Project 1: FYS-STK4155

Regression analysis and resampling methods

Authors: Thomas Larsen Greiner and Hao Zhao

Github repositories for project scripts:

https://github.com/tagreine/school/tree/master/Project1

https://github.com/haozh7109/Machine_Learning_Course/tree/master/Project1

Abstract

The ultimate goal of machine learning methods is to provide automatically detection of patterns in data, and use these "learned" patterns to predict future data. Parameterization of synthetic- and real data was studied using a 2-dimensional function known as Franke's function and topographic data from southern Norway. The regression analysis included least squares (OLS), Ridge and Lasso regression. The different models were evaluated using known metrics such as mean squared error (MSE) and R^2-score. In addition, we analyzed and evaluated the model performance using a resampling method known as bootstrap. Using OLS estimation and bootstrap, the best-fit model was found to be using 8^{th} order polynomial fit for Franke's function, and a 10^{th} order for the real topographic data. Using Ridge and Lasso, we concluded that a penalty parameter value of 10^{-5} was reasonable. The OLS and Ridge estimations showed in overall a better model performance than Lasso, implying that these two methods performed better on this type of data.

Introduction

Regression methods are well known in the field of machine learning for parameterization of data. The ultimate goal of machine learning methods is to provide automatically detection of patterns in data, and use these "learned" patterns to predict future data. The aim of regression analysis using linear regression is to explain a response variable Y in terms of an explanatory variable $\boldsymbol{X}$ via the assumption of a linear relationship between $\boldsymbol{X}$ and $\boldsymbol{Y}$ (Wieringen, 2015). The response variable could be values from different sources, such as stock prices. In that case, regression models are applied to stock price data in order to predict future stock marked. In this project, the response variable included topographic data and the explanatory variable represents the spatial range of the data. Our objective was to study the behavior of three regression methods, which included ordinary least squares (OLS) regression, Ridge regression and Lasso regression. Parameterization was studied using a function known as Franke's function, and real topographic data from southern Norway. Results from the predictions was evaluated by computing the MSE with increasing basis expansion (OLS), i.e. introducing higher polynomial order. The Ridge and Lasso regression methods was evaluated using different values of the penalty parameter.

The following sections are structured by a short introduction to the theoretical background of the regression methods. The theory part also includes an explanation of resampling methods for machine

learning model evaluation, using bootstrap. Next follows results and discussion from the parameterization of the synthetic and real data, and model evaluation using bootstrap resampling. The last section follows a summary of conclusions.

Theoretical background

In machine learning, we commonly denote the jointly distributed set of measurements $\{(\boldsymbol{X}, \boldsymbol{Y})\}$ as the training set. The linear regression model $\boldsymbol{Y}$ is defined as

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \hat{\epsilon} \tag{1}$$

Where $\hat{\epsilon}$ is the noise term and $\boldsymbol{X}$ is a $(n\ x\ p)$-dimensional matrix, commonly referred to as the design matrix, and is defined as

$$\boldsymbol{X} = \begin{bmatrix} X_{0,0} & \cdots & X_{0,p-1} \\ \vdots & \ddots & \vdots \\ X_{n-1,0} & \cdots & X_{n-1,p-1} \end{bmatrix} \tag{2}$$

and $(p\ x\ 1)$-dimensional coefficient vector $\boldsymbol{\beta} = [\beta_0 \quad \cdots \quad \beta_{p-1}]^T$ which are the parameters we want to estimate in order to fit our model to the data. The β_0 parameter is commonly referred to as the bias term (or intercept) implying that in model (1) the first column of the design matrix is defined as a constant variable of one's: $\hat{X}_0 = [X_{0,0} \quad \cdots \quad X_{n,0}]^T = [1 \quad \cdots \quad 1]^T$. Suppose the random variable $\boldsymbol{Y}$ is normally distributed, with expected value of the model $E[\boldsymbol{Y}] = \boldsymbol{X\beta}$ and the variance $Var[\boldsymbol{Y}] = Var[\hat{\epsilon}] = \sigma^2$, hence $\boldsymbol{Y} \sim N(\boldsymbol{X\beta}, \sigma^2)$. From the regression model $\boldsymbol{Y}$ we then have two unknowns: $\boldsymbol{\beta}$ and σ^2. We fit our model to the training data and estimate the parameters by maximizing the log-likelihood function (Wieringen, 2015)

$$\boldsymbol{J}(\boldsymbol{\beta}, \sigma) = -n \ln\left(\sqrt{2\pi\sigma}\right) - \frac{1}{2\sigma^2} (\boldsymbol{Y} - \boldsymbol{X\beta})^T (\boldsymbol{Y} - \boldsymbol{X\beta}) \tag{3}$$

We maximize the log-likelihood in (3) by taking the partial derivative with respect to $\boldsymbol{\beta}$ and setting this to zero

$$\frac{\partial \boldsymbol{J}(\boldsymbol{\beta}, \sigma)}{\partial \boldsymbol{\beta}} = -\frac{1}{\sigma^2} (\boldsymbol{X^T Y} + \boldsymbol{X^T X\beta}) = 0 \tag{4}$$

If we assume that $\boldsymbol{X^T X}$ is non-singular, we have the estimated solution of the model

$$\widetilde{\boldsymbol{\beta}} = (\boldsymbol{X^T X})^{-1} \boldsymbol{X}^T \boldsymbol{Y} \tag{5}$$

The OLS solution to the model described in (1) then reads

$$\widetilde{\boldsymbol{Y}} = \boldsymbol{X}\widetilde{\boldsymbol{\beta}} = \boldsymbol{X}(\boldsymbol{X^T X})^{-1} \boldsymbol{X}^T \boldsymbol{Y} \tag{6}$$

