FYS-STK3155: Regression Analysis and Resampling Techniques in Applied Machine Learning

Oscar Atle Brovold, Eskil Grinaker Hansen, Håkon Ganes Kornstad

October 7, 2024

`Project GitHub Repository`

Abstract

In this paper, we analyze three different methods for polynomial regression in machine learning: Ordinary Least Squares, Ridge, and Lasso Regression. We evaluate their performance with mean squared error and R^2-score, and we explore data resampling techniques such as bootstrapping and k-fold cross validation. First we fit the models to a two dimensional test function, and then to real mountain data from Norway. We find the Ordinary Least Squares to outperform both Ridge and Lasso in approximating real landscapes.

INTRODUCTION

With an ever growing wealth of data, and the computational power to match it, machine learning is a rapidly growing field. It is now being used across a wide array of domains, including finance, physics and healthcare [1]. Its methods can provide us with insight into dimensions beyond human comprehension.

One of the most fundamental principles of machine learning is the use of regression methods [2], employed for either prediction or classification. In this paper, we will investigate three basic methods of *linear* regression: Ordinary Least Squares (OLS), Ridge, and Lasso. First, we apply these regression methods to data generated by the continuous Franke's Function[3], both with and without noise. To improve the models' performance, we apply oversampling techniques such as bootstrapping and k-fold cross validation. Hyperparameters, such as the degree of complexity and the regularization term λ, are then fine-tuned to improve the models' fit to the data. We evaluate and compare the accuracy of each model using statistics such as Mean Squared Error (MSE) and R^2-scores.

Finally, we extend this analysis to real terrain data from Evje outside of Stavanger, Norway, applying the same methods.

THEORETICAL BACKGROUND

Note: If no other sources are referenced, the theoretical part is based on the course lecture notes. [4]

Regression Analysis

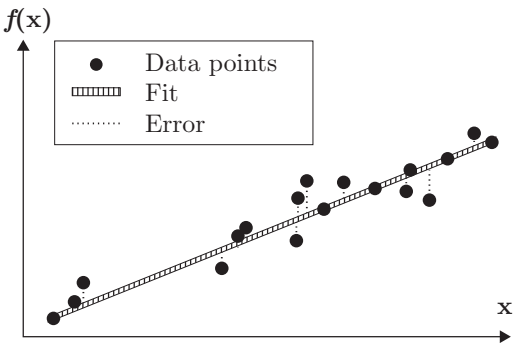


Figure 1: All regression problems in $\mathbb{R}$ involve fitting a line to a set of points, and measuring the errors. This problem can be generalized to higher dimensions, fitting a hyperplane in $\mathbb{R}^n$

Given a dataset, we assume that the observations $\boldsymbol{z}$ are generated by the following equation

$$\boldsymbol{z} = f(\mathbf{x}) + \boldsymbol{\varepsilon} \tag{1}$$

where $\boldsymbol{z}$ represent the observed responses, $f(\boldsymbol{x})$ is an unknown, non-stocahastic function, and $\boldsymbol{\varepsilon}$ is stochastic noise. The objective of regression is to approximate $f(\boldsymbol{x})$

using the observed data. (For more details, appendix D.7)

To achive this we optimize a set of coefficients $\hat{\boldsymbol{\beta}}$ and predict with the following equation

$$\tilde{\boldsymbol{z}} = \mathbf{X}\hat{\boldsymbol{\beta}}$$

Here $\tilde{\boldsymbol{z}}$ is the predicted responses and $\mathbf{X}$ is the *design matrix*. In $\mathbf{X}$, each column represents a feature, and the rows represent samples.

By applying a polynomial transformation to the input data, we can capture the polynomial relationship between the input and output while maintaining a linear framework. This approach is commonly known as polynomial regression. To capture these features, the Vandermonde matrix [5] is commonly used:

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{p-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{p-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{p-1} \end{pmatrix}$$

When given two input sets $\boldsymbol{x}$ and $\boldsymbol{y}$, the permutations for each column is given as all combinations $x_i^a y_j^b$ where $a + b \leq p - 1$. The total number of such combinations, i.e terms in the polynomial, is given by the expression

$$\mathrm{N} = \frac{(p+1)(p+2)}{2}$$

Ordinary Least Squares

Our goal is to calculate the linear model that best fits or approximates the data in equation 1. In the case of OLS the cost function is given by:

$$C_{\mathrm{OLS}}(\boldsymbol{\beta}) = \|\boldsymbol{z} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \tag{2}$$

We aim to find the coefficients $\hat{\boldsymbol{\beta}}$ that minimizes 2. This can be done by differentiating with respect to $\boldsymbol{\beta}$ and set it equal to zero:

$$\frac{\partial}{\partial \boldsymbol{\beta}} \|\boldsymbol{z} - \mathbf{X}\boldsymbol{\beta}\|_2^2 = 0$$

This can be shown to lead to the **normal equation** (See appendix D.1 for full derivation):

$$\hat{\boldsymbol{\beta}}_{OLS} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \boldsymbol{z}$$

The solution assumes that $\mathbf{X}^\top\mathbf{X}$ is invertible, which holds when $\mathbf{X}$ has full column rank [6]. If $\mathbf{X}^\top\mathbf{X}$ is not invertible (singular), alternative methods such as regularization or using a pseudoinverse can be applied.[7]

Ridge Regression

Ridge regression expands on OLS to prevent overfitting and manage multicollinearity. This is done by adding a regularization term $\lambda\|\boldsymbol{\beta}\|_2^2$ to the OLS cost function [8], leading to

$$C_{\mathrm{Ridge}}(\mathbf{X}, \boldsymbol{\beta}) = \|\mathbf{X}\boldsymbol{\beta} - \boldsymbol{z}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 \tag{3}$$

The regularization term introduces a constraint that penalizes larger coefficient values. The degree of this regularization is then controlled by the parameter λ. As $\lambda \to 0$, Ridge regression behaves like OLS. As λ increases, the regularization effect grows, thus shrinking the coefficients more.

When minimizing 3 with respect to $\boldsymbol{\beta}$, we get the optimal coefficients (See appendix D.2 for full derivation):

$$\hat{\boldsymbol{\beta}}_{\mathrm{Ridge}} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\boldsymbol{z}$$

Ridge regression mitigates the issue of $\mathbf{X}^\top\mathbf{X}$ being singular or near singular (ill-conditioned), by replacing it with $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}$, which is always invertible, leading to stabler coefficient estimates. [7]

Lasso Regression

Similar to Ridge regression, Lasso regression introduces a regularization term, but instead of the L2-norm, it uses the L1-norm, defined as $\lambda\|\boldsymbol{\beta}\|_1$ [9]. This results in the following cost function:

$$C_{\mathrm{Lasso}}(\mathbf{X}, \boldsymbol{\beta}) = \|\mathbf{X}\boldsymbol{\beta} - \boldsymbol{z}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 \tag{4}$$

The key difference in Ridge and Lasso regression lies in the nature of their regularization. While Ridge regression shrinks the coefficients towards zero, Lasso regression can set specific coefficients *exactly* to zero. This property makes Lasso useful in feature selection, effectively eliminating less important features. The resulting model is simpler and more interpretable, especially in high-dimensional datasets with feature dependencies.

As with OLS and Ridge, the goal in Lasso is to minimize the cost function 4 to find the optimal coefficients:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left(\|\mathbf{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1\right)$$

However, unlike OLS and Ridge, this expression does give us an analytic expression, and must be computed using optimization techniques [10].

