# Application of Regression and Resampling on USGS Terrain Data

Isak O. Rukan* and Edvard B. Rørnes†
*Institute of Physics, University of Oslo,*
*0371 Oslo, Norway*
(Dated: October 7, 2024)

Regression models are widely used for analyzing and predicting data in many fields. We investigate and compare three regression models[a]: Ordinary Least Squares, Ridge, and Least Absolute Shrinkage and Selection Operator. These were applied and analyzed on a two-dimensional Franke function and later tested on real terrain data from the US Geological Survey Earth Explorer [1]. The models' performances are evaluated using metrics such as the Mean Squared Error and the coefficient of determination $R^2$. The bias-variance trade-off for OLS is studied in detail where it was found that the optimal polynomial degree for the Franke function and terrain data lies in the range $p \in \{4, 5, 6\}$ and $p \in \{x, y, z\}$ respectively. We used resampling techniques such as bootstrapping and cross-validation to probe the quality of the evaluations and determine the predictiveness and generalizability of the models. In our analysis the best performing method was Ridge regression with OLS following shortly after. LASSO performed appreciably worse than the other regression models, likely stemming from the polynomial coefficients not following a Laplace distribution.

## 1. INTRODUCTION

Regression models are essential tools in data analysis and prediction, particularly within the realm of physics. They are used to enable understanding of the relationships between different variables and improve our ability to create predictions. In this report we study and compare three common regression techniques: Ordinary Least Squares (OLS), Ridge and Least Absolute Shrinkage and Selection Operator (LASSO) regression. Each come with their unique strengths and weaknesses that make them suitable for different data sets.

OLS is the simplest and most foundational method which estimates relationship by minimizing the difference between observed and predicted values. A large downside with OLS is when there are many related variables as this can lead to unstable coefficient estimates [2]. Ridge regression can partially fix this issue by adding a penalty to large coefficients via a regulator $\lambda$, effectively allowing us to shrink the coefficients with a new input parameter. This in turn creates a more stable model in the event of correlated variables. LASSO regression takes this a step further by once again shrinking coefficients, but also setting some of them to zero. This allows LASSO to effectively choose important variables, which may be helpful when pursuing a simpler model.

To compare these regression model, we use a two-dimensional Franke function which allows us to test the performance of each under controlled conditions. Later we apply each model to real terrain data which is obtained from [1].

The models are evaluated by considering their Mean Squared Error (MSE) and coefficient of determination $R^2$. Further to make our evaluation more reliable, resampling techniques such as bootstrapping and cross-validation are used. These methods provide a quantitative description of how well each model performs on different data sets.

## 2. THEORY

All claims made in this section which require relatively lengthy derivations are derived in Appendix A. The general structure of all our models is that we have some data set $\{x_i, y_i\}$ where $i \in \{0, 1..., n-1\}$ where $x_i$ are independent variables whilst $y_i$ are dependent variables. The data is assumed to be described by

$$\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\varepsilon}, \tag{1}$$

where $f$ is some continuous function which takes $\boldsymbol{x}$ as input and $\boldsymbol{\varepsilon}$ is a normal distributed error $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2)$. The function $f$ will then be approximated with a model $\tilde{\boldsymbol{y}}$ in which we will consider a polynomial expansion with coefficients $\beta_i$ up to degree $p$:

$$\tilde{y}_i = \sum_{j=0}^{p} \beta_j x_i^j. \tag{2}$$

Defining the $n \times p$ design matrix $\boldsymbol{X}$ with elements $X_{ij} = (x_i)^j$ we can rewrite this as

$$\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}. \tag{3}$$

Further, each model will be defined with a different cost function $C(\boldsymbol{\beta})$ which we minimize to find the coefficients for each respective model.

The metrics MSE and $R^2$ which we use to analyze each

* INSERT EMAIL ISAK
† e.b.rornes@fys.uio.no
[a] Github

model are defined by:

$$\text{MSE}(\boldsymbol{y},\tilde{\boldsymbol{y}}) \equiv \frac{1}{n}\sum_{i=0}^{n-1}(y_i-\tilde{y}_i)^2, \tag{4}$$

$$R^2 \equiv 1 - \frac{\sum_{i=0}^{n-1}(y_i-\tilde{y}_i)^2}{\sum_{i=0}^{n-1}(y_i-\bar{y})^2}, \tag{5}$$

where $\bar{y} \equiv \frac{1}{n}\sum_{i=0}^{n-1} y_i$ is the mean value of $\boldsymbol{y}$. Here we can see that if $\tilde{\boldsymbol{y}} = \boldsymbol{y}$ then $R^2 = 1$ and MSE $= 0$ implies that we have perfect accuracy. Further if $\tilde{\boldsymbol{y}}$ is simply the mean then $R^2 = 0$ implying that our fit performs as well taking the mean. Finally if $R^2 < 0$ it means that our approximation is worse than simply taking the mean. Thus for a good fit we would expect a small MSE and an $R^2$ score closer to 1.

### 2.1. OLS

OLS is a primitive method used in linear regression to estimate coefficients of a linear model. The cost function in OLS is simply defined as the MSE:

$$C_{\text{OLS}}(\boldsymbol{\beta}) = \frac{1}{n}(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2. \tag{6}$$

As mentioned prior, the coefficients $\boldsymbol{\beta}$ are found by minimizing the cost function, i.e. taking the derivative w.r.t. $\boldsymbol{\beta}$. This results in

$$\boldsymbol{\beta}_{\text{OLS}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}, \tag{7}$$

which yields the model

$$\tilde{\boldsymbol{y}}_{\text{OLS}} = \boldsymbol{X}\boldsymbol{\beta}_{\text{OLS}}. \tag{8}$$

Assuming our data takes the form of (1) then the expectation value $\boldsymbol{y}$ is

$$\mathbb{E}(y_i) = \mathbb{E}(f(x_i)) = X_{ij}\beta_j \equiv \boldsymbol{X}_{i,*}\boldsymbol{\beta},$$

since $\mathbb{E}(\varepsilon_i) = 0$ follows from its definition. The variance of $\boldsymbol{y}$ is given by

$$\text{Var}(y_i) = \text{Var}(\varepsilon_i^2) = \sigma^2,$$

which yields $y_i \sim \mathcal{N}(\boldsymbol{X}_{i,*}\boldsymbol{\beta}, \sigma^2)$. The expectation value of the optimal parameters $\hat{\boldsymbol{\beta}}$ can be found to be

$$\mathbb{E}(\hat{\boldsymbol{\beta}}_{\text{OLS}}) = \boldsymbol{\beta},$$

with the variance

$$\text{Var}(\hat{\boldsymbol{\beta}}_{\text{OLS}}) = \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}.$$

### 2.2. Ridge

Ridge regression is an extension of OLS where define the cost function as a modified version of the OLS cost function with an added penalty term which is proportional to the coefficients $\beta_i^2$:

$$C_{\text{Ridge}}(\boldsymbol{\beta}) = C_{\text{OLS}}(\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^2. \tag{9}$$