The residual variance is estimated through taking the partial derivative of (4) with respect to σ and setting this to zero

$$\frac{\partial \boldsymbol{J}(\boldsymbol{\beta},\sigma)}{\partial \sigma} = -\frac{n}{\sigma} + \frac{1}{\sigma^3}(\boldsymbol{Y}-\boldsymbol{X\beta})^T\,(\boldsymbol{Y}-\boldsymbol{X\beta}) = 0 \tag{7}$$

which gives the estimate of the residual variance

$$\tilde{\sigma}^2 = \frac{1}{n}(\boldsymbol{Y}-\boldsymbol{X\beta})^T(\boldsymbol{Y}-\boldsymbol{X\beta}) \tag{8}$$

From the unique solution in (5) we can derive the variance and also its confidence interval of the $\widetilde{\boldsymbol{\beta}}$ parameter. The variance reads

$$\mathrm{Var}\big(\widetilde{\boldsymbol{\beta}}\big) = \mathrm{Var}\big((\boldsymbol{X^T X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}\big) = (\boldsymbol{X^T X})^{-1}\boldsymbol{X}^T\,\mathrm{Var}(\boldsymbol{Y})\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} = (\boldsymbol{X^T X})^{-1}\sigma^2 \tag{9}$$

For 95% confidence interval for $\widetilde{\boldsymbol{\beta}}$ approximately takes the form (James et al, 2013)

$$\widetilde{\boldsymbol{\beta}} \pm 2\sqrt{\mathrm{Var}\big(\widetilde{\boldsymbol{\beta}}\big)} \tag{10}$$

This means that there is a 95% chance that the interval in (10) will contain the true $\boldsymbol{\beta}$. An algorithm for computing the variance as presented in (9) is provided in Appendix A.

In case of singularity of the matrix $\boldsymbol{X^T X}$, we have to turn to other methods in order to give a solution to the problem. The two methods, which we will turn to, is the Ridge regression and Lasso regression. Ridge regression introduce a L^2 norm penalty on the parameters. The Ridge estimation with the penalty parameter can be expressed as follows

$$\boldsymbol{J}(\boldsymbol{\beta},\lambda) = (\boldsymbol{Y}-\boldsymbol{X\beta})^T(\boldsymbol{Y}-\boldsymbol{X\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta} \tag{11}$$

where λ is commonly referred to as the penalty parameter or shrinkage parameter. Taking the derivative with respect to $\boldsymbol{\beta}$ we get

$$\frac{\partial \boldsymbol{J}(\boldsymbol{\beta},\lambda)}{\partial \boldsymbol{\beta}} = (\boldsymbol{X^T Y} + (\boldsymbol{X^T X} + \lambda \boldsymbol{I})\boldsymbol{\beta}) = 0 \tag{12}$$

Which gives the estimated Ridge solution

$$\widetilde{\boldsymbol{\beta}} = (\boldsymbol{X^T X} + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{Y} \tag{13}$$

We see that the penalty parameter introduces a value equal to λ to the diagonal of $\boldsymbol{X^T X}$ which makes it non-singular. As mentioned, the Lasso method is another approach to avoid singularity issues. The Lasso estimate reads

$$\boldsymbol{J}(\boldsymbol{\beta}, \lambda) = (\boldsymbol{Y} - \boldsymbol{X\beta})^T (\boldsymbol{Y} - \boldsymbol{X\beta}) + \lambda \|\boldsymbol{\beta}\|_1 \tag{14}$$

where $\|\cdot\|_1$ denotes the L^1 norm, and involves a different penalty on the parameters $\boldsymbol{\beta}$ compared to Ridge regression. Due to the non-differentiability of the L^1 penalty at zero, the consequence is that we have no explicit term for the solution of $\boldsymbol{\beta}$ as in (5) and (13) (Wieringen, 2015). Instead, the estimation of $\widetilde{\boldsymbol{\beta}}$ is computed iteratively. Algorithms for computing the OLS and Ridge solution is provided in Appendix A. The Lasso solution is included within the algorithm but provided by the tools from Scikit learn.

In order to evaluate the model performance, i.e. how well it predict the training data, we turn towards metrics such as MSE and the R^2 score. The MSE is defined as

$$\mathrm{MSE}(y, \tilde{f}(x)) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \tilde{f}(x_i)\right)^2 \tag{15}$$

and the R^2 score is defined as

$$\mathrm{R}^2(y, \tilde{f}(x)) = 1 - \frac{\sum_{i=1}^{n} \left(y_i - \tilde{f}(x_i)\right)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{16}$$

where y is the true model, $\tilde{f}(x)$is the predicted model and $\bar{y}$ is the mean of the true model. However, for model evaluation in machine learning problems, the prediction error (MSE) measured on training data is not a good measure for model assessment (Hastie et al, 2001). There exists a couple of statistical tools, which provide model assessment in machine learning problems, known as resampling methods. The bootstrap method and the K-fold cross validation are the most common for model assessment in machine learning problems (James et al, 2013).

Suppose now that we split the jointly distributed set of measurements $\{(\boldsymbol{X}, \boldsymbol{Y})\}$ into a training set and an independent test set. The basic idea in bootstrapping is to randomly draw samples from your training set with replacement (i.e. the same sample can be drawn more than once), let's say B times, and refit the model for each bootstrap sample, and then apply the same set of predictors to test set. For bootstrap, the expected prediction error function reads (Hastie et al, 2001)

$$\mathrm{E}_{\mathrm{boot}}\left[\mathrm{L}\left(y_i, \tilde{f}^{*b}(x_i)\right)\right] = \frac{1}{B}\frac{1}{N} \sum_{i=1}^{B} \sum_{i=1}^{N} \mathrm{L}\left(y_i, \tilde{f}^{*b}(x_i)\right) \tag{17}$$