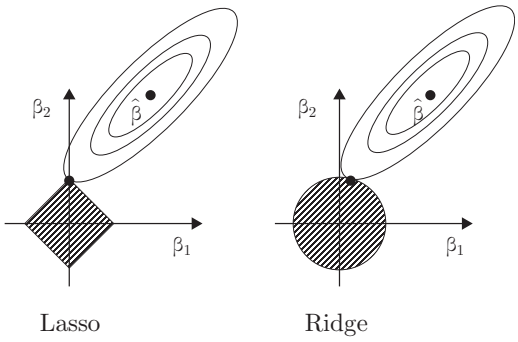


Figure 2: Comparison of the effect of regularization in Lasso and Ridge regression. Lasso uses the L1-norm, which can shrink some coefficients to zero, leading to feature selection. Ridge uses the L2-norm, shrinking all coefficients but retaining non-zero values, which helps manage multicollinearity without feature elimination.

Singular Value Decomposition

When $\mathbf{X}^\top\mathbf{X}$ is singular or ill-conditoned, using the psuedoinverse can provide a more stable optimization. This is where the singular value decomposition (SVD) is useful. For any $n \times m$ matrix $\mathbf{A}$, the SVD is given by:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices, and $\mathbf{\Sigma}$ is a diagonal matrix containing the singular values.[11]

The pseudoinverse of $\mathbf{A}$ can be performed by taking the pseudoinverse of $\mathbf{\Sigma}$ only. This involves flipping its dimensions and replacing every nonzero entry σ_i with $\frac{1}{\sigma_i}$

The pseudoinverse allows us to compute the coefficients for OLS as:

$$\hat{\boldsymbol{\beta}}_{\text{OLS}} = \mathbf{V}(\mathbf{\Sigma}^\top\mathbf{\Sigma})^{+}\mathbf{\Sigma}^\top\mathbf{U}^\top\boldsymbol{y}$$

Equally, we can obtain the coefficients for Ridge:

$$\hat{\boldsymbol{\beta}}_{\text{Ridge}} = \mathbf{V}(\mathbf{\Sigma}^\top\mathbf{\Sigma} + \lambda\mathbf{I})^{+}\mathbf{\Sigma}^\top\mathbf{U}^\top\boldsymbol{y}$$

(For full derivations see appendix D.4 and D.6)

DATA

Franke's function

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

This bivariate function by R. Franke (1975) [3] is given by the weighted sum of four exponentials, and it generates terrain-like data (see figure **F1** and **F2**). It is well-suited for testing interpolation, regression, and surface-fitting algorithms because of its continuous quality, with a varying curvature.

Terrain Data

The real world terrain data for this project is provided by the U.S. Department of the Interior Geological Surveys (USGS) EarthExplorer website [12]. The area of interest in our analysis is from Evje outside Stavanger.

Splitting the Data

Generally, in machine learning, we split the data into a `training_set` and a `test_set` before training the model. This is often done with a ratio of 80:20 in favor of the training set. [13]

Bootstrapping

The bootstrap method is a resampling technique in which samples are repeatedly drawn from a dataset with replacement, effectively generating multiple resampled datasets. The method is non-parametric, meaning it does not assume any distribution, making it a highly versatile method for model evaluation and performance estimation.

When evaluating statistics, there may be some uncertainty in the estimates. By using the bootstrap method, we are likely to obtain a more representative estimate of the given statistic. [14] (See appendix C.1 for the bootstrap algorithm)

k-fold Cross Validation

In k-fold cross validation, the `training_set` is split into k non-overlapping subsets, or *folds*. For each of the k iterations, one fold is used as the `validation_set` (also referred to as the `test_set`), while the remaining $(k-1)$ folds are used to train the model. This process is repeated until every

fold has been used for validation. After all k iterations, the performance of the model can be evaluated by calculating the average of a chosen statistic from each fold's result (see Figure 3).

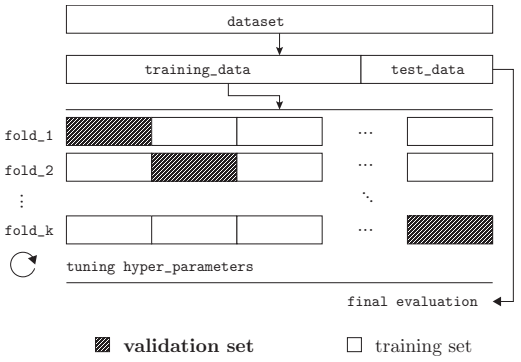


Figure 3: k-fold cross validation, where the average statistic can be calculated as $\frac{1}{k}\sum_{i=0}^{k-1}\hat{\theta}_i$.

Quantification methods

Mean squared error

A common way to quantify the error, is a statistic known as the mean squared error, which is given by

$$\text{MSE} = \frac{1}{n}\sum_{i=0}^{n-1}(z_i - \tilde{z}_i)^2 = \mathbb{E}[(\boldsymbol{z} - \tilde{\boldsymbol{z}})^2]$$

It quantifies the average squared difference between the true response $\boldsymbol{z}$ and the predicted response $\tilde{\boldsymbol{z}}$. The MSE can be further understood by a decomposition into a variance term and a bias term. The decomposistion is given by:

$$\mathbb{E}[(\mathbf{y} - \tilde{\mathbf{y}})^2] = \text{Bias}[\tilde{\mathbf{y}}] + \text{Var}[\tilde{\mathbf{y}}] + \sigma^2 \quad (5)$$

(See appendix D.9 for full derivation)

Bias-Variance Tradeoff

As shown in equation 5, the MSE can be decomposed into bias and variance. **Bias** reflects how consistently a model deviates from the true data, while **variance** measures the model's sensitivity to fluctuations in the training data. We aim for our model to have both low bias and low variance, however this balance is difficult to obtain, hence the **bias-variance tradeoff**. As model complexity increases, bias typically decreases and variance tends to rise. A less complex model might achieve a better balance, which can result in not selecting the model with the lowest MSE.

If we select a model with high bias, its ability to capture the underlying patterns in the data is weakened. This often results in low variance, as the model struggles to adjust to the complexity. This phenomenon is known as **underfitting**. Conversely, **overfitting** is often a result of high variance. This is where the model is to flexible and captures too much of the noise in the data.

Expectation and variance of OLS coefficients

We can further explore the behaviour of the optimal coefficients for OLS by analyzing the estimator's properties. The expectation of $\hat{\boldsymbol{\beta}}$ is:

$$\mathbb{E}[\hat{\boldsymbol{\beta}}] = \boldsymbol{\beta}$$

This shows that the OLS estimator is unbiased. It is however important to note that unbiasedness does not imply low variance. The variance for OLS is given by:

$$\text{Var}[\hat{\boldsymbol{\beta}}] = \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}$$

(See appendix D.8 for full derivation) It can therefore sometimes be useful to include a regularization to make the estimator biased, but reduce the variance.

It is now possible to derive a 95% confidence interval (CI) for each β_j, given as:

$$CI = \beta_j \pm 1.96\sigma^2(\beta_j) \quad (6)$$

This interval gives a range in which we expect the true β_j to lie with 95% confidence.