Here $\lambda \geq 0$ is a regularization parameter, or commonly referred to as a hyperparameter, which controls the strength of this additional penalty. This regulator essentially drives the magnitude of these coefficients allowing for more tweaking in the parameter space. This parametrization of course includes the constraint that $\boldsymbol{\beta}^2 \leq t$ for some $t < \infty$ such that we can choose our arbitrary parameter $\lambda \geq 0$ to be sufficiently small s.t. the cost function (9) does not diverge. The optimal parameters for Ridge regressions can again be found by the same process as for OLS:

$$\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{10}$$

Here we can see that the effect of adding this penalty term is essentially taking $(\boldsymbol{X}^T\boldsymbol{X})^{-1} \to (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}$ when compared to the OLS case. In the past this was generally the starting point for Ridge regression in the cases where the matrix $\boldsymbol{X}^T\boldsymbol{X}$ was not invertible. A direct way of seeing the effect of the regulator is by considering the singular value decomposition (SVD) of $\boldsymbol{X}$. Doing so one can show that

$$\tilde{\boldsymbol{y}}_{\text{Ridge}} = \sum_{j=0}^{p-1}\boldsymbol{u}_j\boldsymbol{u}_j^T\frac{\sigma_j^2}{\sigma_j^2+\lambda}\boldsymbol{y}, \tag{11}$$

where $\boldsymbol{u}_i$ is the $i$-th row (column?) of the orthogonal matrix $\boldsymbol{U}$ stemming from the SVD: $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}$. Since $\lambda \geq 0$ then this added factor compared to OLS is $\leq 1$. We can then see that Ridge regression effectively suppresses all the coefficients, thus $\lambda$ is often called the "shrinkage" factor.

### 2.3. LASSO

Similarly to Ridge, LASSO also includes a penalty factor. The cost function in this case is instead defined to be

$$C_{\text{LASSO}}(\boldsymbol{\beta}) = C_{\text{OLS}}(\boldsymbol{\beta}) + \lambda||\boldsymbol{\beta}||_1, \tag{12}$$

where

$$||\boldsymbol{\beta}||_k \equiv \left[\sum_{i=0}^{n-1}|\beta_i|^k\right]^{1/k}$$

is the $L^k$ norm of $\boldsymbol{\beta}$. Taking the derivative of (9) w.r.t. $\boldsymbol{\beta}$ and requiring that this becomes zero we have

$$0 = \frac{\partial C_{\text{LASSO}}}{\partial\boldsymbol{\beta}} = -2\boldsymbol{X}^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}) + \lambda\,\text{sgn}(\boldsymbol{\beta}). \tag{13}$$

This has the added benefit of being able to set certain parameters to be 0 instead of suppressing them, at the cost of losing analytical expressions for $\hat{\boldsymbol{\beta}}$ in non-trivial cases.

### 2.4. Connection to Statistics

The three aforementioned regression techniques all have relatively simple connections to statistical probability distribution functions. Assuming that the likelihood of an event $y_i$ with the input variables $\boldsymbol{X}$ and parameters $\boldsymbol{\beta}$ is given by a Gaussian distribution:

$$p(y_i, \boldsymbol{X}|\boldsymbol{\beta}) = \prod_{i=0}^{n-1} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y_i - \boldsymbol{X}_{i,*}\boldsymbol{\beta})^2}{2\sigma^2}\right]. \quad (14)$$

Further assuming that all events are independent and identically distributed the total probability density function takes the form

$$p(\boldsymbol{y}, \boldsymbol{X}|\boldsymbol{\beta}) = \prod_{i=0}^{n-1} p(y_i, \boldsymbol{X}|\boldsymbol{\beta}), \quad (15)$$

then taking the negative logarithm of the above and defining that as our cost function we can arrive at the exact expression for the cost function that we defined earlier in (6). A similar analysis for Ridge can be done. Using Bayes' theorem we have that $p(\boldsymbol{\beta}|\boldsymbol{X}, \boldsymbol{y}) \propto p(\boldsymbol{y}, \boldsymbol{X}|\boldsymbol{\beta})p(\boldsymbol{\beta})$ we then feed in the prior:

$$p(\beta_i) = \exp\left[-\frac{\beta_i^2}{2\tau^2}\right], \quad (16)$$

one arrives at the probability of the coefficients $\boldsymbol{\beta}$ given the data $\boldsymbol{y}$

$$p(\boldsymbol{\beta}|\boldsymbol{X}, \boldsymbol{y}) \propto p(\boldsymbol{y}, \boldsymbol{X}|\boldsymbol{\beta}) \cdot \prod_{i=0}^{p-1} \exp\left[-\frac{\beta_i^2}{2\tau^2}\right]. \quad (17)$$

Again, defining the cost function as the negative logarithm of this expression one arrives at the Ridge cost function with $\lambda = 1/(2\tau^2)$. Finally for LASSO one assumes that the initial probability is given by the same as OLS but with the addition of the $\beta$ coefficients following a Laplace distribution

$$p(\beta_i) = \exp\left[-\frac{|\beta_i|}{\tau}\right], \quad (18)$$

one arrives at the LASSO cost function with $\lambda = 1/\tau$.

These statistical connections help us better understand the assumptions made with each technique. OLS implicitly assumes that each event $y_i$ are all independent and Gaussian. When this is not the case one should not be surprised if OLS performs poorly. Ridge and LASSO however do account for the chance that there are correlated events, and assumes that the coefficients we compute are themselves Gaussian or Laplace distributed respectively. The hyperparameter $\lambda$ now has a clear interpretation, i.e. that we essentially insert a standard deviation of the $\beta$ coefficients. The performance of these latter two can then also be used to indicate which distribution more accurately describes the coefficients.

### 2.5. Bias-Variance

The so-called Bias-Variance Trade-Off can be summarized in a single equation:

$$\mathbb{E}[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2] = \text{Bias}[\tilde{\boldsymbol{y}}] + \text{Var}[\tilde{\boldsymbol{y}}] + \sigma^2, \quad (19)$$

where the bias is defined as $\mathbb{E}[(f(\boldsymbol{x}) - \mathbb{E}[\tilde{\boldsymbol{y}}])^2]$. The LHS of (19) is the expected value of the MSE which tells us how well the model's predictions match the true data on average. The equation shows that we can decompose this expected MSE into 3 different components.

- Bias: This quantity measures how much the model's average prediction differs from its true value. A high bias implies that the assumptions and simplifications which the model is built on are not valid, i.e. that we are underfitting the data.
- Var: The variance measures how much the model's predictions vary when trained on different datasets. It captures the sensitivity of the model to small changes in the training data. A high variance suggests overfitting, meaning it performs well on the training data but may be capturing noise or false patterns.
- $\sigma^2$: This is the irreducible error or noise in the data itself which cannot be explained by the model.