where $\mathrm{L}\left(y_i, \tilde{f}^{*b}(x_i)\right) = \left(y_i - \tilde{f}^{*b}(x_i)\right)^2$, i.e. the squared error between the model y_i and the predicted model $\tilde{f}^{*b}(x_i)$. Assessing the model performance is done by computing the average of the

expected prediction error on both the training set and the test set at each point x_i. The purpose of splitting the data into a training set and an independent test set, is that the prediction error for the training set tends to decrease using higher model complexity, but for the test set, the prediction error tends to have a turning point where the prediction error increases. As prediction increases for the test set using higher model complexity, the model is overfitting, i.e. the training model is not able to generalize the predictions. This is an inevitable consequence, and is related to the variance-bias tradeoff of the prediction, which we can derive from expected prediction error

$$
\begin{aligned}
E\left[L\left(\boldsymbol{Y},\tilde{f}(\boldsymbol{X})\right)\right] &= E\left[\left(\boldsymbol{Y}-\tilde{f}(\boldsymbol{X})\right)^2\right] = E\left[\boldsymbol{Y}^2 - 2\tilde{f}(\boldsymbol{X})\boldsymbol{Y} + \tilde{f}(\boldsymbol{X})^2\right] \\
&= E[\boldsymbol{Y}^2] - 2E\left[\tilde{f}(\boldsymbol{X})\boldsymbol{Y}\right] + E\left[\tilde{f}(\boldsymbol{X})\right]^2 + E\left[\tilde{f}(\boldsymbol{X})^2\right] - E\left[\tilde{f}(\boldsymbol{X})\right]^2 + E[\boldsymbol{Y}]^2 - E[\boldsymbol{Y}]^2 \\
&= E[\boldsymbol{Y}^2] - E[\boldsymbol{Y}]^2 + E\left[\tilde{f}(\boldsymbol{X})^2\right] - E\left[\tilde{f}(\boldsymbol{X})\right]^2 + E\left[\tilde{f}(\boldsymbol{X})\right]^2 - 2E\left[\tilde{f}(\boldsymbol{X})\boldsymbol{Y}\right] + E[\boldsymbol{Y}]^2 \\
&= \sigma^2 + \mathrm{Var}\left(\tilde{f}(\boldsymbol{X})\right) + \mathrm{Bias}^2\left(\tilde{f}(\boldsymbol{X}),\boldsymbol{Y}\right)
\end{aligned}
$$

where we have used the following relationships

$$\sigma^2 = \mathrm{Var}(\boldsymbol{Y}) = \mathrm{E}[\boldsymbol{Y}^2] - \mathrm{E}[\boldsymbol{Y}]^2$$

$$\mathrm{Var}\left(\tilde{f}(\boldsymbol{X})\right) = \mathrm{E}\left[\tilde{f}(\boldsymbol{X})^2\right] - \mathrm{E}\left[\tilde{f}(\boldsymbol{X})\right]^2$$

$$\mathrm{Bias}^2\left(\tilde{f}(\boldsymbol{X}),\boldsymbol{Y}\right) = \left(\mathrm{E}\left[\tilde{f}(\boldsymbol{X})\right] - \boldsymbol{Y}\right)^2 = \mathrm{E}\left[\tilde{f}(\boldsymbol{X})\right]^2 - 2\mathrm{E}\left[\tilde{f}(\boldsymbol{X})\boldsymbol{Y}\right] + E[\boldsymbol{Y}]^2$$

The first term is the irreducible error, which contain the variance of the input data. Algorithms for computing the MSE and R^2 score is provided in Appendix A. The bootstrap method was implemented using functions provided by Scikit learn, to split the model and evaluate the model performance. We also tried the OLS on the Franke's function with k-fold cross-validation, but we focused on the bootstrap method for all numerical tests.

Results and discussion

Parameterization of synthetic data

In this section, we apply the methods, as explained in the theory section, on both synthetic- and real topographic terrain data from the southern part of Norway. In order to test and evaluate parameterization of surface data using OLS, Ridge and Lasso, we apply the methods on a function known as "Franke's function", which will be our synthetic testing data. Franke's function is displayed in Figure 1, where it is defined in the range $(\mathrm{x}, \mathrm{y}) \in [0,1]$, with a spatial sampling of $\Delta \mathrm{x}, \Delta \mathrm{y} = 0.05$ Franke's function is a sum of four exponentials and reads as follows

$$f(x,y) = \frac{3}{4}\exp\left[-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right] + \frac{3}{4}\exp\left[-\frac{(9x+1)^2}{4} - \frac{9y+1}{4}\right] + \frac{1}{2}\exp\left[-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right] + \frac{3}{4}\exp[-(9x-4)^2 - (9y-7)^2]$$

Stochastic noise was introduced to Franke's function before prediction. The included noise was normal distributed $\sim N(0, \sigma^2)$ and using a magnitude of 10% of $\max[f(x,y)]$. The result of Franke's function with added noise is displayed in Figure 2. The implementation procedure for predicting the Franke's function using OLS, Ridge and Lasso estimations is summarized in three steps below:

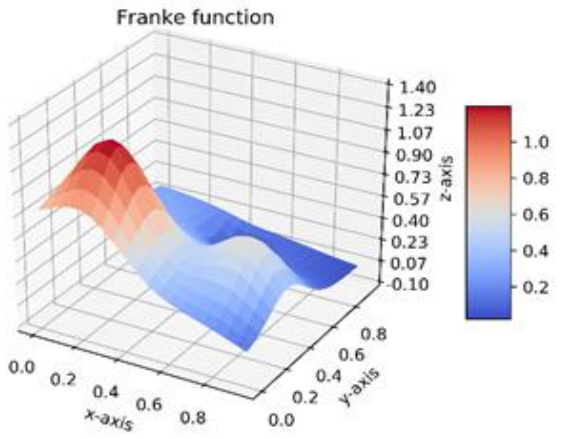

Figure 1: Franke's function where we defined the function at $(x,y) \in [0,1]$ and spatial sampling of $\Delta x, \Delta y = 0.05$.

