Regression analysis and resampling methods

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

[1] Institute of Theoretical Astrophysics, University of Oslo, P.O. Box 1029 Blindern, N-0315 Oslo, Norway
[2] Center for Computational Science, University of Oslo, Norway

September 30, 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).

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

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

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**?**:

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

where Σ is a $n \times p$ diagonal matrix with the singular values of A as diagonal elements in descending order: $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \cdots \geq \sigma_r \geq 0$. V is a $p \times p$ square matrix whose columns are the eigenvectors of $A^{\intercal}A$, and thus is an orthonormal basis spanning $\mathbb{R}^p$. U is a $n \times n$ square matrix whose columns also form an orthonormal basis set that spans $\mathbb{R}^n$.

2.2. Principal component analysis

DO THIS LINALG P445.

2.3. 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 2.3.1) and two forms of penalized estimation (section 2.3.2 and section 2.3.3).

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:

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

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

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:

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

2.3.1. 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

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

...

2.3.2. 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,

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

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

COULD ADD SOMETHING HERE ABOUT SVD DECOMP AND PRINCIPAL VALUE to explain the behaviour of the penalisation.

2.3.3. 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,

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

$$\hat{\boldsymbol{\beta}}^{\mathrm{Lasso}} = \ldots$$

2.4. 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.

2.4.1. 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

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

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

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:

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

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.

2.4.2. 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:

$$\begin{aligned} f(x,y) &= \frac{3}{4}\exp\left\{-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right\} \\ &+ \frac{3}{4}\exp\left\{-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right\} \\ &+ \frac{1}{2}\exp\left\{-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right\} \\ &- \frac{1}{5}\exp\left\{-(9x-4)^2 - (9y-7)^2\right\}. \end{aligned} \tag{8}$$

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:

$$\begin{aligned} \mathbf{y} &= f(x,y) + \eta\mathcal{N}(0,1) \\ &= f(\mathbf{x}) + \varepsilon \end{aligned} \tag{9}$$

where $\mathbf{x} = (x,y)$. NECESSARY?From section section 2.3 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 $N = 40$. Figure Figure 2 show the same function, for the same points but now with an added noise of $\eta = 0.1$.

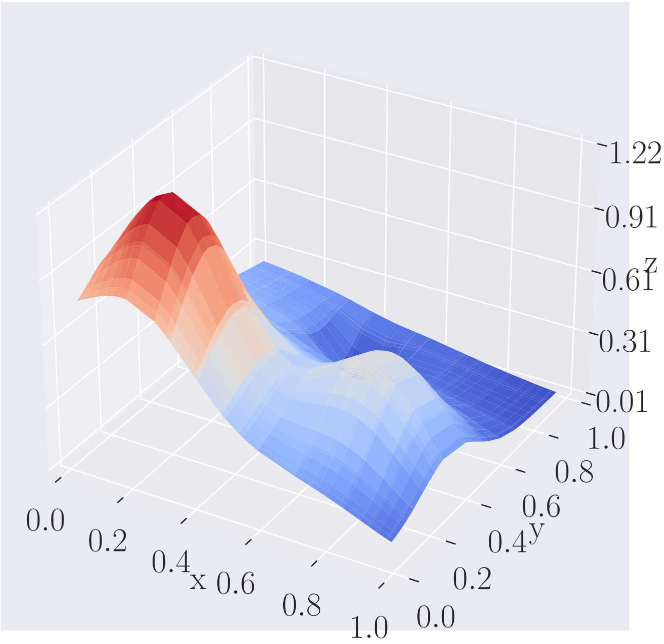

Fig. 1. The Franke function plotted on a grid where $N = 40$ and $\eta = 0$

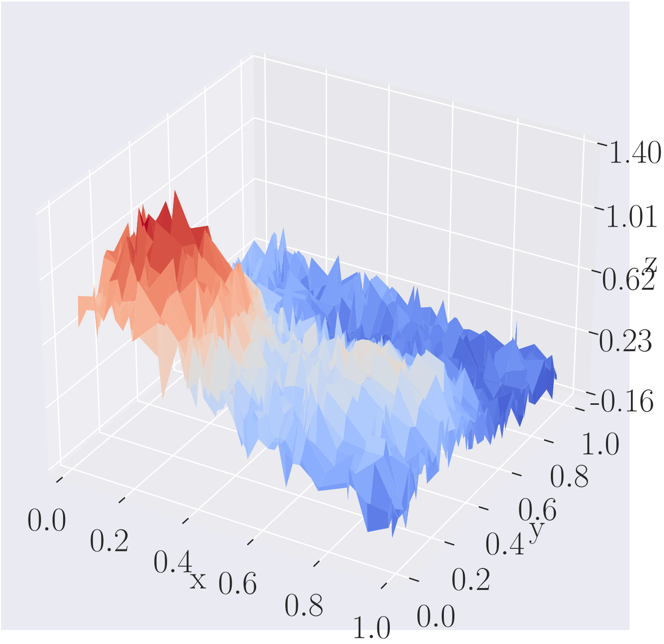

Fig. 2. The Franke function with added noise plotted on a grid where $N = 40$ and $\eta = 0.1$

