Astronomy & Astrophysics manuscript no. project1
October 5, 2022
©ESO 2022

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 Computational Science, University of Oslo, Norway

October 5, 2022

ABSTRACT

The ...

1. Introduction

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)[1] described in eq. (1), and the R2-score[2], described in eq. (2).

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

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

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, lets 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 penalized 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:

[1] Describe MSE briefly
[2] describe R2-score briefly

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

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 minimizing 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 minimizing 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

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.

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[3]. 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

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

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

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

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

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

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

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*.

CAN ADD MORE HERE IF WE WANT

2.2.4. Lasso regression

If we add the penalty term $\lambda\|\boldsymbol{\beta}\|_1$, now using the ℓ^1-norm[4], 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.

2.3. 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

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:

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

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

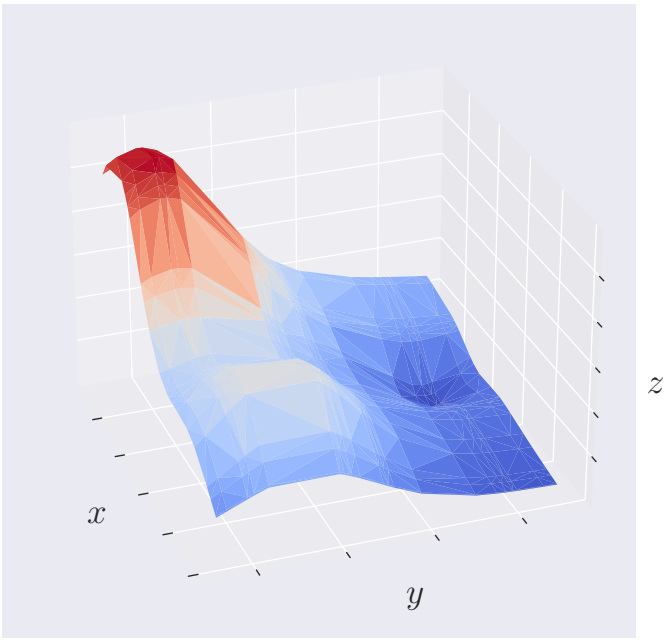

Fig. 1. The Franke function plotted on a grid where $N = 20$ and $\eta = 0$. Since we scale the data, the z becomes arbitrary and we choose to leave it out. The important information is the shape of the graph which is smooth when there is no noise.

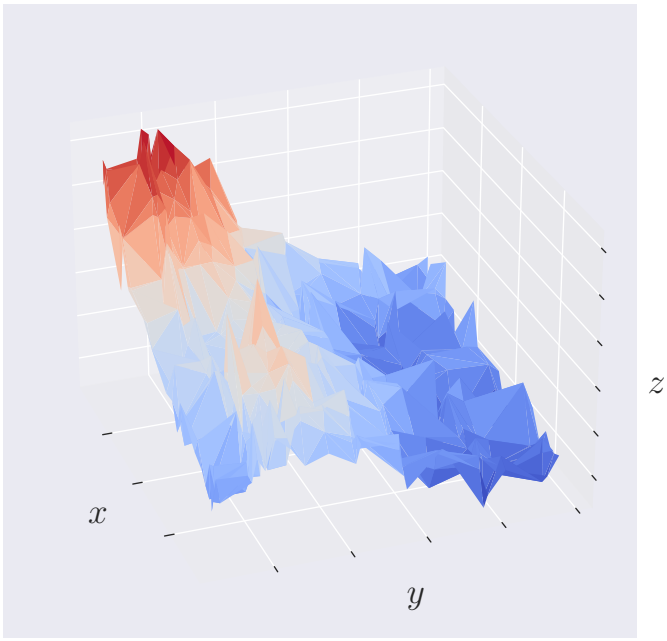

Fig. 2. The Franke function with added noise plotted on a grid where $N = 20$ and $\eta = 0.1$. Since we scale the data, the z becomes arbitrary and we choose to leave it out. The important information is the shape of the graph which becomes quite ragged even for this small level of noise

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). From this we set up the design matrix with the following terms (THIS IS WRONG; CORRECT THIS):

$$X_{ij} = \sum_{l=1}^{d} \sum_{k=0}^{l} x^{l-k} y^k \tag{12}$$

The design matrix is set up without an intercept (constant term) (WHY?). Since we need to set up to design matrices, they account for a different amount of data points, and thus n will be different between the two, but p is the same.