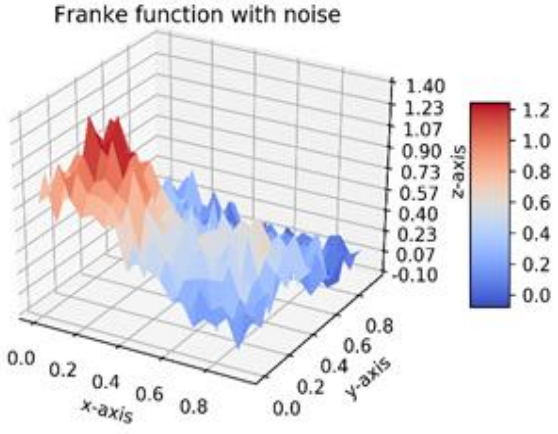

Figure 2: Franke's function with stochastic noise normally distributed $\sim N(0, \sigma^2)$ and with magnitude of 10% of $\max[f(x,y)]$.

1. Define our training set as $\{x_i, y_i, z_i\}$ where $i = 0,1, \dots, n$; $(x_i, y_i) \in [0,1]$; $z_i = f(x_i, y_i) + \epsilon_i$. The function z_i was transformed into a vector of shape $(n \cdot p \text{ x } 1)$.
2. The design matrix was generated by transforming x and y into vectors of shape $(n \cdot p \text{ x } 1)$ and including basis expansions and x and y dependencies. The design matrix with bias included reads

$$X = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x & y & xy & x^2 & y^2 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

3. Transform the two-dimensional function z to a vector of shape $(n * p \text{ x } 1)$. Predict the data z_i using the estimated parameters $\tilde{\boldsymbol{\beta}}$ from either OLS, Ridge or Lasso.

The predictions using OLS, Ridge and Lasso are displayed in Figure 3. For OLS estimation, we used a polynomial fit of 8th order. For the Ridge and Lasso predictions we used a penalty parameter of $\lambda = 10^{-5}$. From the MSE and R2 score presented in table 1, the prediction result from OLS gives a prediction slightly closer to true model compared to the Ridge and Lasso estimations.

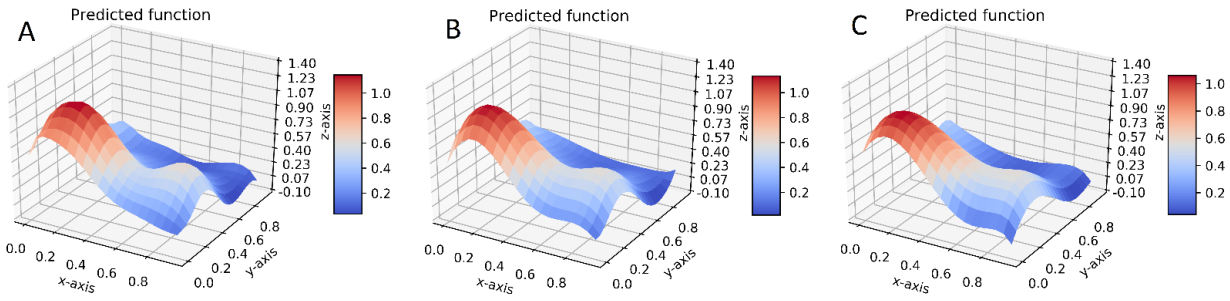

Figure 3: Prediction of the Franke's function using OLS solution (A), Ridge (B) and Lasso (C) on the noisy data displayed in Figure 2. The design matrix was set up to 8th order polynomial fit. The penalty parameter for Ridge and Lasso was set to $\lambda = 10^{-5}$.

Table 1: MSE score and R2 score from the OLS, Ridge and Lasso predictions of the function displayed in Figure 2, using a design matrix up to eight order. Ridge and Lasso methods was computed using a penalty parameter value $\lambda = 1 * 10^{-5}$. The MSE and R2 score shows that the predicted response from OLS lies closer to the true response than Ridge and Lasso does for this particular case.

Method	**MSE score**	R^2 **score**
OLS	≈ 0.013	≈ 0.859
Ridge	≈ 0.014	≈ 0.849
Lasso	≈ 0.015	≈ 0.834

From preliminary testing of Ridge and Lasso estimations, it was clear that model performance was very much dependent on the choise of the penalty parameter λ. For a large parameter, say $\lambda = 1.5$, the regression fit using Ridge and Lasso closely resembled a two dimensional plane, which indicates that we were shrinking the parameters $\tilde{\boldsymbol{\beta}}$ corresponding to second order polynomial and higher order, towards zero. As the penalty parameter was lowered, $\lambda \to 0$ we eventually obtain the solution from OLS. Eventually we settled for a value of $\lambda = 10^{-5}$, which gave reasonable results. By comparing the parameters from the different regression models, we observe that even while using a small value of the penalty parameter $\lambda = 10^{-5}$, most of the parameters are severely dampened (Figure 4). However, it does not affect the model performance compared to the OLS estimate. The similar performance could be due to the parameters from Ridge and Lasso are still within parameters confidence interval from the OLS estimate, as can be observed in Figure 5.