The R^2-score

Another statistic used to quantify the model's performance is the R^2-score. This statistic provides a measure of how well the model predicts unseen data, and is defined as:

$$R^2(\boldsymbol{z}, \tilde{\boldsymbol{z}}) = 1 - \frac{\sum_{i=0}^{n-1}(z_i - \hat{z}_i)^2}{\sum_{i=0}^{n-1}(z_i - \bar{z})^2}$$

The score is typically in the range $[0, 1]$, where 1 indicates a perfect fit, meaning that the model explains all variance in the data. In particular, if our score is r, we can interpret this as our model accounting for $r \times 100\%$ of the variance in the data.

INITIAL TESTING WITH FRANKE'S FUNCTION

We begin by generating data from Franke's function, and plotting it (**F1**, **F2**). F1 is the continous Franke's funtion, and F2 is Franke's function with a regu larized Gaussian noise, given by:

$$f_{\text{Franke}}(\boldsymbol{x}) + \eta \cdot \mathcal{N}(0, 1)$$

The plots show that it generates synthesized terrain-data, which would be meaningful to use for testing. We create the design matrix (Appendix A.1), split the data into a `training_set` and `test_set` using `train_test_split()` from `scikit learn` [15], before scaling the data.

> We use **subtract mean** scaling throughout this project: Each feature is centered by subtracting the mean of its column, but no further scaling by standard deviation is applied. This ensures that all features are centered, without altering their original variance. It also prevents any feature from dominating the model and maintains the data (i.e. the terrain's original curvature), by retaining its natural variability.

We now apply scaling on our training data, `X_train`, using subtract mean scaling to center each feature around zero. Similarly, `z_train` is centered by subtracting its mean, ensuring that the target values are also zero-centered. To maintain consistency between training and testing, the test data, `X_test`, is scaled using the mean values computed from the training data. This ensures that the test data is scaled in the same way as the training set.

Furthermore, in our analysis we use `intercept = False`, meaning that the intercept is included in the design matrix, rather than being calculated separately after training (See appendix B.1 for detailis). We test with Franke's function and find that the choice has a negligible impact on model performance, so we adopt this as a default for all regression models from now on.

Method of Inversion: SVD

To ensure stability in our regression methods (OLS and Ridge), we need a robust approach for computing the inverse when the matrices are singular or ill-conditioned. From the theory section on SVD, we know that SVD can be used to compute a pseudoinverse. In figure **F3**, we compare the performance of using the pseudoinverse and the standard matrix inverse for OLS. From polynomial degree 9 onward, SVD stabilizes the OLS method, whereas standard inversion leads to instability. Therefore, we employ SVD as a default for matrix inversion.

Tuning the Noise Parameter η

To mimic real terrain with Franke's function, we need to find a noise term η that best replicates the variations in a natural landscape. We can observe that as $\eta \to 0$, OLS outperforms L2-norm regularization. On the other hand, if η is increased, more regularization shows improvement in MSE. If we want Franke's function to mimic a real terrain we need a sweetspot for η. If η is too large, the terrain becomes unrealisticly noisy; if η is too small, the terrain becomes too smooth. Based on our observations, we propose that $\eta = 0.1$ could be our sweetspot. We will use this value when further analysing Franke's function.

PERFORMANCE EVALUATION OF FRANKE'S FUNCTION

Having established our fundamental methods, we are now prepared to conduct model analysis on Franke's function. We begin with k-fold cross validation which will help identifying the model that performs best with respect to the MSE.

To further evaluate the models, we employ the bootstrap method in conjunction with a bias-variance tradeoff analysis. This approach will deepen our understanding of the models' stability and generalization capabilities across different datasets.

Using k-fold cross validation

Before determining which hyperparameters yield the best model, we need to select an appropriate value for k. The choice of k is partly arbritrary, however $k = 10$ is often regarded as the best practice. It provides stable estimates of error (See figure **F4**) without excessive computational expense.

With the cross-validation framework established, we proceed to tuning of the hyperparameters λ and degree of complexity. We test the following values for λ:

$$\lambda \in \{\, 0 \,\} \cup \{10^k \mid k \in \{-9, -8, \ldots, 1\}\}$$

where $\lambda = 0$ corresponds to the OLS solution, and complexity $\in \{0, 1, \ldots, 20\}$. For each

combination of λ and complexity we apply cross-validation to obtain more accurate estimates of the MSE-values.

The resulting MSE-values are visualized in a heatmap (see figures **F6**, **F7**), with λ along the x-axis and complexity along the y-axis. The left plot corresponds to Ridge regression (with OLS included at $\lambda = 0$) and the right corresponds to Lasso regression. In both plots we have highlighted the combination yielding the minimal MSE-value.

The plot for Ridge and OLS shows that complexity = 18 and $\lambda = 10^{-9}$ result in a minimal MSE = 0.01034. Lasso regression performs worse than both Ridge and OLS, with a minimal MSE = 0.01433 for complexity = 18 and $\lambda = 10^{-7}$. This suggests that the L2-norm regularization could be benefical in estimating Franke's function. On the other hand, the L1-norm regularization appears to be less effective.

Applying bootstrap for bias-variance analysis

To further analyze the data, we employ the bootstrap method. We implement bootstrap resampling with OLS, exploring model complexity $\in \{1, 2, \ldots, 25\}$. Specifically, we use it to derive a bias-variance tradeoff analysis.

From figure **F8** we can see that the MSE and bias decreases significantly within the first four degrees of complexity. However, after degree four, the values stabilize, with the MSE showing a slight increase starting around degree 18.

It is important to also take the variance into consideration. For lower complexities the variance is close to zero, however it increases rapidly as the the model gets more complex. This rise in variance highlights the danger of overfitting with more complex models.

From a visual inspection, it seems that the optimal tradeoff lies within polynomial degree seven and ten.

Analysis of the optimized coefficients

The way we set up the design matrix has some unfortunate effects on the analysis of the coefficients. Specifically, we will not obtain the same β_i for a given permutation of x and y across models with different complexities. However, the plot will still be meaningful for a general analysis, as we can observe the overall tendencies in bias and variance.

As stated in theory section we can estimate the coefficients along with their confidence intervals. This could improve our understanding of the bias-variance tradeoff. As shown in figure **F5**, the spread of each coefficient-estimate (β_i) increases as model complexity grows, reflecting higher variance. Furthermore, analysis of each β_i within the same model complexity shows that increased complexities yield more spread, hence lower bias. This supports our bias-variance reasoning.

A note on the R^2-score

We will now use the R^2-score to verify our results. As discussed in the theoretical section, it provides a measure of how much of the variability in the data we are able to capture. From Figure **F9**, we can see that both OLS and Ridge regression achieve stable and high R^2-scores (around 90%).