The idea is to minimize the LHS of (19), so clearly we want to minimize both the bias and variance at the same time. However these are correlated to one another, so lowering e.g. the bias can in general increase the variance and vice-versa. So bias-variance trade-off is essentially trying to optimize the complexity of the model such that we neither overfit nor underfit the model such that it can be generalized to other cases whilst not being too simplistic. These quantities can then be used as means to fine tune a model.

### 2.6. Resampling

Resampling techniques are critical in understanding the performance and robustness of a model. They tend to better show how well a model will do on data it has not seen before, and are thus useful for diagnosing problems like over- and underfitting. In this report focus on two resampling methods: bootstrapping and cross-validation.

#### 2.6.1. Bootstrap

Bootstrapping is a statistical method that involves taking random samples with replacement from the original dataset to create several new datasets. These datasets are the same size as the original dataset, but since our sampling was done with replacement, some observations

are presented more than once whilst others are eliminated. Here we train and evaluate a model over each of these resampled datasets allowing us to estimate how the performance varies across different datasets. This method is particularly important when the datasets are small as it heavily increases the amount information that can be extracted from a given data set. Averaging across all the performance metrics from the bootstrap samples can provide a clearer description of how well our model performs.

#### 2.6.2. Cross-Validation

$k$-fold cross-validation is the other resampling method which we considered in this project. In $k$-fold cross-validation, the dataset is divided into $k$ subsets (or folds). The model is trained on $k-1$ of these folds and tested on the remaining fold. This process is repeated $k$ times, with each fold being used as the test set exactly once. The performance metrics, MSE and $R^2$, are averaged over the $k$ iterations to provide a more reliable estimate of how the model performs on unseen data.

Cross-validation is particularly advantageous for its ability to reduce the variance of performance estimates by using different portions of the data for both training and validation. In our analysis, we used 5-fold cross-validation to evaluate the stability and predictive power of each regression method. This method gave us a clearer picture of how each model generalizes, helping us tune the hyperparameters, such as the regularization parameter $\lambda$ for Ridge and LASSO.

Both resampling techniques are used to gain insight into the models performance and helps us ensure that the conclusions drawn from our analysis are not overly dependent on a particular division of the dataset.

### 2.7. Scaling

Data scaling is an important tool in machine learning and data analysis. It may be used to simply make data more understandable as to keep them on a more human friendly scale, or if the data varies over several order of magnitude can be used to avoid numerical error and instability. Two most common ways to scale data are `MIN-MAX` and `StandardScaling`. MIN-MAX does as its name suggests; the range of the feature $x_i$ gets scaled to the interval $[a, b]$ one applies the following transformation

$$x_i \to (b-a)\frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} - a. \tag{20}$$

StandardScaling however subtracts the mean value of the smple from each data point and divides this by the standard deviation of the sample:

$$x_i \to \frac{x_i - \bar{x}_i}{\sigma(x_i)}. \tag{21}$$

## 3. IMPLEMENTATION

In this project we look to implement the three polynomial regression models: OLS, Ridge and LASSO, on 2D surface functions $f(x, y) = z$. Before testing our models on actual data, we work with the Franke function which can be visualized in Fig. 1 shows a visualization and is given by:

$$\begin{aligned} f(x,y) = &\frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) \\ &+ \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)^2}{10}\right) \\ &+ \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) \\ &- \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right). \end{aligned} \tag{22}$$

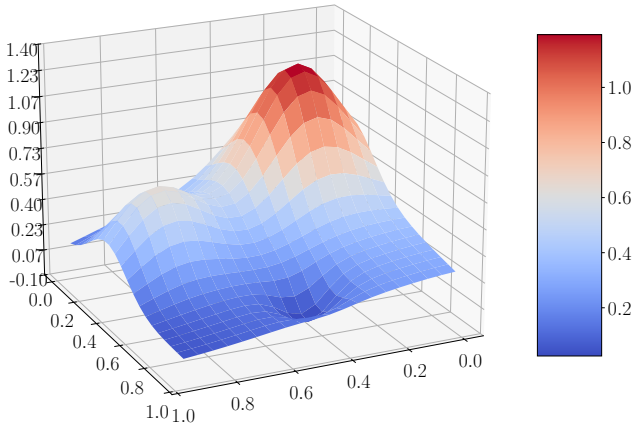


Fig. 1. Visualization of the Franke function (22)

Here we input $x, y \in [0, 1]$ implying that we have already essentially scaled the data with MIN-MAX scaling. Since considering the function for negative values of $x$ and $y$ essentially gives us a different surface we have decided that there is no reason to perform StandardScaling for this case. As such we proceed without any further scaling of the data for the Franke function. To perform an analysis on this function we consider a polynomial fit up

to degree $n$ analogous to (2) where:

$$\begin{aligned}\tilde{\boldsymbol{z}} = \frac{1}{n+1}\sum_{i=0}^{n}\Big(&\boldsymbol{\beta}_{0,0}\\ &+\boldsymbol{\beta}_{1,0}x_i+\boldsymbol{\beta}_{1,1}y_i\\ &+\boldsymbol{\beta}_{2,0}x_i^2+\boldsymbol{\beta}_{2,1}x_iy_i+\boldsymbol{\beta}_{2,2}y_i^2\\ &+...\\ &+\boldsymbol{\beta}_{n,0}x_i^n+\boldsymbol{\beta}_{n,1}x_i^{n-1}y+...+\boldsymbol{\beta}_{n,n-1}x_iy_i^{n-1}+\boldsymbol{\beta}_{n,n}y_i^n\Big)\\ =\frac{1}{n+1}\sum_{i,j=0}^{n}\sum_{k=0}^{j}&x_i^{j-k}y_i^k\boldsymbol{\beta}_{j,k}\equiv\boldsymbol{X}\boldsymbol{\beta}. \end{aligned} \tag{23}$$

Here the components $x_i$ and $y_i$ are entries in the input vectors $\boldsymbol{x}^T=\begin{bmatrix}x_0 & ... & x_n\end{bmatrix}$ and $\boldsymbol{y}^T=\begin{bmatrix}y_0 & ... & y_n\end{bmatrix}$ respectively which are our independent variables. Each $\boldsymbol{\beta}_{i,j}$ is a $\frac{(n+1)(n+2)}{2}$ component vector with a single non-zero entry with magnitude $|\beta_{i,j}|$ and the design matrix $\boldsymbol{X}$ is then an $(n+1)\times\frac{(n+1)(n+2)}{2}$ matrix of the form:

$$\boldsymbol{X}=\frac{1}{n+1}\begin{bmatrix}1 & x_0 & y_0 & x_0^2 & x_0y_0 & ... & x_0^n & ... & y_0^n\\ 1 & x_1 & y_1 & x_1^2 & x_1y_1 & ... & x_1^n & ... & y_1^n\\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots\\ 1 & x_n & y_n & x_n^2 & x_ny_n & ... & x_n^n & ... & y_n^n\end{bmatrix}, \tag{24}$$

and the $\boldsymbol{\beta}$ vector contains the $\frac{(n+1)(n+2)}{2}$ components

$$\boldsymbol{\beta}^T=\sum_{i=0}^{n}\sum_{j=0}^{i}\boldsymbol{\beta}_{i,j}^T=\begin{bmatrix}\beta_{0,0} & \beta_{1,0} & \beta_{1,1} & ... & \beta_{n(n-1)} & \beta_{n,n}\end{bmatrix}.$$

It should now be clear which unit vectors correspond to each term in (23). Note that for all practical purposes the factor $(1+n)^{-1}$ in (24) can be omitted as this simply scales the $\beta_{i,j}$ coefficients with the same factor.