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 standardize the features as much as possible to ensure equal (relative) emphasize 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.

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

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 $d = 5$.

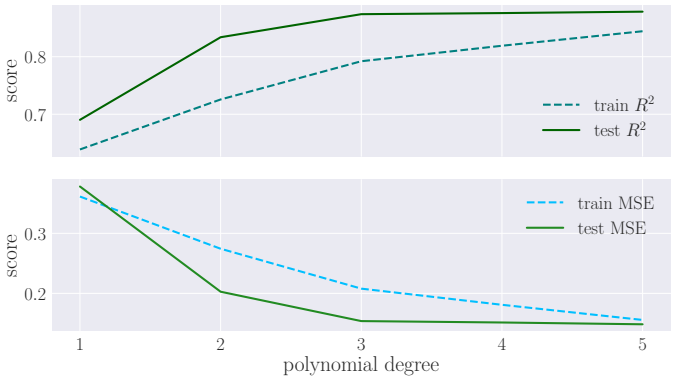

Fig. 4. MSE and R^2 scores for train and test data for polynomials up to order $d = 5$ for the OLS analysis of the Franke function ($N = 20$, $\eta = 0.1$). We see that for the polynomial degrees present, the MSE decrease, while the R^2 increase towards 1 for increasing polynomial degree.

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 increase. 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 *overfitting*: 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 $k = 400$

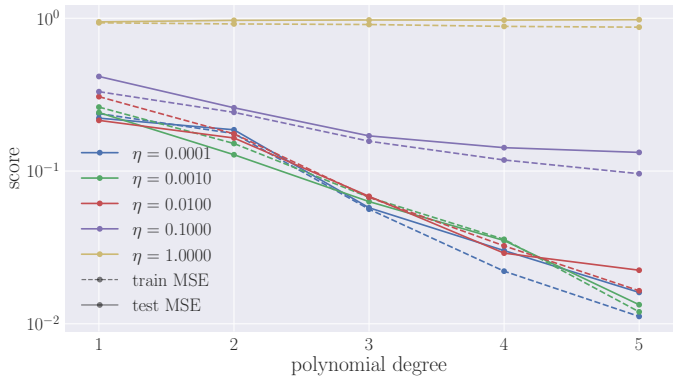

Fig. 5. MSE for train and test data for polynomials up to order $d = 5$ for the OLS analysis of the Franke function ($N = 20$, $\eta = 0.1$), for different noise parameters $\eta = 10^{\gamma}, \gamma \in [-4, -3, -2, -1, 0]$. The MSE is large for noisy function (large η), but decreasing with polynomial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

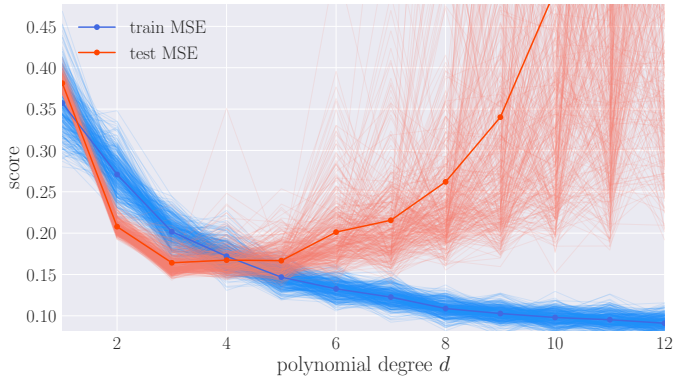

Fig. 6. MSE for train and test data for polynomials up to order $d = 12$ for the OLS analysis of the Franke function ($N = 20$, $\eta = 0.1$) generated with a $k = 400$ bootstrap. When we increase the model complexity up to order $d = 12$ we easily spot that the MSE for train data starts to diverge,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