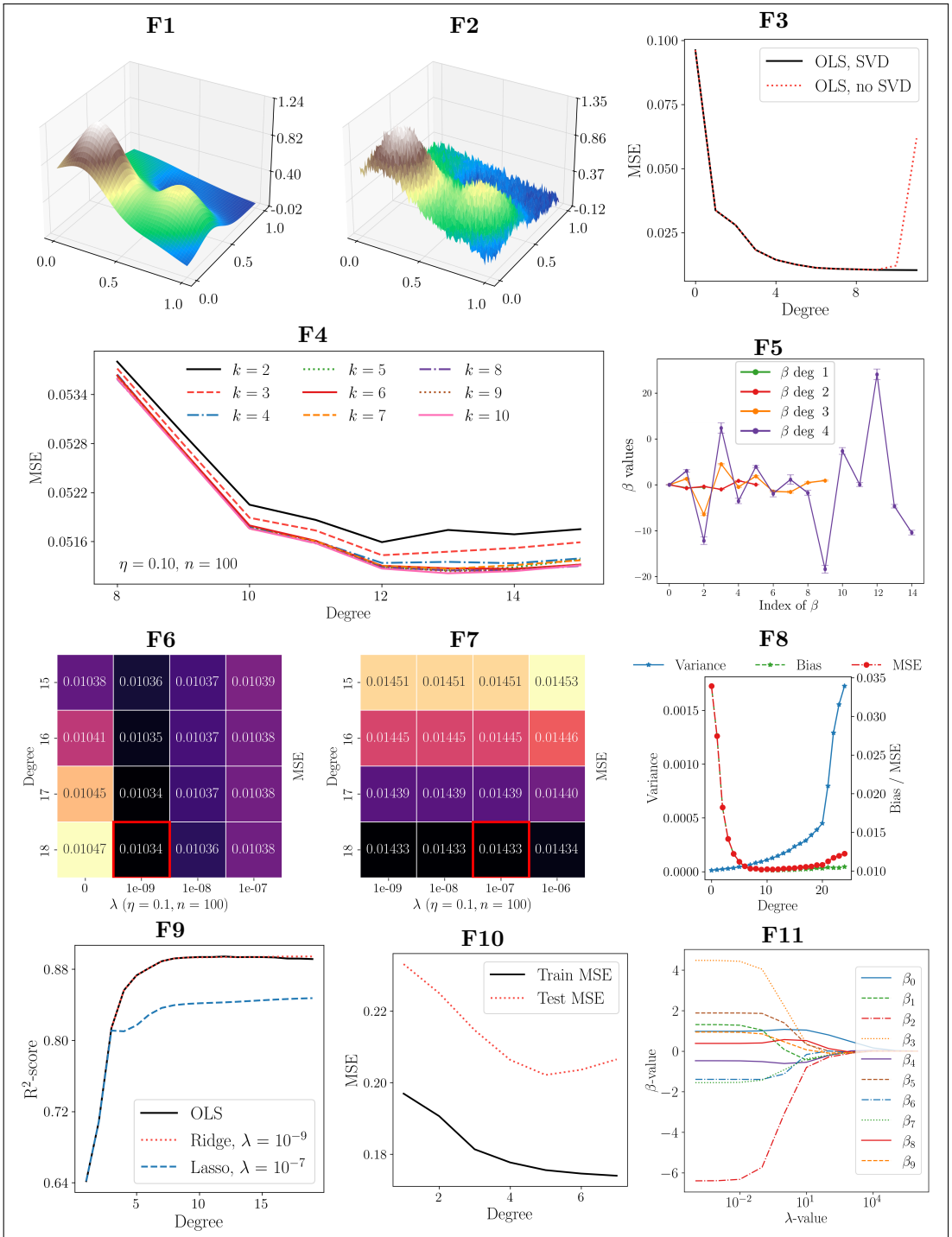
In contrast, Lasso regression performs worse, as also observed in the MSE analysis. This suggests that Lasso struggles to capture the variability effectively. Most likely due to how it handles sparsity in the model coefficients.

Concluding the initial testing

We now need to take all of our analysis into consideration: From the k-fold cross validation analysis, we find the best hyperparameters and model to minimize the MSE to be Ridge regression with regularization term $\lambda = 10^{-9}$ and complexity 18.

However, to perform regression on an arbitrary new set of data, a small MSE can not be the only component to determine the best-fitting model. Therefore we also take a look at the bias-variance analysis. This method of analysis favours smaller complexities, where the best tradeoff lies in the range of complexity seven to ten.

With this knowledge we are ready to test our methods on real terrain data.



Plots from the testing of Franke's function: **F1** A visualisation of the terrain generated by Franke's function without noise, for $n = 100$. **F2** The same function with added stochastic noise $\sim N(0, 1)$, $n = 100, \eta = 0.10$. **F3** MSE for OLS regression as function of the degree of complexity, with and without the Singular Value Decomposition (SVD) as inversion-method. **F4** k-fold cross-validation, demonstrating that higher values of k lead to more stable error estimates. $k = 10$ strikes a good balance between accuracy and computational efficiency. **F5** Coefficients with β with 95% CI for degree 4 **F6** Heat maps for Ridge regression. The red frame indicates the lowest MSE at 0.01034 for $\lambda = 10^{-9}$ and polynomial degree = 18. **F7** Heat map for lasso regression. Lowest MSE value 0.01433 for $\lambda = 10^{-7}$ and polynomial degree = 18. **F8** Plot of bias-variance tradeoff for OLS with bootstrap. Bias and MSE stabilize around degree 6-10, with the MSE climbing again from around degree 18. **F9** Both OLS and Ridge regression achieve stable and high R^2-scores (around 90%) at polynomial degree ~ 6. **F10** Train/test-plot for added noise $\sim N(0, 1)$, $\eta = 0.1$. Here we observed wrong values when using our own methods. `scikit learn`'s methods has been applied. We don't see that this error affects any of the other analysis. **F11** Shrinkage plot of β-coefficients for increasing λ in Ridge.

TESTING WITH REAL TERRAIN DATA

While Franke's function provides a good test case for evaluating our methods, it remains synthetic, and does not reflect the full complexity of real terrain. Having established that our methods work, they can be further applied. This will allow us to assess how well the best of our former parameters will work on new data, as well as fine-tune them further for improved accuracy in a more complex terrain.

During the testing for Franke's function we have established that the bootstrap method works excellent for assessing a bias-variance tradeoff analysis, as well as the k-fold cross-validation method for assessing the MSE (We look here at $\sqrt{MSE}$, as this provides us with the error given in meters). We will analyze a 50×50 section of the terrain illustrated in Figure **T1**, representing a complex mountainous region.

The terrain data is imported as a `.tif` file, which we convert into a 2D array using the `imageio` library in Python. Each pixel in the file has a resolution of 1 arc-second, approximately corresponding to 30 meters. Each entry in the array represents the height of a pixel in the selected area. Specifically, we extract data from the region defined by the coordinates $[200, 250]$ in both the x and y dimensions from the `stavanger.tif` file.

k-fold to assess error

Figure **T2** presents a heatmap comparing real-world performance of Ridge regression (OLS included at $\lambda = 0$) and Lasso. For Franke's function we identify the optimal parameters through a L2-norm regularization. However when applying the same models to real-world terrain, the optimal parameters are found without regularization (i.e. OLS). Interestingly, we find that the same degree of complexity (degree = 18) minimizes both Franke's function and the real-world terrain. The minimal MSE for the terrain is found to be 3.669 meters. We deem this to be a low MSE considering that we are examining a complex area.

Despite that some regularization to Franke's function has shown improvement in MSE (for $\eta = 0.1$), we observe that OLS outperforms any means of regularization with the mountainous area. We propose that this could be due to the nature of a real terrain. Noise $\eta = 0.1$ in combination with Franke's function might not capture the nature of a real terrain.

Bias-variance analysis for real terrain

As discussed for Franke's function, analyzing performance using only k-fold cross-validation for MSE might not capture the full picture. Therefore, we also examine the bias-variance tradeoff for real terrain data. Once again, we employ the bootstrap method, as it proved to be a powerful method for analyzing Franke's function.