The inputs $\boldsymbol{x}$ and $\boldsymbol{y}$ are then split randomly into a training and test set with the latter comprising 1/4 of the data throughout our analysis. These are then used to calculate both the train and test variations of $\boldsymbol{z}$ and $\boldsymbol{X}$ using the function `train_test_split` from `sklearn.model_selection`. The training and testing set may then be scaled (or not), before the $\beta$-coefficient is calculated.

For Ridge, $\beta_{\text{Ridge}}$ is found by numpy's `np.linalg.pinv` to calculate the pseudoinverse of (10). This function performs an SVD on the matrix $\boldsymbol{X}$ as was done in (11) to obtain an inverse even if the matrix is singular. Obtaining $\beta_{\text{OLS}}$ is then the special case of $\beta_{\text{Ridge}}$ with $\lambda=0$. The $\beta$-coefficient for LASSO on the other hand, is computed using `sklearn.Lasso`. Sklearn's LASSO is dependent on the chosen maximum number of iterations and a tolerance parameter. Throughout this project these parameters are set to $10^5$ and $10^{-1}$, respectively. Lastly, the mean squared error and the R2 score is then calculated using Sklearn's `metrics` library.

Bootstrap was then implemented by running a for-loop over the polynomial degrees and another loop over the samples. Throughout we decided to use $x$ samples and only considered OLS for this part. The error and bias were calculated from their definitions whilst `np.var` was used to calculate the variance. Note that for the bias the approximation

$$\mathbb{E}[(f(\boldsymbol{x})-\mathbb{E}[\tilde{\boldsymbol{y}}])^2]\approx\mathbb{E}[(\boldsymbol{y}-\mathbb{E}[\tilde{\boldsymbol{y}}])^2] \tag{25}$$

was used. This is only valid in the low $\boldsymbol{\varepsilon}$ limit and thus $\boldsymbol{\varepsilon}=\mathcal{N}(0,0.01)$ was used for this part.

Next $k$-fold Cross-Validation (CV) was implemented by taking in the design matrix $\boldsymbol{X}$, the target vector $\boldsymbol{z}$, the number of folds $k$ and the hyperparameter $\lambda$. The data set is then shuffled in hopes that each fold does not contain any ordering bias. The shuffled data set is split into $k$ folds where one of the folds is the test set and the remaining $k-1$ folds are used as training sets. This is iterated over $k$ times as explained in the theory section and finally the average and standard deviation over the $k$ runs are calculated.

Once everything has been implemented, we use the programs created to run on both the Franke function and the aforementioned terrain data from [1]. For the latter we decided on two topologically interesting places: Mount Everest and the Grand Canyon. The image and contour plot for these are given in Fig. 2. Note however that these figures have a resolution of $2^{15}\times2^{15}$, but for computational purposes we will limit ourselves to a small subsection of this. We implemented both MIN-MAX scaling and StandardScaling for the terrain data but opted into using StandardScaling for all the results.

## 4. RESULTS & DISCUSSION

### 4.1. The Franke Function

We first consider the results obtained from the data samples generated from the Franke test function (22) using OLS, Ridge, and LASSO with an added normal-distributed noise proportional to $\varepsilon=0.1$.

#### 4.1.1. OLS

The MSE and $R^2$-score for both the train and test data as a function of the polynomial degree are given in Figs. 3 and 4. The MSE of the test data exhibits a square-well shape with its lowest point at around $p=6$, while the MSE of the training data steadily decreases as $p$ increases. Similar traits are seen for the $R^2$ plot, however mirrored vertically. This is as expected for low $p$ as the Franke function is too complicated to be accurately described by low order polynomials. Once the polynomial degree becomes too high we start to clear signs of overfitting as we see sharp increases (decreases) in the MSE ($R^2$).

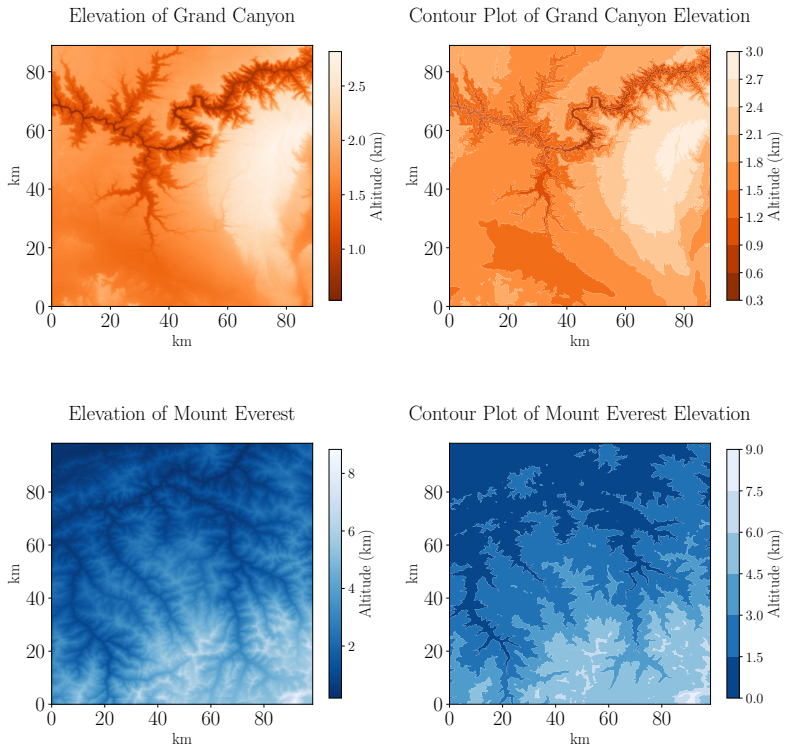


Fig. 2. Image and contour-plot of the Grand Canyon and Mount Everest in orange and blue respectively obtained from [1].

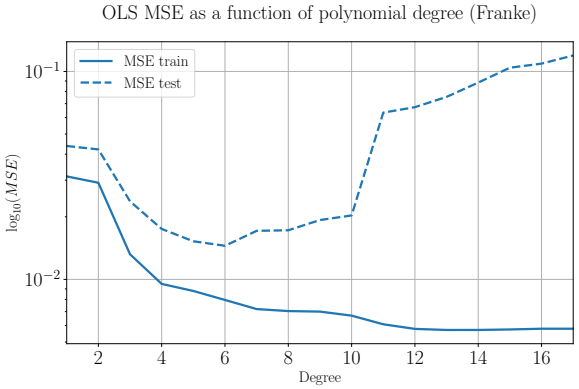


Fig. 3. The (log) MSE of the train (solid) and test (dashed) data from the OLS regression method using data generated by $N = 100$ datapoints from the Franke function in (22), with noise proportional to $\varepsilon = 0.1$. The $y$-axis is logarithmic, and the $x$-axis gives the polynomial degree $p \in [0, 17]$.