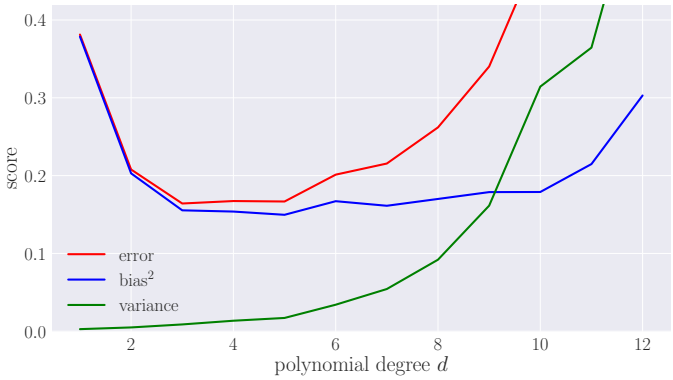

Fig. 7. Bias, variance and residual error for polynomials up to order $d = 12$ for the OLS analysis of the Franke function ($N = 20$, $\eta = 0.1$), generated with a $k = 400$ bootstrap. We see that for low polynomial degrees there is a high bias and low variance, the opposite for larger degrees, and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

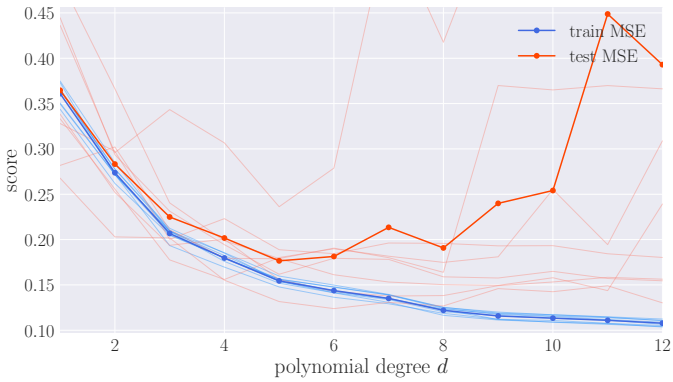

Fig. 8. MSE for train and test data for polynomials up to order $d = 12$ for the OLS analysis of the Franke function ($N = 20$, $\eta = 0.1$) generated with an 8−fold cross validation. We observe a similar behaviour with Figure 6 where the MSE for the test data reaches a minimum at $d = 5$ and then seem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

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 minimizing 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, 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 emphasized in figure Figure 3 where 0 appear to be the by-eye mean. The result-

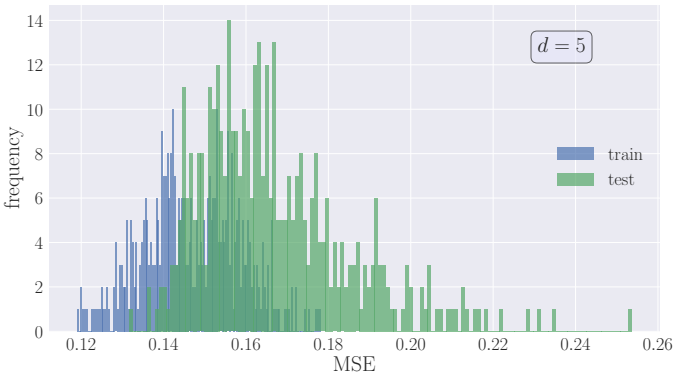

Fig. 9. Histogram of average MSE for train and test data for polynomial degree $d = 5$ for the OLS analysis of the Franke function ($N = 20$, $\eta = 0.1$), generated with a $k = 400$ bootstrap. The train data seem to have a slightly lower MSE than the test data. This is consistent with Figure 6 for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