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):

$$X_{ij} = \sum_{l=1}^{d} \sum_{k=0}^{l} x^{l-k} y^k \tag{10}$$

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:

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

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, $j \in [0, p]$:

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

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 X and the data $\mathbf{y}$,

MORE ON WHY EVERYTHING NEED TO BE SCALED WRT TRAIN Data

$$\hat{\boldsymbol{\beta}}' = \hat{\boldsymbol{\beta}} \frac{\sigma_X}{\sigma_{\mathbf{y}}} \implies \hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}' \frac{\sigma_{\mathbf{y}}}{\sigma_X}$$

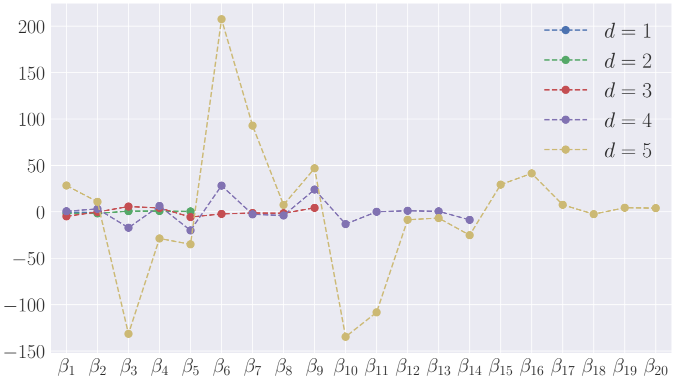

Fig. 3. Numerical value of the feature parameters β, with 1σ error bars, for polynomials up to order $d = 5$.

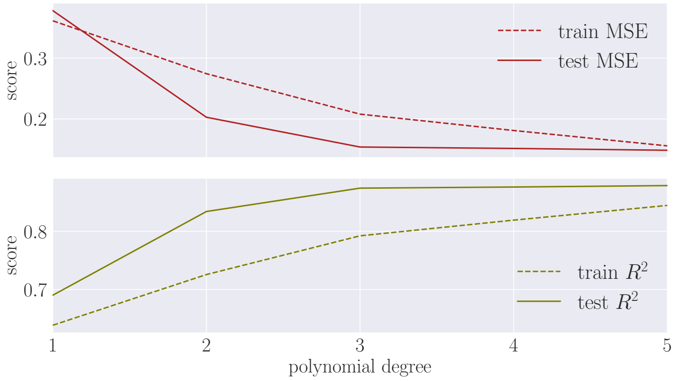

Fig. 4. SCORES

3.5. Regression analysis

3.5.1. OLS

We start by performing an Ordinary Least Square regression analysis, as outlined in section section 2.3.1, where we find $\hat{\boldsymbol{\beta}}^{\text{OLS}}$ from eq. (6) and $\text{Var}\big[\hat{\boldsymbol{\beta}}^{\text{OLS}}\big]$ from eq. (7). The error can be estimated with the standard deviation $\sigma_\beta = \text{Var}\big[\hat{\boldsymbol{\beta}}^{\text{OLS}}\big]^{1/2}$.

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.

Having found the optimal parameter $\hat{\boldsymbol{\beta}}^{\text{OLS}}$ we can make predictions by computing $\tilde{\mathbf{y}} = X\hat{\boldsymbol{\beta}}^{\text{OLS}}$ for 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. COMMENT ON PLOT

SOMETHING ABOUT NOSIE

Fig. 5. MSE error for different noise values η

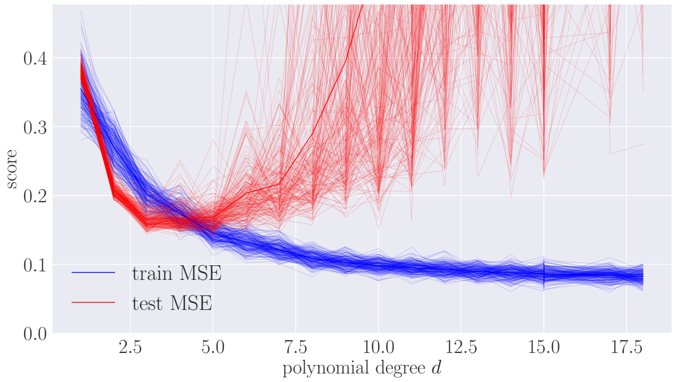

Fig. 6. Model complexity HASTIE ET AL.