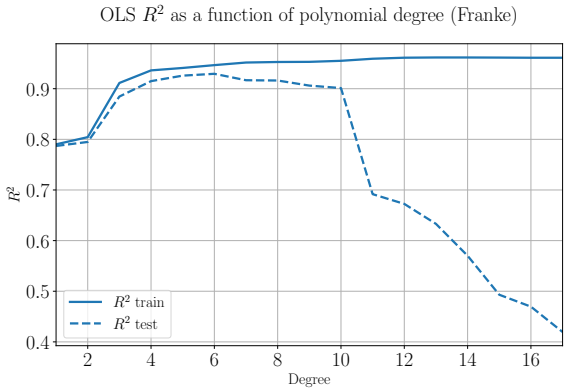


Fig. 4. The $R^2$-score of train (solid) and test (dashed) data from the OLS regression method using data generated by $N = 100$ datapoints from the Franke function in (22), with noise proportional to $\varepsilon = 0.1$. The $x$-axis gives the polynomial degree $p \in [0, 17]$.

The $\beta$-coefficients for $N = 100$ and $N = 1000$ datapoints can be seen in Figs. 5 and 6, respectively. The color scaling indicates what polynomial degree the $\beta$-coefficients correspond to. For $N = 100$ the $\beta$-coefficients of higher degree polynomial seem larger in overal size compared to lower degree polynomial ($p \in [0, 8]$). However, for $N = 1000$ this behavior is opposite, meaning that the $\beta$-coefficients for $p \in [0, 8]$ seem larger than those corresponding to higher degree polynomials.

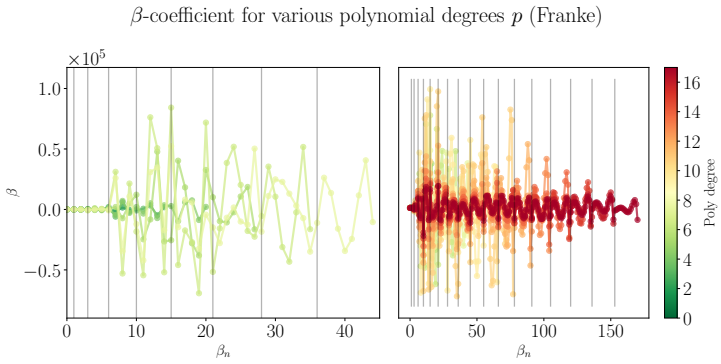


Fig. 5. The $\beta$-coefficients for the OLS-fit of $N = 100$ datapoints generated from the Franke function (22) with noise proportional to $\varepsilon = 0.1$. The plots shows the polynomial degree from 0 (green) to 17 (red). The left hand side is restricted to $\beta$-coefficients corresponding to up to $p = 8$. The solid lines in the background indicate where the last coefficient of a given polynomial degree is placed.

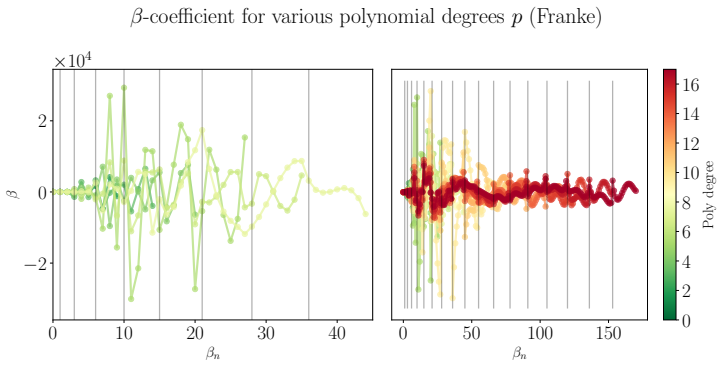


Fig. 6. The $\beta$-coefficients for the OLS-fit of $N = 1000$ datapoints generated from the Franke function (22) with noise proportional to $\varepsilon = 0.1$. The plots shows the polynomial degree from 0 (green) to 17 (red). The left hand side is restricted to $\beta$-coefficients corresponding to up to $p = 8$. The solid lines in the background indicate where the last coefficient of a given polynomial degree is placed.

#### 4.1.2. Ridge

This section presents the MSE and $R^2$-score for the Ridge regression method using data generated from the Franke function (22) given in Figs. 7 and 8 respectively. These metrics are presented as functions of the polynomial degree $p$ as before but now with various fixed $\lambda$'s.

Here we can clearly see that there is some discrepancy for different values of $\lambda$. The case where $\lambda = 10^{-10}$ can be seen to have its MSE ($R^2$) increase (decrease) quite early on as the polynomial degree increases whilst the cases for larger $\lambda$ remain relatively stable throughout. From the three stable values of $\lambda$ we see that $\lambda = 10^{-7}$ and $\lambda = 10^{-4}$ perform much better than the others. As such one would expect the optimal $\lambda$ value to lie in between these values. All the MSE and $R^2$ scores can be seen to flatten out as the polynomial degree increases. This is

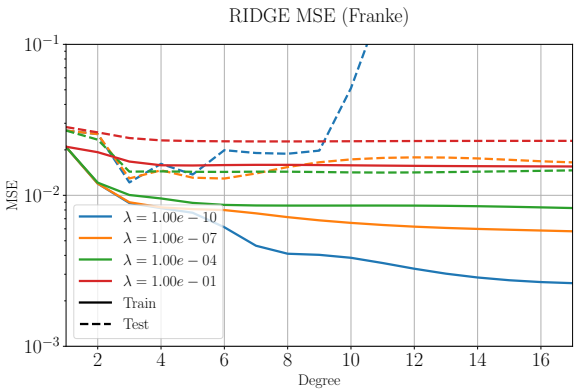


Fig. 7. Ridge MSE as a function of the max polynomial degree with $N = 100$, $p_{\max} = 17$ and $\varepsilon = 0.1$. The four chosen values for the hyperparameter $\lambda$ are $10^{-15}, 10^{-7}, 10^{-4}$ and $10^{-1}$ which are denoted by the different colors. The solid and dashed lines correspond to the training and test set respectively.

due to the increased stability Ridge has due to its penalty term. This essentially works to drive down higher order polynomials reducing overfitting. As such Ridge is less sensitive to getting the correct model complexity in place unlike OLS.