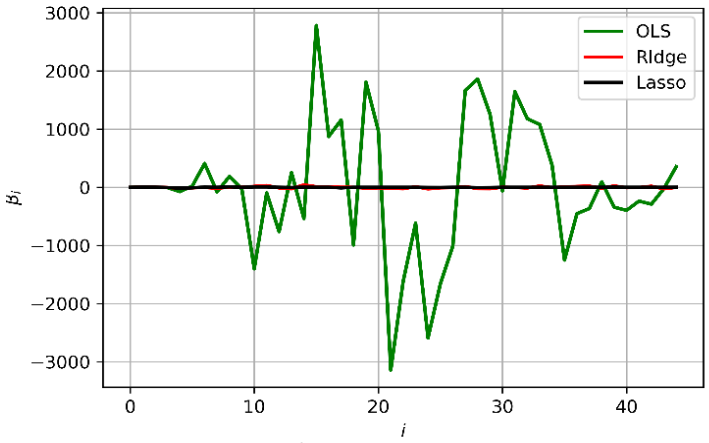

Figure 4: Comparison of the $\tilde{\boldsymbol{\beta}}$ parameters from OLS, Ridge and Lasso. We can see that the parameters are severely dampened by using Ridge and Lasso regression, but does not greatly decrease the performance as we see from the prediction result in Figure 3 and the MSE and R2 score in Table 1.

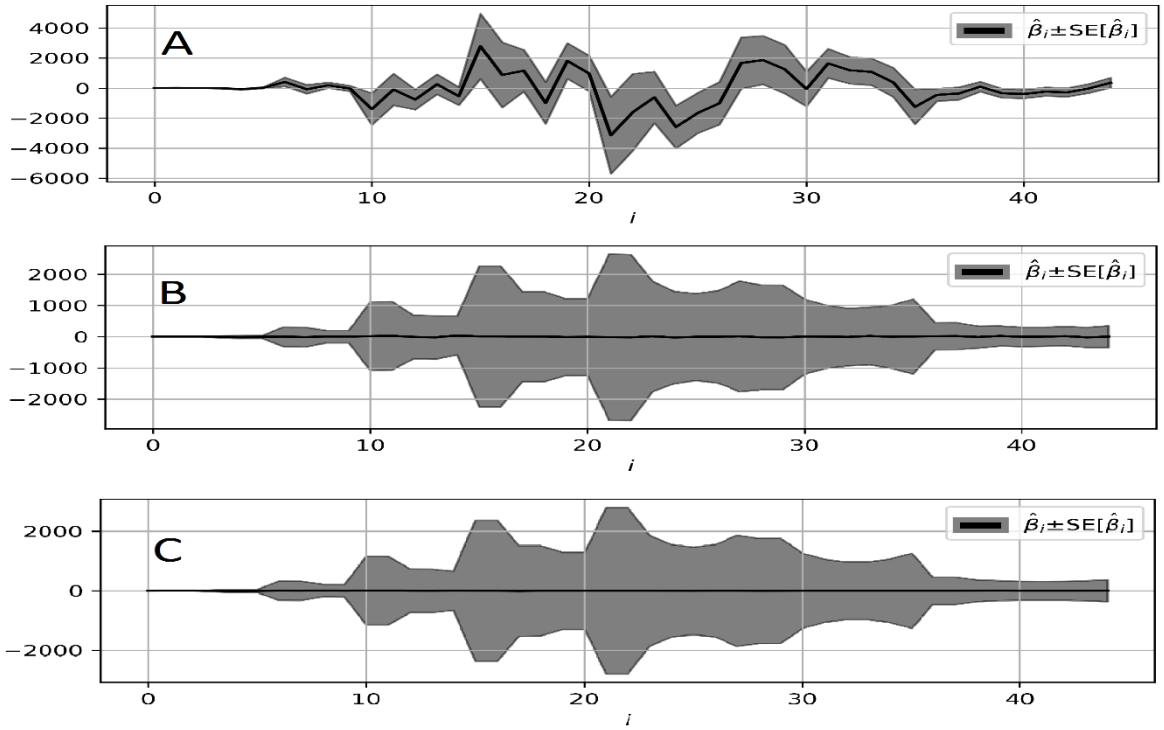

Figure 5: The $\tilde{\boldsymbol{\beta}}$ parameters from OLS (A), Ridge (B) and Lasso (C) and their corresponding confidence interval. The $\tilde{\boldsymbol{\beta}}$ parameters from Ridge and Lasso lies within (more or less) the confidence interval of the OLS. This means that even though the parameters in Ridge and Lasso are severely dampened they lie within the interval where we are 95% confident the true parameters lie.

One issue arises by using a very low penalty parameter, which relates to the Lasso estimate, as it is iterative. Using a very low value means that we need a large number of iterations for the Lasso to converge. This means that the Lasso method could be much more computationally expensive compared to OLS and Ridge.

By computing MSE- and R2 score wrt model complexity, we observe that by using higher order polynomials the performance of the models increases (Figure 6). For Ridge and Lasso estimates we see that the performance of the model decreases as we increase the value of penalty parameter λ. We also observe that the prediction error increases with more noise introduced. However, as mentioned in the theory section, training prediction error is not a good measure for model performance, because of possible overfitting. In order to evaluate and find the best-fit model, we introduce the bootstrap method for model evaluation and find the best-fit model based on the prediction error from the test set. In this case, the data was split using an 80/20 relationship on the training- and test set. We used 500 bootstrap iteration and computed polynomial fit up to 12th degree. The MSE results, and variance and bias decomposition from the OLS, Ridge and Lasso estimations using bootstrap resampling are displayed in Figure 9, Figure 10 and Figure 11 respectively. From the OLS estimation, the test set gives a best fit around polynomial degree of 6th order, where the minimum prediction error for this computation is located. In this case, using a 8th order polynomial fit as we used for the predictions is overfitting the data. From the bias and variance decomposition (Figure 9 right) we see that variance increases as model complexity increases, and the bias decreases up to 6th degree where the prediction error increases. The same trend is observed using the Ridge estimation. We observe the minimum for the test set when using a penalty parameter $\lambda = 10^{-5}$. The best-fit for Lasso estimation is more difficult to conclude, as the prediction error

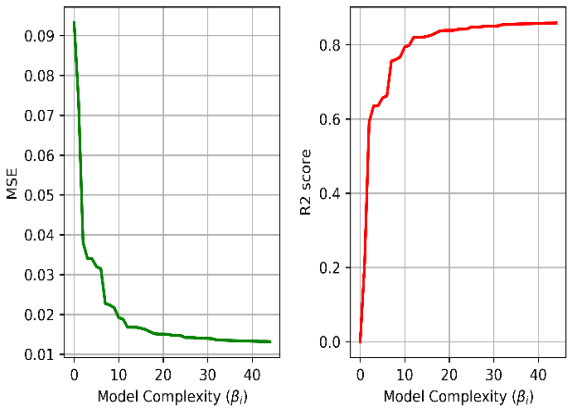

Figure 6: Computed MSE and R^2 score wrt model complexity, i.e. order of polynomial fit within the prediction, using OLS. The prediction error decreases, and model performance increases as higher order polynomials are introduced to the model.