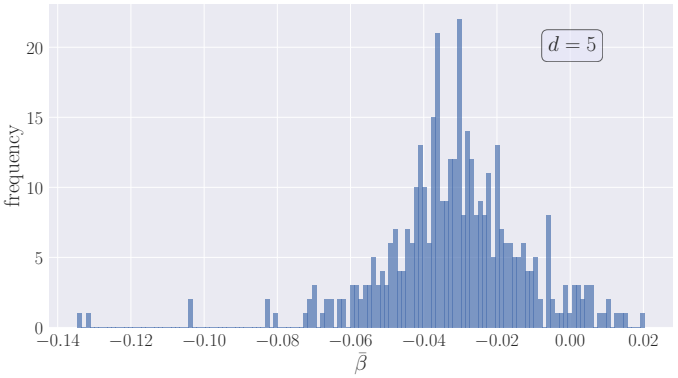

Fig. 10. Histogram of average feature parameters $\boldsymbol{\beta}$ for polynomial degree $d = 5$ for the OLS analysis of the Franke function ($N = 20$, $\eta = 0.1$), generated with a $k = 400$ bootstrap. Since all the features are scaled to mean $\mu = 0$, we expect the average features parameter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

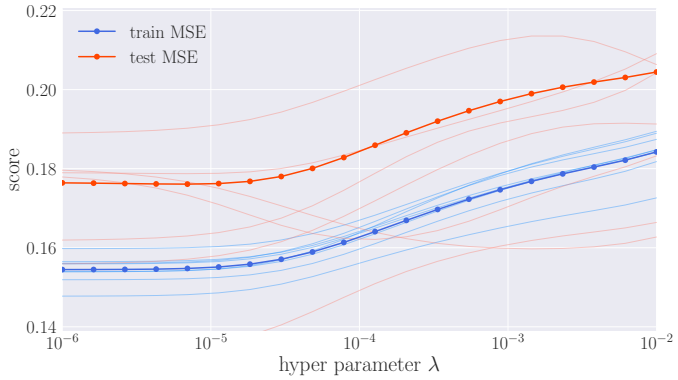

Fig. 11. Cross-validation for Ridge regression.

ing 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.

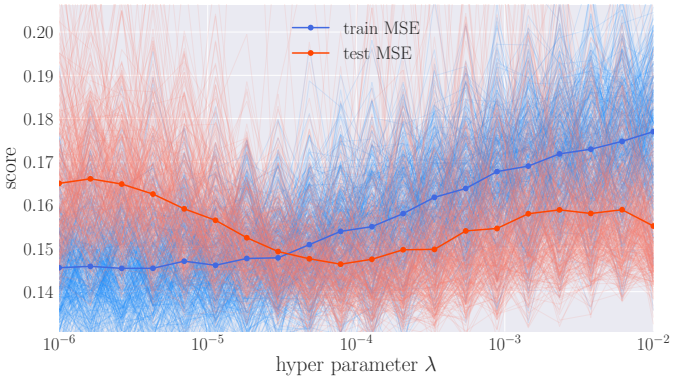

Fig. 12. Model complexity for Ridge regression. (RIKTIG PLOT?)

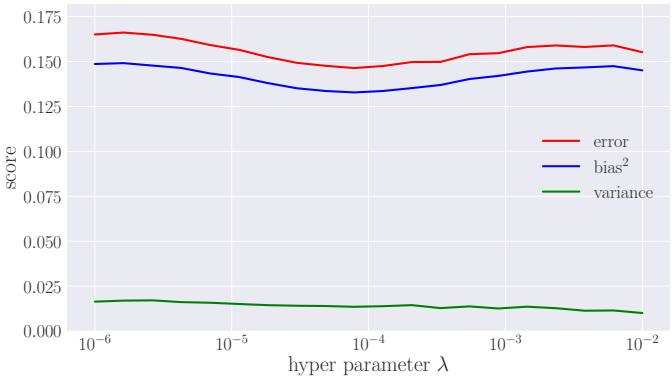

Fig. 13. Bias-variance for Ridge regression.

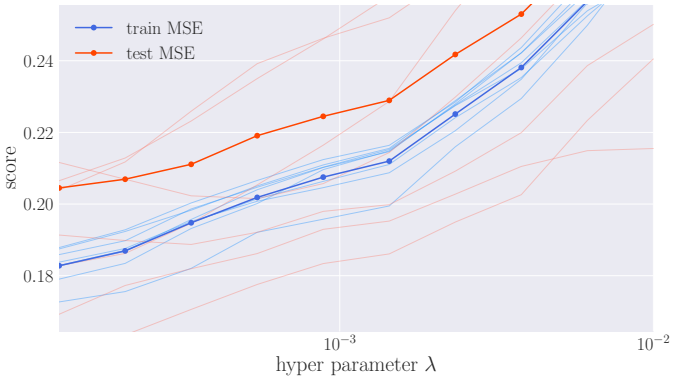

Fig. 14. Cross-validation for Lasso regression.