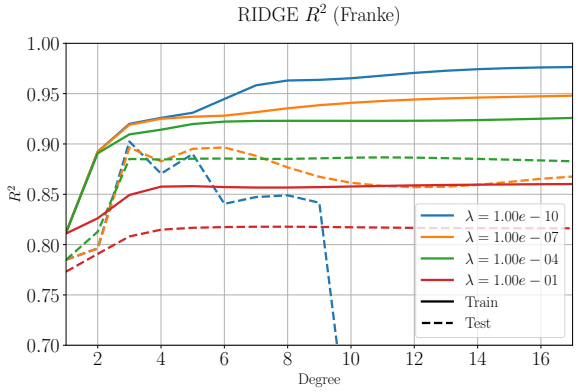


Fig. 8. Ridge $R^2$-score as a function of the max polynomial degree with $N = 100$, $p_{\max} = 10$ and $\varepsilon = 0.1$. The line and color scheme is the same as for Fig. 7.

To view the dependence on $\lambda$ more explicitly we fixed the degree to $p = 6$ and used 100 logarithmically spaced values of $\lambda$ and plotted the MSE and $R^2$ as a function of $\lambda$ in Figs. 9 and 10 respectively. The chosen value of $p = 6$ was considered due to the diminishing returns from increasing $p$ past this value, as seen in fig. 7 and 8. The minimum MSE and maximum $R^2$-score for the test data in fig. 9 and 10, respectively, suggest an optimal value of $\lambda \sim 10^{-6}$.

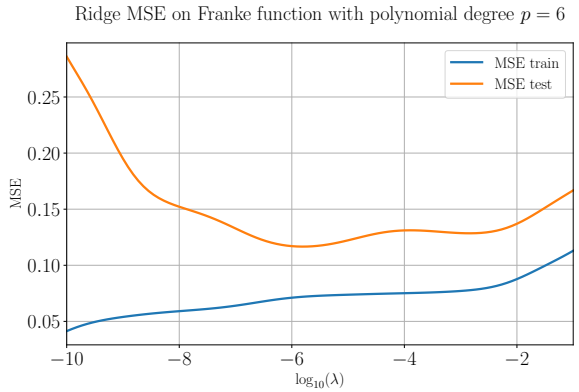


Fig. 9. Ridge MSE with max polynomial degree 4 as a function of the hyperparameter $\lambda$ with $N = 100$ and $\varepsilon = 0.1$.

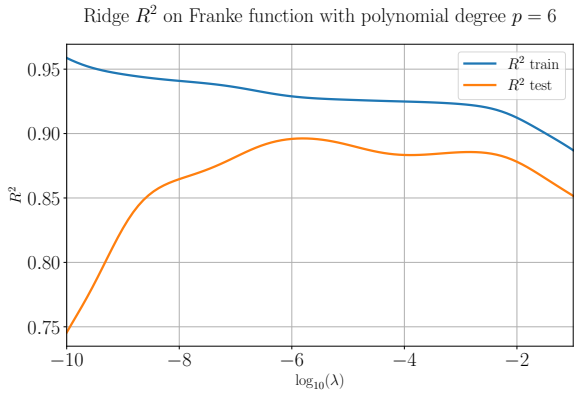


Fig. 10. Ridge $R^2$-score with max polynomial degree 4 as a function of the hyperparameter $\lambda$ with $N = 100$ and $\varepsilon = 0.1$.

#### 4.1.3. LASSO

At last we look at the results for LASSO. Again the MSE and $R^2$-scores as a function of the polynomial degree are given in Figs. 11 and 12 respectively. Unlike in the cases of OLS and Ridge, these plots are strikingly constant, showing little to no variation. There is a negligible effect on the MSE and $R^2$-score from increasing the polynomial degree, and seemingly the only effect of increasing $\lambda$ is to make the MSE ($R^2$) higher (lower).

This can be seen from Figs. 13 and 14, as the two metrics remain close to constant between $\lambda = 10^{-10}$ and $\lambda = 10^{-3}$. However, for $\lambda \geq 10^{-3}$, the MSE ($R^2$) climbs (falls) almost vertically. The domain $\lambda \gtrsim 10^{-3}$ is likely when $\lambda$ becomes too large to set crucial $\beta$ coefficients to 0 and thus causing the method to be underfitting the data.

For the current case the only advantage that LASSO has over OLS is just insensitivity to increase in degree, and thus it is less important to keep the complexity of the model at an optimal level. However Ridge also accomplishes this whilst not massively increasing and decreasing the MSE and $R^2$-score respectively.

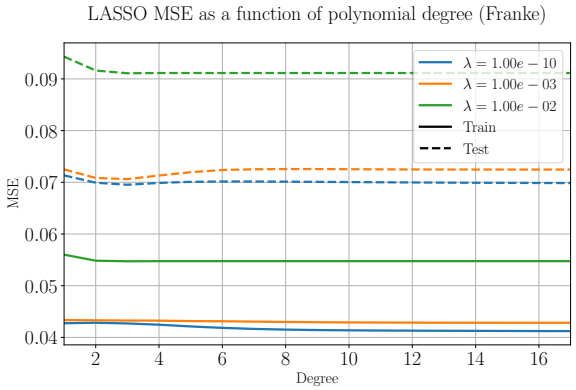


Fig. 11. LASSO MSE as a function of the max polynomial degree with $N = 100$, $p_{\max} = 17$ and $\varepsilon = 0.1$. The three chosen values for the hyperparameter $\lambda$ are $10^{-10}, 10^{-3}$ and $10^{-2}$. The remaining description can be found under Fig. 7

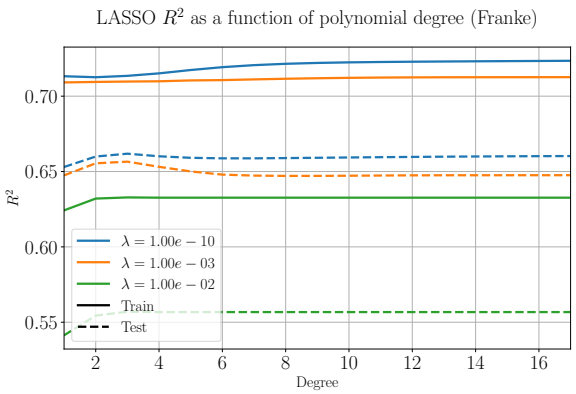


Fig. 12. LASSO $R^2$ as a function of the max polynomial degree with $N = 100$, $p_{\max} = 17$ and $\varepsilon = 0.1$. The three chosen values for the hyperparameter $\lambda$ are $10^{-10}, 10^{-3}$ and $10^{-2}$.

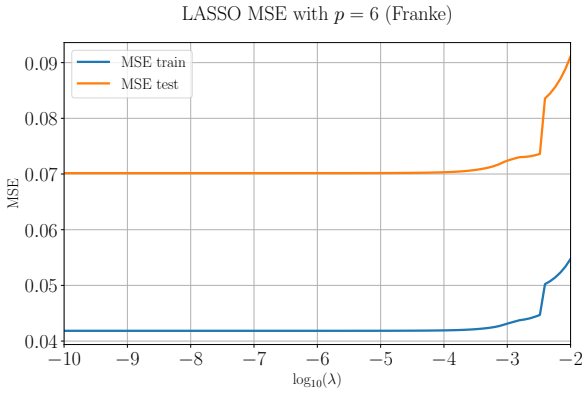


Fig. 13. LASSO MSE with max polynomial degree 4 as a function of the hyperparameter $\lambda$ with $N = 100$ and $\varepsilon = 0.1$.