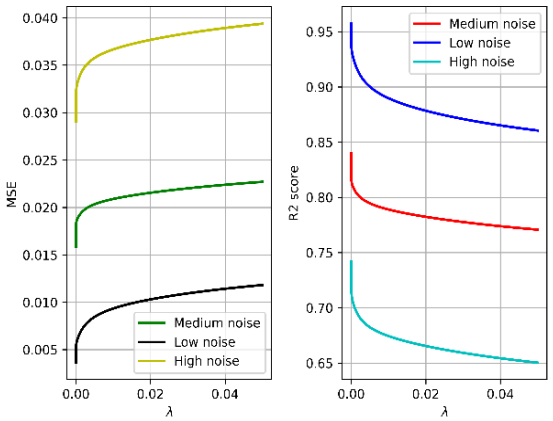

Figure 7: Computed MSE and R^2 score wrt λ, using Ridge regression. High, medium and low noise corresponds to a magnitude of 15%, 10% and 5% of $\max[f(x,y)]$. The prediction error decreases, and model performance increases with lower λ values.

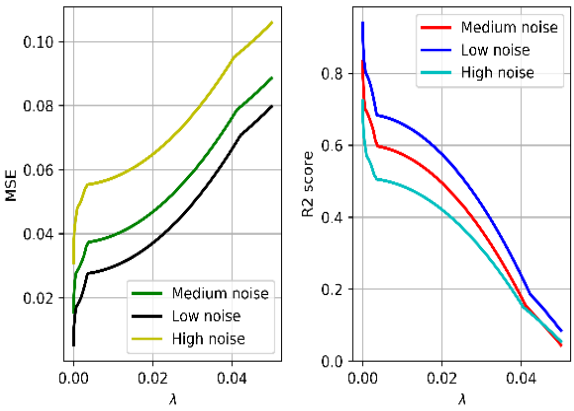

Figure 6: Computed MSE and R2 score wrt λ, using Lasso regression. High, medium and low noise corresponds to a magnitude of 15%, 10% and 5% of $max[f(x,y)]$. The prediction error decreases, and model performance increases with lower λ values.

increases with increasing penalty parameter value. However, the difference in prediction error between penalty parameter $\lambda = 0$ (OLS estimation) and $\lambda = 10^{-5}$ is not substantial.

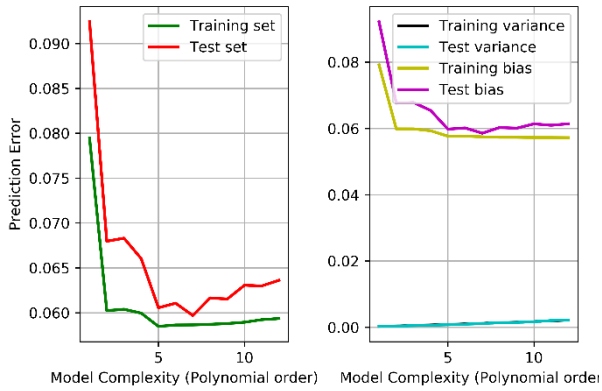

Figure 9: Computed prediction error (MSE) (left) for training set and test set wrt to model complexity using OLS estimation. The training/test split was set to 80/20 and using 500 bootstrap iterations. The prediction error decreases as model complexity increases considering the training set. The test set prediction error increases for polynomial fit above 6th degree. From the bias and variance decomposition (right figure) we see that variance increases as model complexity increases, and the bias shows a turning point at around 6th degree.

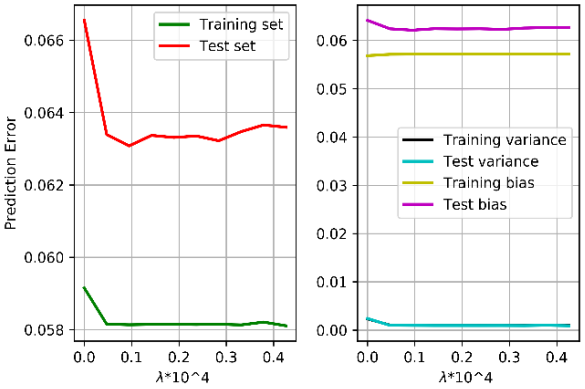

Figure 10: Computed prediction error (MSE) (left figure) for training set and test set wrt to penalty parameter λ using Ridge regression, and with the same training/test split as in OLS. We observe that prediction error decreases with increasing λ, but introduce a turning point for test set prediction error at $\lambda = 10^{-5}$.