Figure **T3** shows the bias-variance tradeoff for OLS for real-terrain data. By reducing the degree of complexity, we are able to reduce the variance significantly. By the eye analysis, shows that the best tradeoff lies in the range $[14, 17]$. So even though complexity 18 proved the lowest MSE for cross validation, it could be beneficial to reduce the complexity marginally to gain even more stable results.

CONCLUSION

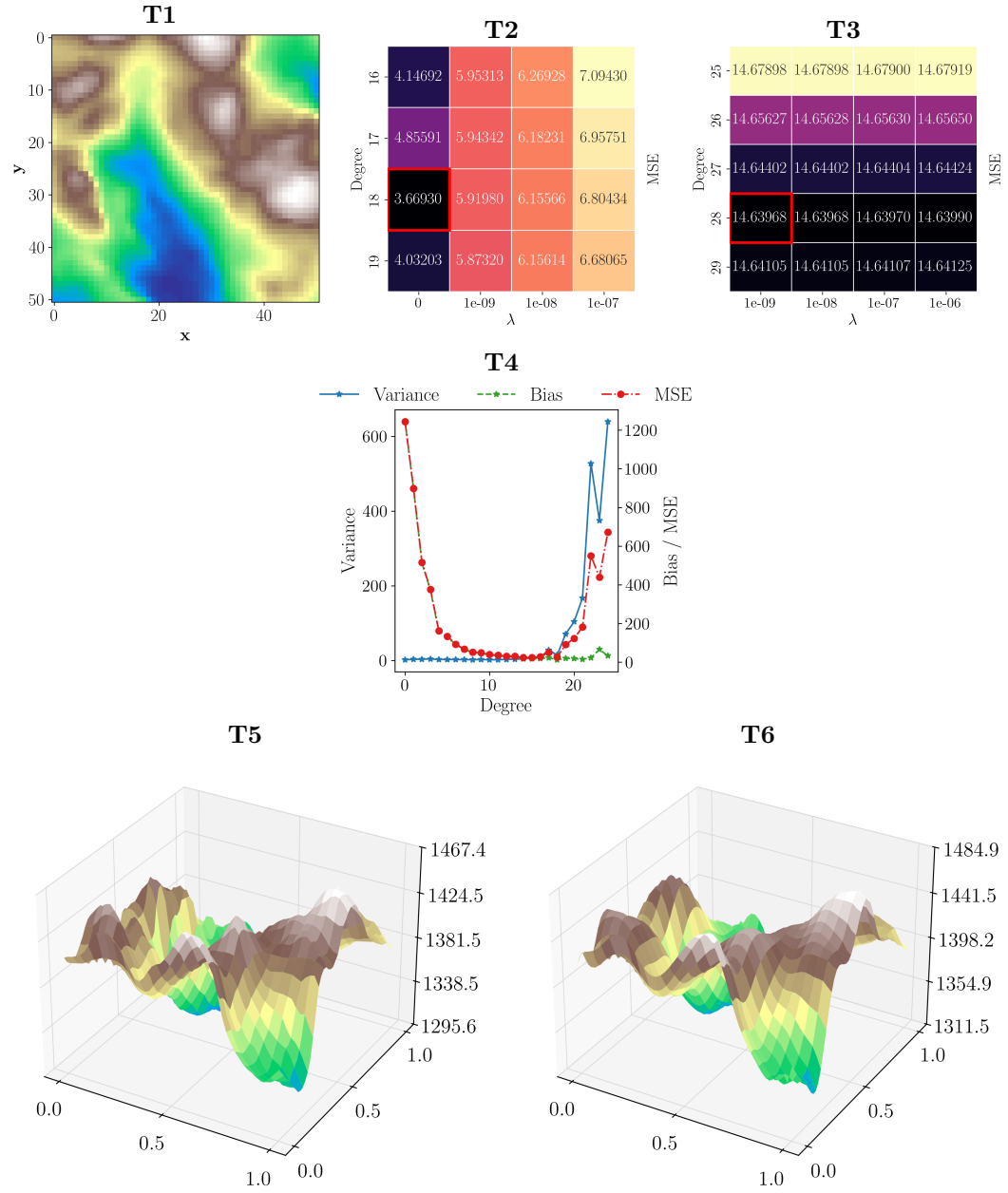
In this project, we have explored the use of three different polynomial regression methods: Ordinary Least Squares (OLS), Ridge, and Lasso regression. Our analysis was conducted on both synthetic data generated by Franke's function and real terrain data from a mountainous area outside Stavanger, Norway.

Our findings indicates that OLS outperforms both Ridge and Lasso in approximating the real world terrain. The best OLS model achieved a low $\sqrt{\text{MSE}}$ of 3.669 meters.

While it may be tempting to choose the model with the lowest MSE, we also considered a bias-variance tradeoff. This led us to conclude that a model with less degree of complexity might provide more stable estimates.

Given more time, we would conduct our analysis on multiple real-world terrains to develop a set of hyperparameters that could generalize well across a diverse range of landscapes.

Some final words: Although this has been a project in machine learning, it has involved quite a bit of **human learning**, too. We are three bachelor students who have now written our first research paper. It has been a steep uphill journey of valuable learning and great collaboration. Now, our hyperparameters are tuned, and our brains' models on paper-writing are trained, and we feel very ready to tackle a new assignment.



Plots from the testing of real terrain from Evje, outside Stavanger: **T1** Plot of the 50×50 terrain grid, representing a complex mountainous region. **T2** Heat map of Ridge/OLS regression MSEs for increasing regularization term λ and polynomial degree for the same terrain. The red square indicates the lowest MSE for Degree 18. This shows that OLS yields the best result in our model, with a $\sqrt{\text{MSE}}$ of 3.669 meters. **T3** Heat map of Lasso MSEs for increasing λ and degree. The red square indicates the lowest MSE for Degree 28. This shows that Lasso is unable to outperform OLS, with a $\sqrt{\text{MSE}}$ of 14.640 meters. **T4** Bias-variance tradeoff plot for OLS for the terrain data. Visual inspection indicates best tradeoff can be found between polynomial degrees $[14 - 17]$. **T5** 3D-visualization of the *real* landscape in figure T1. **T6** Visualization of the modelled landscape, with the chosen model and hyper parameters tuned: OLS with degree = 16, which is in accordance with our bias-variance tradeoff findings (T3).

REFERENCES

[1] Iqbal H. Sarker. Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, 2(3):160, May 2021.

[2] Trevor Hastie, Robert Tibshirani, and J. H. Friedman. *The elements of statistical*

learning: data mining, inference, and prediction. Springer series in statistics. Springer, New York, NY, 2nd ed edition, 2009.

[3] R. Franke. A Critical Comparison of Some Methods for Interpolation of Scattered Data. Technical Report NPS-53-79-003, Naval Postgraduate School, 1975.

[4] Morten Hjorth-Jensen. Applied Data Analysis and Machine Learning — Applied Data Analysis and Machine Learning, 2021.

[5] Eric W. Weisstein. Vandermonde Matrix. Publisher: Wolfram Research, Inc.

[6] Aditya Guntuboyina. Fall 2013 Statistics 151 (Linear Models) : Lecture Four.

[7] Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong. *Mathematics for machine learning*. Cambridge University Press, Cambridge, UK New York, NY, 2020.

[8] What Is Ridge Regression? | IBM, October 2023.

[9] Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. Publisher: [Royal Statistical Society, Oxford University Press].