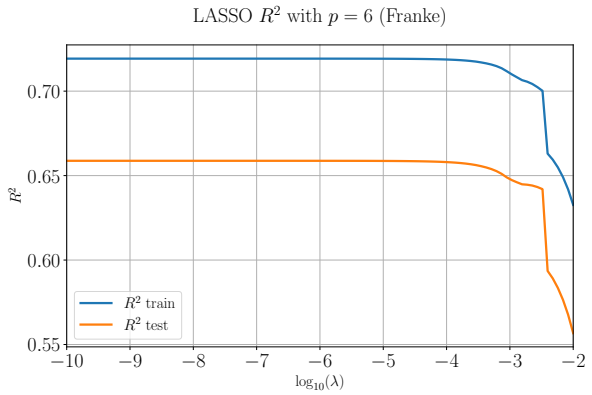


Fig. 14. LASSO $R^2$-score with max polynomial degree 4 as a function of the hyperparameter $\lambda$ with $N = 100$ and $\varepsilon = 0.1$.

### 4.2. Bootstrap and Cross Validation

Using the Bootstrap method with 100 samples, the bias, variance and error is shown in fig. 15 for an OLS regression of $N = 100$ data samples from the Franke function (22). The figure shows a close-to constant variance for polynomial degrees smaller than $p \sim 7$, whereas for $p \gtrsim 7$ the variance increases steadily. On the other hand, the bias starts out above $\sim 0.1$, gradually decreasing, reaching a minimum for $p$ in the range $4 - 9$, before rising again for $p \gtrsim 9$. These specific values of $p$ is dependent on the sample size $N = 100$ and the amount of samples taken by the Bootstrap method. However, the general shape of the error curve in fig. 15 is expected independent of these settings. The error is supposed to be proportional to the sum of the variance and the bias, as seen in (19). Visually, fig. 15 confirms this, although the sum is not explicitly given.

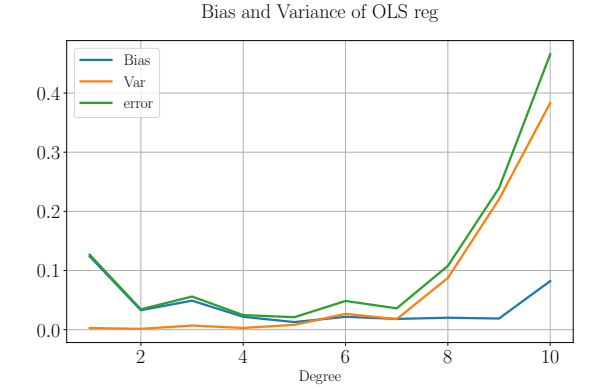


Fig. 15. The bias, variance and error of a OLS regression on data generated from $N = 25$ samples of the Franke function (22) with noise proportional to $\varepsilon = 0.01$. The metrics are generated using the bootstrap method.

Cross validation is performed for the three regression methods OLS, Ridge and LASSO on data generated from the Franke function (22). The results are obtained with $\lambda$-parameter for Ridge and LASSO taking the values $\lambda \in \{10^{-10}, 10^{-7}, 10^{-4}, 10^{-1}\}$ ← we only show $\lambda = 10^{-6}$, should we show more?.

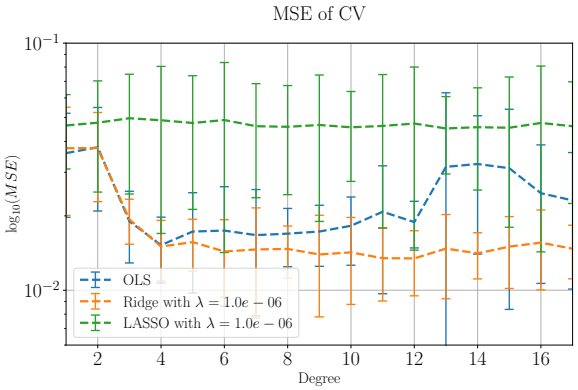


Fig. 16. Cross validation for the three models OLS, Ridge and LASSO as a function of the polynomial degree on data generated from $N = 100$ points and samples of the Franke function (22) with noise proportional to $\varepsilon = 0.1$. The $y$-axis shows the (logarithmic) MSE with error-bars for the three methods, with $\lambda = 10^{-6}$.

### 4.3. Terrain Data

For the terrain data we opted

## 5. CONCLUSION

OLS and Ridge clearly outperform LASSO on both the Franke function and terrain data. The current hypothesis for this, following the discussion in section 2.4, is that LASSO performs worse due to the $\beta$ coefficients not following a Laplace distribution. If these coefficients do instead follow a Gaussian distribution and there is a low correlation between the different events $z_i$ then this is exactly what one would expect. The expected bias-variance trade-off was observed explicitly by using bootstrapping. For higher (lower) polynomial degrees we clearly see an increase in the variance (bias) and helps us choose which model complexity we should aim for.

On the terrain data we still see that OLS and Ridge outperform LASSO, however all regression methods perform relatively poorly all things considered. As a result we conclude that the specific parametrization we have performed using (23) may not be suitable for these particular areas due to...

content...

[1] U.S. Geological Survey, "EarthExplorer: Terrain Elevation Data." `https://earthexplorer.usgs.gov`, 2024. Accessed: 20/09/2024. Extracted terrain elevation data.
[2] C. M. Bishop, *Pattern Recognition and Machine Learning*, vol. 1. Springer, 2006.

## Appendix A: Derivations

### 1. Model

The variance of $\boldsymbol{y}$ calculated as follows:

$$\begin{aligned}\mathrm{Var}(y_i) &= \mathbb{E}\{[y_i - \mathbb{E}(y_i)]^2\} \\ &= \mathbb{E}\{(\boldsymbol{X}_{i,*}\boldsymbol{\beta} + \varepsilon_i)^2\} - (\boldsymbol{X}_{i,*}\boldsymbol{\beta})^2 \\ &= (\boldsymbol{X}_{i,*}\boldsymbol{\beta})^2 + \mathbb{E}(\varepsilon_i^2) + 2\mathbb{E}(\varepsilon_i)\boldsymbol{X}_{i,*}\boldsymbol{\beta} - (\boldsymbol{X}_{i,*}\boldsymbol{\beta})^2 \\ &= \mathrm{Var}(\varepsilon_i^2) = \sigma^2\end{aligned}$$

