, Earth Explorer, https://earthexplorer.usgs.gov
Hjorth-Jensen, M. 2021, Applied Data Analysis and Machine Learning (Jupyter Book)