3.5.3. Lasso

3.6. Regression analysis for real terrain data

4. Conclusion

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

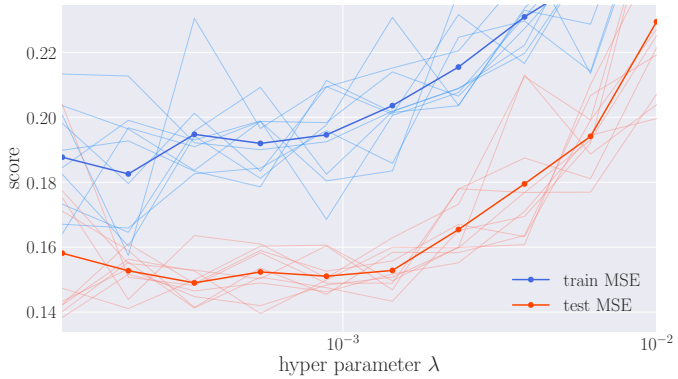

Fig. 15. Model complexity for Lasso regression. (RIKTIG PLOT?)

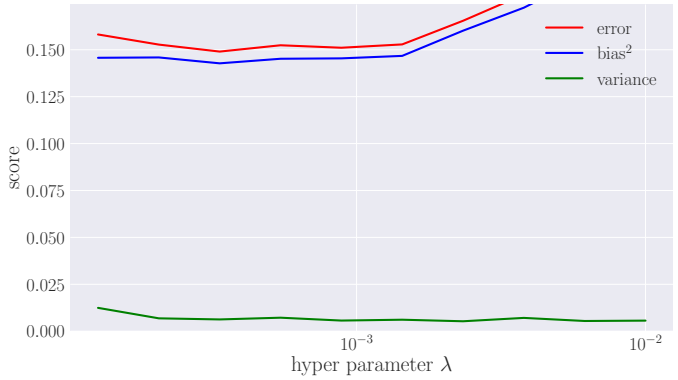

Fig. 16. Bias-variance for Lasso regression.

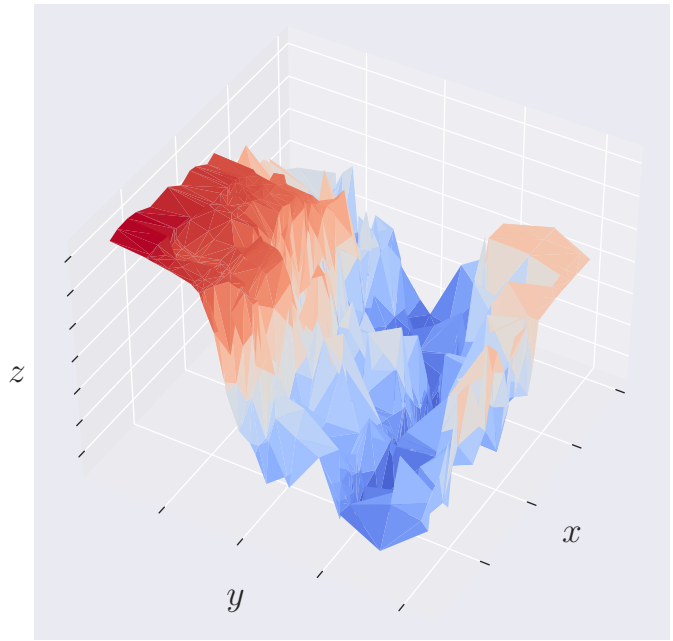

Fig. 17. Scaled terrain data representing a part of the Grand Canyon, downloaded from (Gelogigical Survey (U.S.) & EROS Data Center 2000). We use every 30$^{\text{th}}$ point in each direction of the original 300 $\times$ 300-cutout, leaving us with an xy-grid of 30 $\times$ 30 points.