[10] Niharika Gauraha. Introduction to the LASSO: A Convex Optimization Approach for High-dimensional Problems. *Resonance*, 23(4):439–464, April 2018.

[11] David C. Lay, Steven R. Lay, and Judi J. McDonald. *Linear algebra and its applications*. Pearson, Boston, 5. edition, global edition edition, 2016.

[12] Earth Resources Observation And Science (EROS) Center. Shuttle Radar Topography Mission (SRTM) 1 Arc-Second Global, 2017.

[13] V. Roshan Joseph. Optimal Ratio for Data Splitting. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 15(4):531–538, August 2022. arXiv:2202.03326 [cs, stat].

[14] Geir Storvik. Bootstrapping og simulering – Tilleggslitteratur for STK1100.

[15] Pedregosa et al. Scikit-learn: Machine Learning in Python. *JMLR*, 12(pp. 2825-2830), 2011.

APPENDIX

Regression Analysis and Resampling Techniques in Applied Machine Learning

A ALGORITHMS

A.1 Design matrix setup - Pseudocode

We set up the design matrix with the following algorithm:

```
idx = 0
for i in range(degree+1):
    for j in range(degree+1):
        if i+j <= degree:
            entry = (x**i * y**j)
            X[:, idx] = entry
            idx += 1
```

B PREPROCESSING STEPS

B.1 A note on the Intercept

In a standard polynomial regression analysis, the first coefficient, $\boldsymbol{\beta}_0$ is always multiplied by 1, eliminating the dependency on the variable $\boldsymbol{x}$. As a result, when all predictors are set to zero, $\boldsymbol{\beta}_0$ becomes the sole predictor, denoted as the `intercept`.

A common approach is to include the intercept in the `design_matrix` as a column of $1s$. This ensures that the regression model fits the intercept during training. Alternatively, the intercept can be excluded during training, and calculated afterwards using the this equation:

$$\beta_0 = \frac{1}{n}\sum_{i=0}^{n-1} y_i - \sum_{j=1}^{p-1} \mu_{\boldsymbol{x}_j}\beta_j$$

For OLS regression, the disticinton is marginal. However, in Ridge and Lasso regression, exclucing the intercept from the regularization term, can make a more significant difference.

C RESAMPLING TECHNIQUES

C.1 The bootstrap algorithm

The **algorithm** is as follows:

1. **Initialize:** Given a dataset $D = \{x_0, x_1, \ldots, x_{n-1}\}$, choose the number of bootstrap iterations `B`.
2. **Resample:** For each bootstrap iteration $b = 1, 2, \ldots, B$:
   - Randomly draw a sample of size n from D, with replacement, to form a resampled dataset D_b.
   - Compute the desired statistic $\hat{\theta}_b$ using D_b.
3. **Estimate:** After B iterations, calculate the final estimate of the statistic as the average of $\hat{\theta}_b$:

$$\hat{\theta}_{\text{bootstrap}} = \frac{1}{B}\sum_{b=0}^{B-1} \hat{\theta}_b$$

D MATHEMATICAL DERIVATIONS

D.1 Optimal coefficients for OLS

To show that the optimal coefficients $\hat{\beta}$ for OLS regression is

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{z}$$

we can minimize the following cost-function

$$C(\mathbf{X}, \boldsymbol{\beta}) = \{(\boldsymbol{z} - \mathbf{X}\boldsymbol{\beta})^T(\boldsymbol{z} - \mathbf{X}\boldsymbol{\beta})\}$$

We can rewrite $C(\mathbf{X}, \boldsymbol{\beta})$ to

$$C(\mathbf{X}, \boldsymbol{\beta}) = \boldsymbol{z}^T\boldsymbol{z} - \boldsymbol{z}^T\mathbf{X}\boldsymbol{\beta} - \boldsymbol{\beta}^T\mathbf{X}^T\boldsymbol{z} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}$$

We have that

$$(\boldsymbol{z}^T\mathbf{X}\boldsymbol{\beta})^T = \boldsymbol{\beta}^T\mathbf{X}^T\boldsymbol{z} \tag{7}$$

And since 7 is a scaler we have that the transpose of 7 is itself.
We can therefore rewrite $C(\mathbf{X}, \boldsymbol{\beta})$ to

$$C(\mathbf{X}, \boldsymbol{\beta}) = \boldsymbol{z}^T\boldsymbol{z} - 2\boldsymbol{\beta}^T\mathbf{X}^T\boldsymbol{z} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}$$

We can now optimize $C(\mathbf{X}, \boldsymbol{\beta})$, we can do this by setting $\frac{\partial C(\mathbf{X}, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0$.

$$\begin{aligned}\frac{\partial C(\mathbf{X}, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= -2\frac{\partial}{\partial \boldsymbol{\beta}}\boldsymbol{\beta}^T\mathbf{X}^T\boldsymbol{z} + \frac{\partial}{\partial \boldsymbol{\beta}}\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} \\ &= -2\boldsymbol{z}^T\mathbf{X} + 2\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\end{aligned}$$

Setting this equal to zero yields

$$\boldsymbol{z}^T\mathbf{X} - \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X} = 0$$

This rewrites to

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{z}$$

D.2 Optimal coefficients for Ridge

To show that the optimal parameters $\hat{\beta}$ for ridge regression is

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\boldsymbol{z}$$

we can minimize the following cost-function

$$C(\mathbf{X}, \boldsymbol{\beta}) = \{(\boldsymbol{z} - \mathbf{X}\boldsymbol{\beta})^T(\boldsymbol{z} - \mathbf{X}\boldsymbol{\beta})\} + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

We can rewrite $C(\mathbf{X}, \boldsymbol{\beta})$ to

$$C(\mathbf{X}, \boldsymbol{\beta}) = \boldsymbol{z}^T\boldsymbol{z} - \boldsymbol{z}^T\mathbf{X}\boldsymbol{\beta} - \boldsymbol{\beta}^T\mathbf{X}^T\boldsymbol{z} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

We have that

$$(\boldsymbol{z}^T\mathbf{X}\boldsymbol{\beta})^T = \boldsymbol{\beta}^T\mathbf{X}^T\boldsymbol{z} \tag{8}$$

And since 8 is a scaler we have that the transpose of 8 is itself.
We can therefore rewrite $C(\mathbf{X}, \boldsymbol{\beta})$ to

$$C(\mathbf{X}, \boldsymbol{\beta}) = \boldsymbol{z}^T\boldsymbol{z} - 2\boldsymbol{\beta}^T\mathbf{X}^T\boldsymbol{z} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

We can now optimize $C(\mathbf{X}, \boldsymbol{\beta})$, we can do this by setting $\frac{\partial C(\mathbf{X}, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0$.

$$\begin{aligned}\frac{\partial C(\mathbf{X}, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= -2\frac{\partial}{\partial \boldsymbol{\beta}}\boldsymbol{\beta}^T\mathbf{X}^T\boldsymbol{z} + \frac{\partial}{\partial \boldsymbol{\beta}}\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} + \lambda\frac{\partial}{\partial \boldsymbol{\beta}}\boldsymbol{\beta}^T\boldsymbol{\beta} \\ &= -2\boldsymbol{z}^T\mathbf{X} + 2\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X} + 2\lambda\boldsymbol{\beta}^T\end{aligned}$$

Setting this equal to zero yields

$$\boldsymbol{z}^T\mathbf{X} - \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X} - \lambda\boldsymbol{\beta}^T = 0$$