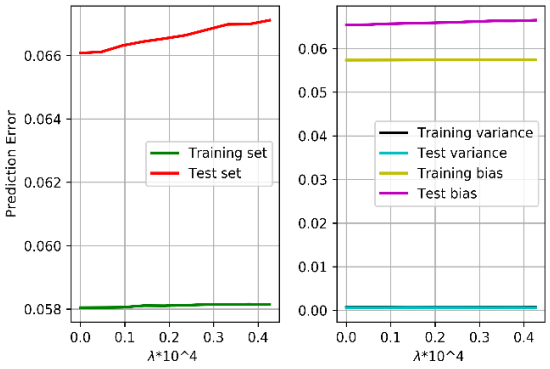

Figure 11: Computed prediction error (MSE) (left figure) for training set and test set wrt to penalty parameter λ using Lasso regression, and with the same training/test split as in OLS. Compared to the Ridge estimations, the prediction error from Lasso increases with increasing penalty parameter value, for both training and test set.

Parameterization of topographic data from southern Norway

We apply the same methods as we did on Franke's function above, in order to parameterize the topographic data. A small part of the data (50x50) was extracted. The full topographic data and the extracted patch and surface plot are displayed in Figure 12 from left to right. The data was defined in the range $(x, y) \in [0,49]$, with a spatial sampling of $\Delta x, \Delta y = 1$. In order to find the best-fit models for the different regression methods we computed the MSE using bootstrap resampling method.

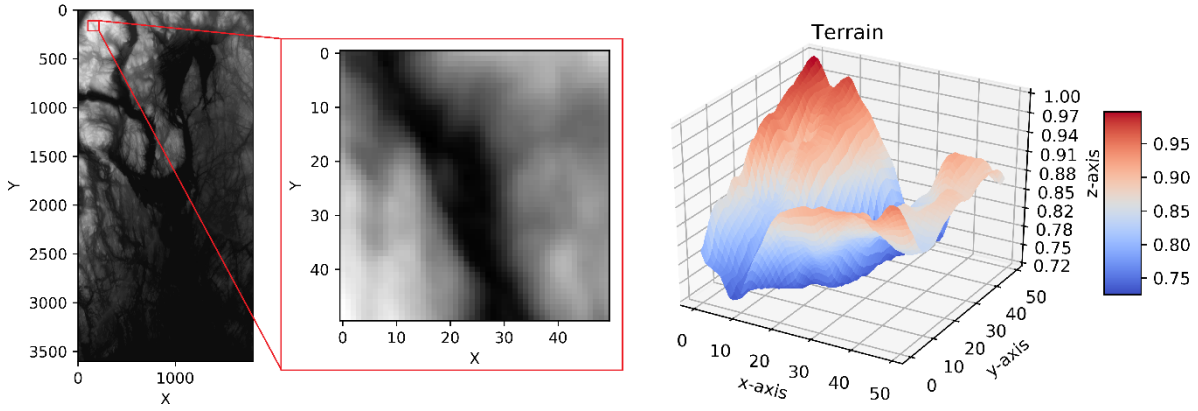

Figure 12: The terrain data from southern Norway (left), the small patch we extracted for training (middle) and the surface plot of the small patch (right).

The results from bootstrap resampling using OLS, Ridge and Lasso estimation are presented in Figure 13, 14 and 15 respectively. The training/test split was set to 80/20 and using 200 bootstrap iterations for all models. From the OLS and Ridge estimation, we observe that prediction error decreases as model complexity and penalty parameter increases respectively. A best-fit model for OLS estimation is observed at 10^{th} order polynomial fit. For Ridge estimation, we observe a best fit for a penalty parameter $\lambda = 10^{-5}$. The Lasso estimation gives a more ambiguous result, showing multiple best-fit models. In this case, choosing a penalty parameter $\lambda = 10^{-5}$ would be as good as any other. However, since the bootstrap method is a statistical method using random resampling, it may show another result using more bootstrap iterations. The results from OLS, Ridge and Lasso estimations are presented in Figure 16, Figure 17 and Figure 18 respectively. The MSE and R^2 score for all predictions using the final models is presented in table 2. We see that the OLS and Ridge estimations gives similar prediction values and values implying a closer fit to the true model, than what we see for Lasso estimation. Even though the metrics shows quite satisfying values, the results from parameterization of the topographic data are quite smooth and lacks details. It could be that introduction of other types of functions than polynomials would be better suited, such as wavelets or sinusoids.

Figure 13: Computed prediction error (MSE) (left figure) for training set and test set wrt to model complexity using OLS estimation. The training/test split was set to 80/20 and using 200 bootstrap iterations. The prediction error decreases as model complexity increases considering. The test set prediction error increases for polynomial fit above 10^{th} degree.

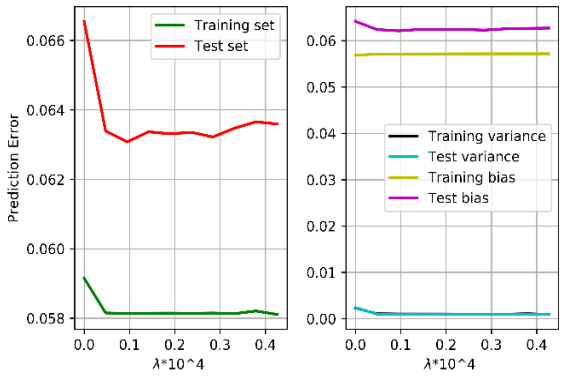

Figure 14: The prediction error (MSE) computed wrt penalty parameter (left) from Ridge estimation, by using 200 bootstrap iterations. Its respective bias and variance decomposition presented on the figure to the right. We observe a best-fit model by using a penalty parameter $\lambda = 10^{-5}$.

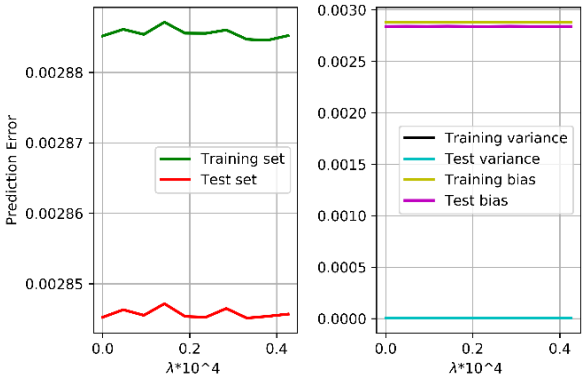

Figure 15: The prediction error (MSE) computed wrt penalty parameter (left) from Lasso estimation, by using 200 bootstrap iterations. Its respective bias and variance decomposition presented on the figure to the right. The best-fit model from Lasso estimation is ambiguous, as the prediction error shows best fit for more than one penalty parameter.