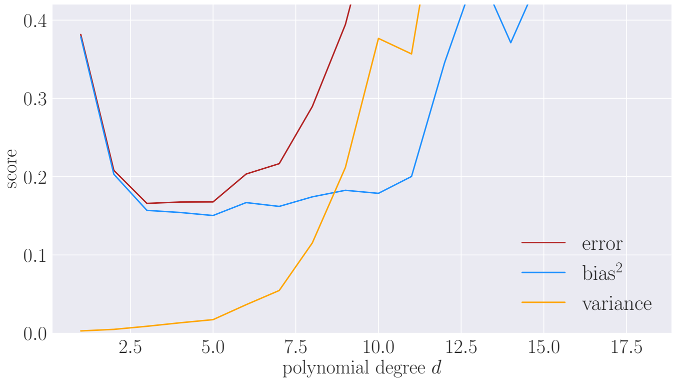

Fig. 7. Bias Variance

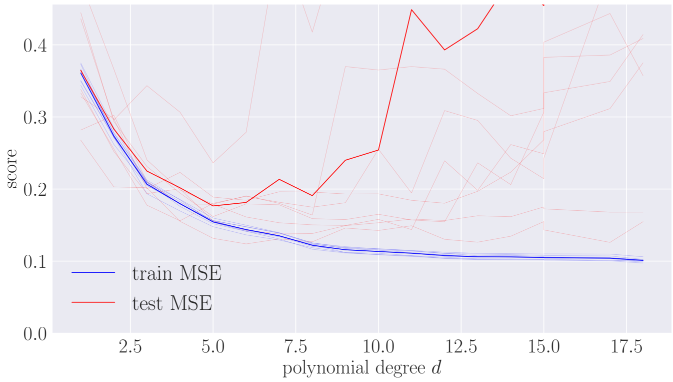

Fig. 8. Cross-validation

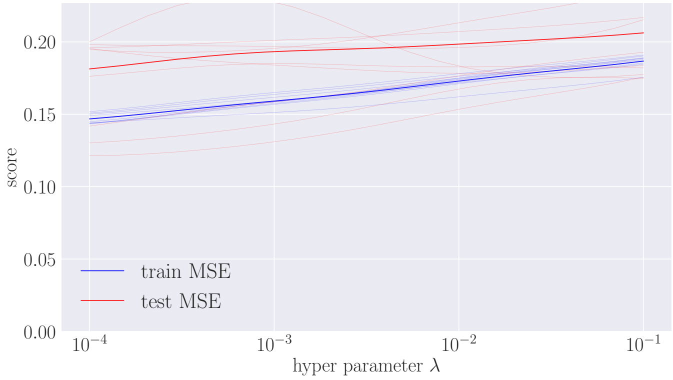

Fig. 9. Cross-validation for Ridge regression.

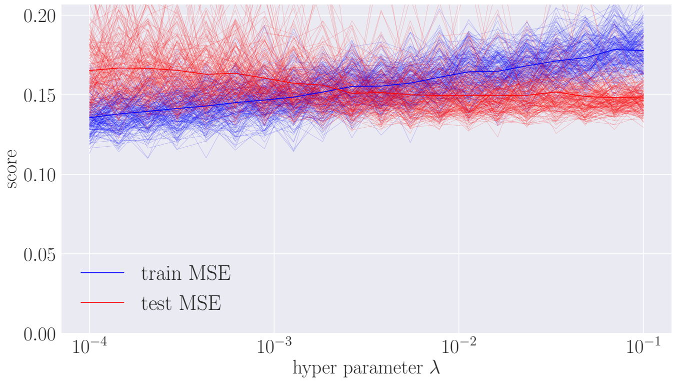

Fig. 10. Model complexity for Ridge regression.

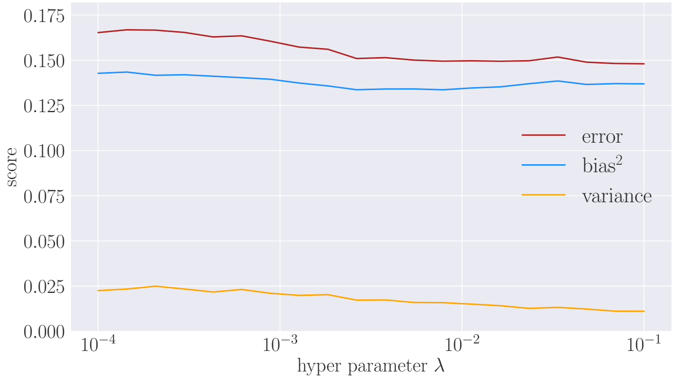

Fig. 11. Bias-variance for Ridge regression.

3.5.2. Ridge

3.5.3. Lasso

3.6. Applying best fit model

3.7. Real terrain data

4. Conclusion