This rewrites to

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\boldsymbol{z}$$

D.3 Optimal predicts using OLS with SVD approach

To show that we can write the OLS solutions in terms of the eigenvectors of the orthogonal matrix $\mathbf{U}$ as

$$\tilde{\boldsymbol{z}}_{OLS} = \mathbf{X}\boldsymbol{\beta}_{OLS} = \sum_{j=0}^{p-1} \boldsymbol{u}_j \boldsymbol{u}_j^T \boldsymbol{z}$$

We can start with the OLS equation and combine it with the singular value decomposition, meaning

$$\tilde{\boldsymbol{z}}_{OLS} = \mathbf{X}\boldsymbol{\beta} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{z}$$
$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$$

Combining these yields

$$= \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T(\mathbf{V}\tilde{\boldsymbol{\Sigma}}^2\mathbf{V}^T)^{-1}\mathbf{V}\boldsymbol{\Sigma}^T\mathbf{U}^T\boldsymbol{z}$$
$$= \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\mathbf{V}(\tilde{\boldsymbol{\Sigma}}^2)^{-1}\mathbf{V}^T\mathbf{V}\boldsymbol{\Sigma}^T\mathbf{U}^T\boldsymbol{z}$$
$$= \mathbf{U}\boldsymbol{\Sigma}(\tilde{\boldsymbol{\Sigma}}^2)^{-1}\boldsymbol{\Sigma}^T\mathbf{U}^T\boldsymbol{z}$$

The expression $\boldsymbol{\Sigma}(\tilde{\boldsymbol{\Sigma}}^2)^{-1}\boldsymbol{\Sigma}^T$ is a matrix with the same dimensions as $\boldsymbol{\Sigma}$. If $\boldsymbol{\Sigma}$ has p singular values then $\boldsymbol{\Sigma}(\tilde{\boldsymbol{\Sigma}}^2)^{-1}\boldsymbol{\Sigma}^T$ has p ones along the diagonal, and n-p zero columns. This gives

$$\tilde{\boldsymbol{z}}_{OLS} = \mathbf{U}\mathbf{U}^T\boldsymbol{z} = \sum_{j=0}^{p-1} \boldsymbol{u}_j \boldsymbol{u}_j^T \boldsymbol{z}$$

D.4 Optimal coefficients using OLS with SVD approach

Under follows the full derivation for the optimal OLS coefficients, here we have used the SVD approch:

$$\begin{aligned}
\hat{\boldsymbol{\beta}}_{\text{OLS}} &= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{z} \\
&= (\mathbf{V}\boldsymbol{\Sigma}^\top\mathbf{U}^\top\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top)^{-1}\mathbf{V}\boldsymbol{\Sigma}^\top\mathbf{U}^\top\boldsymbol{z} \\
&= (\mathbf{V}\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}\mathbf{V}^\top)^{-1}\mathbf{V}\boldsymbol{\Sigma}^\top\mathbf{U}^\top\boldsymbol{z} \\
&= \mathbf{V}(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma})^{-1}\mathbf{V}^\top\mathbf{V}\boldsymbol{\Sigma}^\top\mathbf{U}^\top\boldsymbol{z} \\
&= \mathbf{V}(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma})^{-1}\boldsymbol{\Sigma}^\top\mathbf{U}^\top\boldsymbol{z}
\end{aligned}$$

D.5 Optimal predicts using Ridge regression with SVD approach

For Ridge regression we want to show that the corresponsing equation is

$$\tilde{\boldsymbol{z}}_{Ridge} = \mathbf{X}\boldsymbol{\beta}_{Ridge} = \sum_{j=0}^{p-1} \boldsymbol{u}_j \boldsymbol{u}_j^T \frac{\sigma_j^2}{\sigma_j^2+\lambda}\boldsymbol{z}$$

We start with the optimized cost-function for Ridge and combine it with SVD.

$$\tilde{\boldsymbol{z}}_{Ridge} = \mathbf{X}\boldsymbol{\beta}_{Ridge} = \mathbf{X}(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^T\boldsymbol{z}$$
$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$$

Combining these yields

$$\begin{aligned}
\tilde{\boldsymbol{z}}_{Ridge} &= \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T(\mathbf{V}\boldsymbol{\Sigma}^T\mathbf{U}^T\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T+\mathbf{I}\lambda)^{-1}\mathbf{V}\boldsymbol{\Sigma}^T\mathbf{U}^T\boldsymbol{z} \\
&= \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T(\mathbf{V}\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}\mathbf{V}^T+\mathbf{I}\lambda)^{-1}\mathbf{V}\boldsymbol{\Sigma}^T\mathbf{U}^T\boldsymbol{z} \\
&= \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T(\mathbf{V}\tilde{\boldsymbol{\Sigma}}^2\mathbf{V}^T+\mathbf{I}\lambda)^{-1}\mathbf{V}\boldsymbol{\Sigma}^T\mathbf{U}^T\boldsymbol{z} \\
&= \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T(\mathbf{V}\tilde{\boldsymbol{\Sigma}}^2\mathbf{V}^T+\lambda\mathbf{V}\mathbf{V}^T)^{-1}\mathbf{V}\boldsymbol{\Sigma}^T\mathbf{U}^T\boldsymbol{z} \\
&= \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T(\mathbf{V}(\tilde{\boldsymbol{\Sigma}}^2+\lambda\mathbf{I})\mathbf{V}^T)^{-1}\mathbf{V}\boldsymbol{\Sigma}^T\mathbf{U}^T\boldsymbol{z} \\
&= \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\mathbf{V}(\tilde{\boldsymbol{\Sigma}}^2+\lambda\mathbf{I})^{-1}\mathbf{V}^T\mathbf{V}\boldsymbol{\Sigma}^T\mathbf{U}^T\boldsymbol{z} \\
&= \mathbf{U}\boldsymbol{\Sigma}(\tilde{\boldsymbol{\Sigma}}^2+\lambda\mathbf{I})^{-1}\boldsymbol{\Sigma}\mathbf{U}^T\boldsymbol{z}
\end{aligned}$$

$\mathbf{\Sigma}(\tilde{\mathbf{\Sigma}}^2 + \lambda\mathbf{I})^{-1}\mathbf{\Sigma}$ is the same situation as for OLS, but instead of ones along the diagonal we get $\frac{\sigma_j^2}{\sigma_j^2+\lambda}$ along the diagonal. In total we therefor have

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

D.6 Optimal coefficients using Ridge regression with SVD approach

Under follows the full derivation for the optimal Ridge coefficients, here we have used the SVD approch:

$$\begin{aligned}
\hat{\boldsymbol{\beta}}_{\text{Ridge}} &= (\mathbf{X}^\top\mathbf{X} + \lambda I)^{-1}\mathbf{X}^\top\boldsymbol{z} \\
&= (\mathbf{V}\mathbf{\Sigma}^\top\mathbf{U}^\top\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top + \lambda I)^{-1}\mathbf{V}\mathbf{\Sigma}^\top\mathbf{U}^\top\boldsymbol{z} \\
&= (\mathbf{V}\mathbf{\Sigma}^\top\mathbf{\Sigma}\mathbf{V}^\top + \lambda I)^{-1}\mathbf{V}\mathbf{\Sigma}^\top\mathbf{U}^\top\boldsymbol{z} \\
&= (\mathbf{V}\mathbf{\Sigma}^\top\mathbf{\Sigma}\mathbf{V}^\top + \mathbf{V}\lambda\mathbf{V}^\top)^{-1}\mathbf{V}\mathbf{\Sigma}^\top\mathbf{U}^\top\boldsymbol{z} \\
&= (\mathbf{V}(\mathbf{\Sigma}^\top\mathbf{\Sigma} + \lambda I)\mathbf{V}^\top)^{-1}\mathbf{V}\mathbf{\Sigma}^\top\mathbf{U}^\top\boldsymbol{z} \\
&= \mathbf{V}(\mathbf{\Sigma}^\top\mathbf{\Sigma} + \lambda I)^{-1}\mathbf{V}^\top\mathbf{V}\mathbf{\Sigma}^\top\mathbf{U}^\top\boldsymbol{z} \\
&= \mathbf{V}(\mathbf{\Sigma}^\top\mathbf{\Sigma} + \lambda I)^{-1}\mathbf{\Sigma}^\top\mathbf{U}^\top\boldsymbol{z}
\end{aligned}$$