A direct way of seeing the effect of the regulator is by considering the singular value decomposition of $\boldsymbol{X}$. Writing $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}$ where $\boldsymbol{U}, \boldsymbol{V}$ are orthogonal and $\boldsymbol{\Sigma}$ only contains elements on the diagonal the proof goes as follows:

$$\begin{aligned}\tilde{\boldsymbol{y}}_{\mathrm{Ridge}} &= \boldsymbol{X}\boldsymbol{\beta}_{\mathrm{Ridge}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y} \\ &= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T((\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T)^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T + \lambda\boldsymbol{I})^{-1}(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T)^T\boldsymbol{y} \\ &= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T(\boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}\boldsymbol{V}^T + \lambda\boldsymbol{I})^{-1}\boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{U}^T\boldsymbol{y} \\ &= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T(\boldsymbol{V}(\boldsymbol{\Sigma}^T\boldsymbol{\Sigma} + \lambda\boldsymbol{I})\boldsymbol{V}^T)^{-1}\boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{U}^T\boldsymbol{y} \\ &= \boldsymbol{U}\boldsymbol{\Sigma}(\boldsymbol{\Sigma}^T\boldsymbol{\Sigma} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Sigma}^T\boldsymbol{U}^T\boldsymbol{y} \\ &= \sum_{j=0}^{p-1}\boldsymbol{u}_j\boldsymbol{u}_j^T\frac{\sigma_j^2}{\sigma_j^2 + \lambda}\boldsymbol{y}\end{aligned}$$

where the last step is valid due to the orthogonality of $\boldsymbol{U}$ and $\sigma_j$ are the elements on the diagonal of $\boldsymbol{\Sigma}$.

### 2. Coefficients

The expectation value of the optimal parameters $\hat{\boldsymbol{\beta}}$ can be found to be

$$\begin{aligned}\mathbb{E}(\hat{\boldsymbol{\beta}}_{\mathrm{OLS}}) &= \mathbb{E}[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}] \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}[\boldsymbol{y}] \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{\beta}.\end{aligned}$$

with the variance

$$\begin{aligned}\mathrm{Var}(\hat{\boldsymbol{\beta}}_{\mathrm{OLS}}) &= \mathbb{E}\{[\boldsymbol{\beta} - \mathbb{E}(\boldsymbol{\beta})][\boldsymbol{\beta} - \mathbb{E}(\boldsymbol{\beta})]^T\} \\ &= \mathbb{E}\{[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{\beta}] \\ &\quad \times [(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{\beta}]^T\} \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}\{\boldsymbol{y}\boldsymbol{y}^T\}\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} \\ &\quad - \boldsymbol{\beta}\boldsymbol{\beta}^T \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T[\boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T + \sigma^2]\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} \\ &\quad - \boldsymbol{\beta}\boldsymbol{\beta}^T \\ &= \boldsymbol{\beta}\boldsymbol{\beta}^T + \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \\ &= \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}\end{aligned}$$

The optimal parameters for Ridge regressions can again be found by the same process as for OLS:

$$\begin{aligned}0 = \frac{\partial C_{\mathrm{Ridge}}}{\partial\boldsymbol{\beta}} &= -\frac{2}{n}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T\boldsymbol{X} + 2\lambda\boldsymbol{\beta}^T \\ &= \frac{2}{n}(\boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{X} - \boldsymbol{y}^T\boldsymbol{X}) + 2\lambda\boldsymbol{\beta}^T \\ 0 &= \boldsymbol{\beta}^T(\boldsymbol{X}^T\boldsymbol{X} + \tilde{\lambda}\boldsymbol{I}) - \boldsymbol{y}^T\boldsymbol{X} \\ \boldsymbol{\beta}^T &= \boldsymbol{y}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X} + \tilde{\lambda}\boldsymbol{I})^{-1} \\ \boldsymbol{\beta} &= (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}\end{aligned}$$

where we defined $\tilde{\lambda} \equiv n\lambda$, renamed $\tilde{\lambda} \to \lambda$ and used that the matrix in the parenthesis is a symmetric matrix and thus its inverse must also be symmetric.

### 3. Bias-Variance

For ease of notation we write $f(\boldsymbol{x}) = f$ and simply ignore vector notation since everything is a scalar in the end. Then we have

$$\begin{aligned}\mathbb{E}[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2] &= \mathbb{E}[(f + \boldsymbol{\varepsilon} - \tilde{\boldsymbol{y}})^2] \\ &= \mathbb{E}[(f - \tilde{\boldsymbol{y}})^2] + 2\underbrace{\mathbb{E}[(f - \tilde{\boldsymbol{y}})\boldsymbol{\varepsilon}]}_{=0} + \underbrace{\mathbb{E}[\boldsymbol{\varepsilon}^2]}_{=\sigma^2} \\ &= \mathbb{E}[((f - \mathbb{E}[\tilde{\boldsymbol{y}}]) - (\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{\boldsymbol{y}}]))^2] + \sigma^2 \\ &= \mathbb{E}[(f - \mathbb{E}[\tilde{\boldsymbol{y}}]))^2] + \mathbb{E}[(\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{\boldsymbol{y}}])^2] \\ &\quad - 2\,\mathbb{E}[((f - \mathbb{E}[\tilde{\boldsymbol{y}}])(\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{\boldsymbol{y}}]))] + \sigma^2 \\ &= \mathrm{Bias}[\tilde{\boldsymbol{y}}] + \mathrm{Var}[\tilde{\boldsymbol{y}}] + \sigma^2 \\ &\quad - 2\,\mathbb{E}[(f - \mathbb{E}[\tilde{\boldsymbol{y}}])(\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{\boldsymbol{y}}])]\end{aligned}$$

where $\mathbb{E}[(f - \tilde{\boldsymbol{y}})\boldsymbol{\varepsilon}] = 0$ is justified by $\boldsymbol{\varepsilon}$ being independent and we note that the wrong definition of the Bias is given in the problem text (with that definition $\sigma^2$ gets put into the 'Bias'). All that remains is to show that the last term

is 0. Since $\mathbb{E}[f] = f$ and $\mathbb{E}[f\,\mathbb{E}[\tilde{\boldsymbol{y}}]] = f\,\mathbb{E}[\mathbb{E}[\tilde{\boldsymbol{y}}]] = f\,\mathbb{E}[\tilde{\boldsymbol{y}}]$ which proves the claim.
then

$$\begin{aligned}\mathbb{E}[(f - \mathbb{E}[\tilde{y}])(\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{y}])] &= \mathbb{E}[f\tilde{\boldsymbol{y}} - f\,\mathbb{E}[\tilde{\boldsymbol{y}}] - \tilde{\boldsymbol{y}}\,\mathbb{E}[\tilde{y}] + \mathbb{E}^2[\tilde{\boldsymbol{y}}]] \\ &= f\,\mathbb{E}[\tilde{\boldsymbol{y}}] - f\,\mathbb{E}[\tilde{\boldsymbol{y}}] - \mathbb{E}^2[\tilde{\boldsymbol{y}}] + \mathbb{E}^2[\tilde{\boldsymbol{y}}] \\ &= 0\end{aligned}$$