D.7 Expectation and Variance of the True Response Function

We can further understand the behaviour of $\boldsymbol{z} = f(\boldsymbol{x}) + \epsilon$ by looking at its expectation and variance. We first look at the expectation:

$$\begin{aligned}
\mathbb{E}[z_i] &= \mathbb{E}[f(\boldsymbol{x}_i) + \epsilon_i] \\
&= \mathbb{E}[f(\boldsymbol{x}_i)] + \mathbb{E}[\epsilon_i]
\end{aligned}$$

$f(\boldsymbol{x}_i)$ is non-stochastic and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ In total we therefore have

$$\mathbb{E}[z_i] = f(\boldsymbol{x}_i) = \sum_j x_{ij}\beta_j = \mathbf{X}_{i,*}\boldsymbol{\beta}$$

Now the variance:

$$\begin{aligned}
\text{Var}[z_i] &= \mathbb{E}[z_i^2] - \mathbb{E}[z_i]^2 \\
&= \mathbb{E}[f(\boldsymbol{x}_i + \epsilon_i)^2] - f(\boldsymbol{x}_i)^2 \\
&= \mathbb{E}[f(\boldsymbol{x}_i)^2] + \mathbb{E}[\epsilon_i^2] + \mathbb{E}[2f(\boldsymbol{x}_i)\epsilon_i] - f(\boldsymbol{x}_i)^2 \\
&= f(\boldsymbol{x}_i)^2 + \mathbb{E}[\epsilon_i^2] - f(\boldsymbol{x}_i)^2 \\
&= \mathbb{E}[\epsilon_i^2] = \text{Var}[\epsilon_i] + \mathbb{E}[\epsilon_i]^2 \\
&= \sigma^2
\end{aligned}$$

D.8 Expectation for beta and variance for OLS

We want to show that $\mathbb{E}[\hat{\boldsymbol{\beta}}] = \boldsymbol{\beta}$. This can be done with the following reasoning:

$$\begin{aligned}
\mathbb{E}[\hat{\boldsymbol{\beta}}] &= \mathbb{E}[(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{z})] \\
&= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbb{E}[\boldsymbol{z}] \\
&= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} \\
&= \boldsymbol{\beta}
\end{aligned}$$

Further we want to show that $\text{Var}[\hat{\boldsymbol{\beta}}] = \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}$. This can be done in the following derivation:

$$\text{Var}[\hat{\boldsymbol{\beta}}] = \text{Var}[(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{z})]$$

Set

$$\text{H} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$$

We now use the follwoing identity

$$\text{Var}[H\boldsymbol{z}] = H\text{Var}[\boldsymbol{z}]H^\top$$

Where $\text{Var}[\boldsymbol{z}] = \sigma^2$ We therefore have

$$\text{Var}[H\boldsymbol{z}] = \sigma^2 HH^\top$$

$HH^\top$ can be rewritten

$$\begin{aligned} HH^\top &= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{X}((\mathbf{X}^\top\mathbf{X})^{-1})^\top \\ &= ((\mathbf{X}^\top\mathbf{X})^{-1})^\top \\ &= (\mathbf{X}^\top\mathbf{X})^{-1} \end{aligned}$$

In total we have therefore shown that

$$\text{Var}[\hat{\boldsymbol{\beta}}] = \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}$$

D.9 Bias-variance decomposition

We want to show that MSE can be rewritten as follows

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

We can expand the MSE

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

We will now rewrite each term, first $\mathbb{E}[\boldsymbol{z^2}]$

$$\begin{aligned} \mathbb{E}[\boldsymbol{z^2}] &= \text{Var}[\boldsymbol{z}] + \mathbb{E}[\boldsymbol{z}]^2 \\ &= \text{Var}[f + \boldsymbol{\epsilon}] + \mathbb{E}[f + \boldsymbol{\epsilon}]^2 \\ &= \text{Var}[f] + \text{Var}[\boldsymbol{\epsilon}] + (\mathbb{E}[f] + \mathbb{E}[\epsilon])^2 \\ &= \sigma^2 + f^2 \end{aligned}$$

Second $\mathbb{E}[\tilde{\boldsymbol{z}}^2]$

$$\mathbb{E}[\tilde{\boldsymbol{z}}^2] = \text{Var}[\tilde{\boldsymbol{z}}] + \mathbb{E}[\tilde{\boldsymbol{z}}]^2$$

Last $\mathbb{E}[\boldsymbol{z}\tilde{\boldsymbol{z}}]$

$$\begin{aligned} \mathbb{E}[\boldsymbol{z}\tilde{\boldsymbol{z}}] &= \mathbb{E}[(f + \boldsymbol{\epsilon})\tilde{\boldsymbol{z}}] \\ &= \mathbb{E}[f\tilde{\boldsymbol{z}} + \boldsymbol{\epsilon}\tilde{\boldsymbol{z}}] \\ &= f\mathbb{E}[\tilde{\boldsymbol{z}}] + \mathbb{E}[\boldsymbol{\epsilon}]\mathbb{E}[\tilde{\boldsymbol{z}}] \\ &= f\mathbb{E}[\tilde{\boldsymbol{z}}] \end{aligned}$$

Putting it all together we have

$$\begin{aligned} \mathbb{E}[(\boldsymbol{z} - \tilde{\boldsymbol{z}})^2] &= \sigma^2 + f^2 + \text{Var}[\tilde{\boldsymbol{z}}] + \mathbb{E}[\tilde{\boldsymbol{z}}]^2 - 2f\mathbb{E}[\tilde{\boldsymbol{z}}] \\ &= \sigma^2 + \text{Var}[\tilde{\boldsymbol{z}}] + f^2 + \mathbb{E}[\tilde{\boldsymbol{z}}]^2 - 2f\mathbb{E}[\tilde{\boldsymbol{z}}] \\ &= \sigma^2 + \text{Var}[\tilde{\boldsymbol{z}}] + (f - \mathbb{E}[\tilde{\boldsymbol{z}}])^2 \end{aligned}$$

Where the last term is the bias squared, often just called bias (it will always be positive). In total we therefore have

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

In practice, the true function $\boldsymbol{f}$ is generally unknown. As a consequence, we must replace $\boldsymbol{f}$ with the observed response $\boldsymbol{z}$ which incorporates the noise $\boldsymbol{\epsilon}$. This allows us to estimate bias and variance using the available data, though it introduces additional uncertainty due to the noise that comes included with $\boldsymbol{z}$.