Table 2: MSE score and R2 score from the OLS, Ridge and Lasso estimations. OLS was computed using a polynomial fit of 6th order. Ridge and Lasso estimations were computed using a penalty parameter value $\lambda = 10^{-5}$. The MSE and R2 score shows that the predicted response from OLS and Ridge lies closer to the true response than Lasso does for this particular case.

Method	**MSE score**	R^2 **score**
OLS	≈ 0.0002	≈ 0.9496
Ridge	≈ 0.0002	≈ 0.9496
Lasso	≈ 0.0004	≈ 0.8946

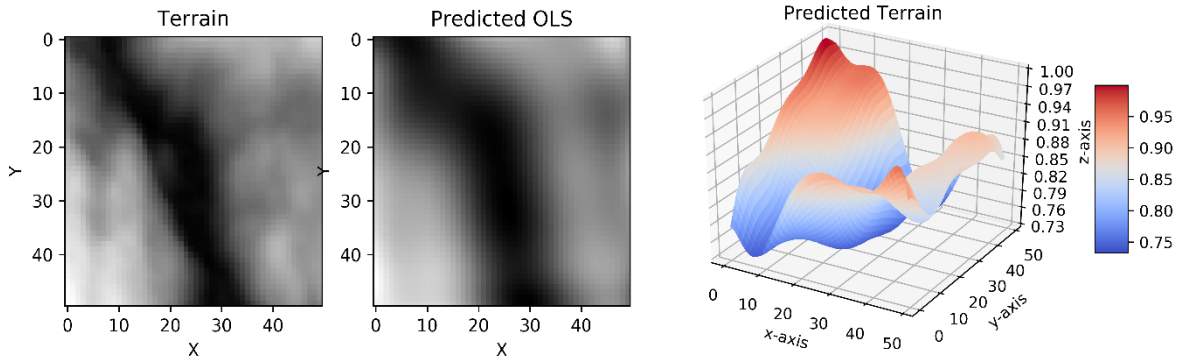

Figure 16: The small patch extracted for training (left), the predicted terrain using OLS estimation with 10th order polynomial fit (middle) and the surface plot of the predicted terrain.

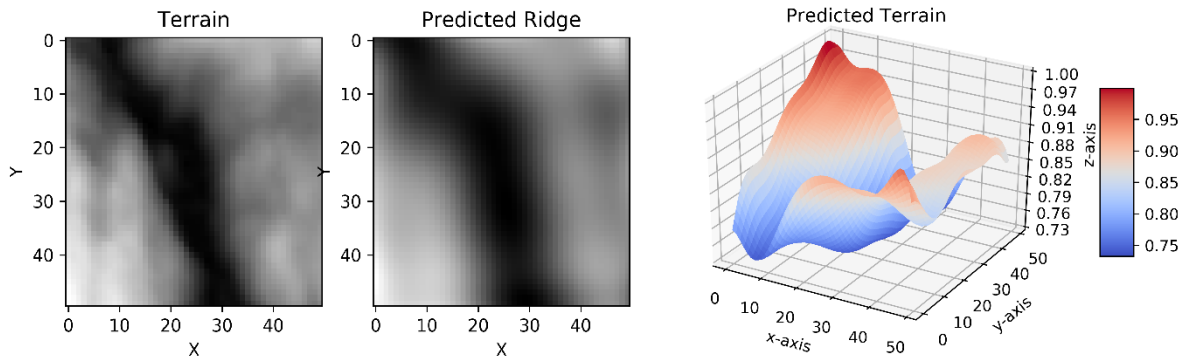

Figure 18: The small patch extracted for training (left), the predicted terrain using Lasso estimation with penalty parameter $\lambda = 1 * 10^{-5}$ (middle) and the surface plot of the predicted terrain.

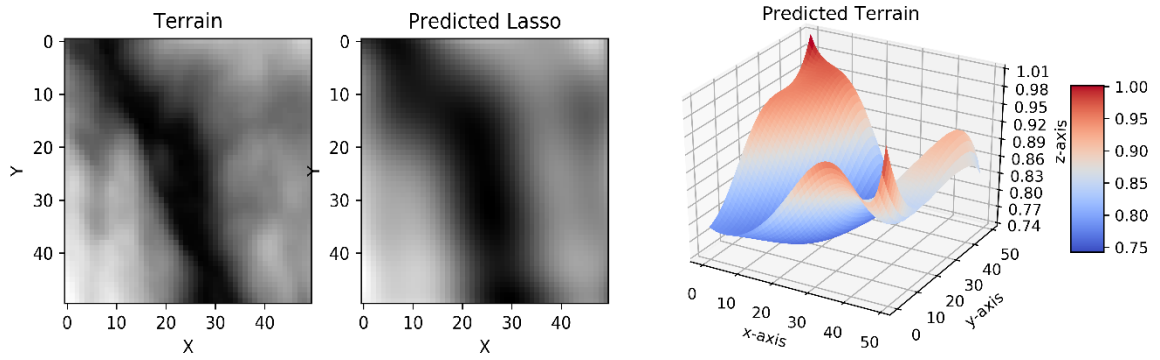

Figure 17: The small patch extracted for training (left), the predicted terrain using Ridge estimation with penalty parameter $\lambda = 1 * 10^{-5}$ (middle) and the surface plot of the predicted terrain.

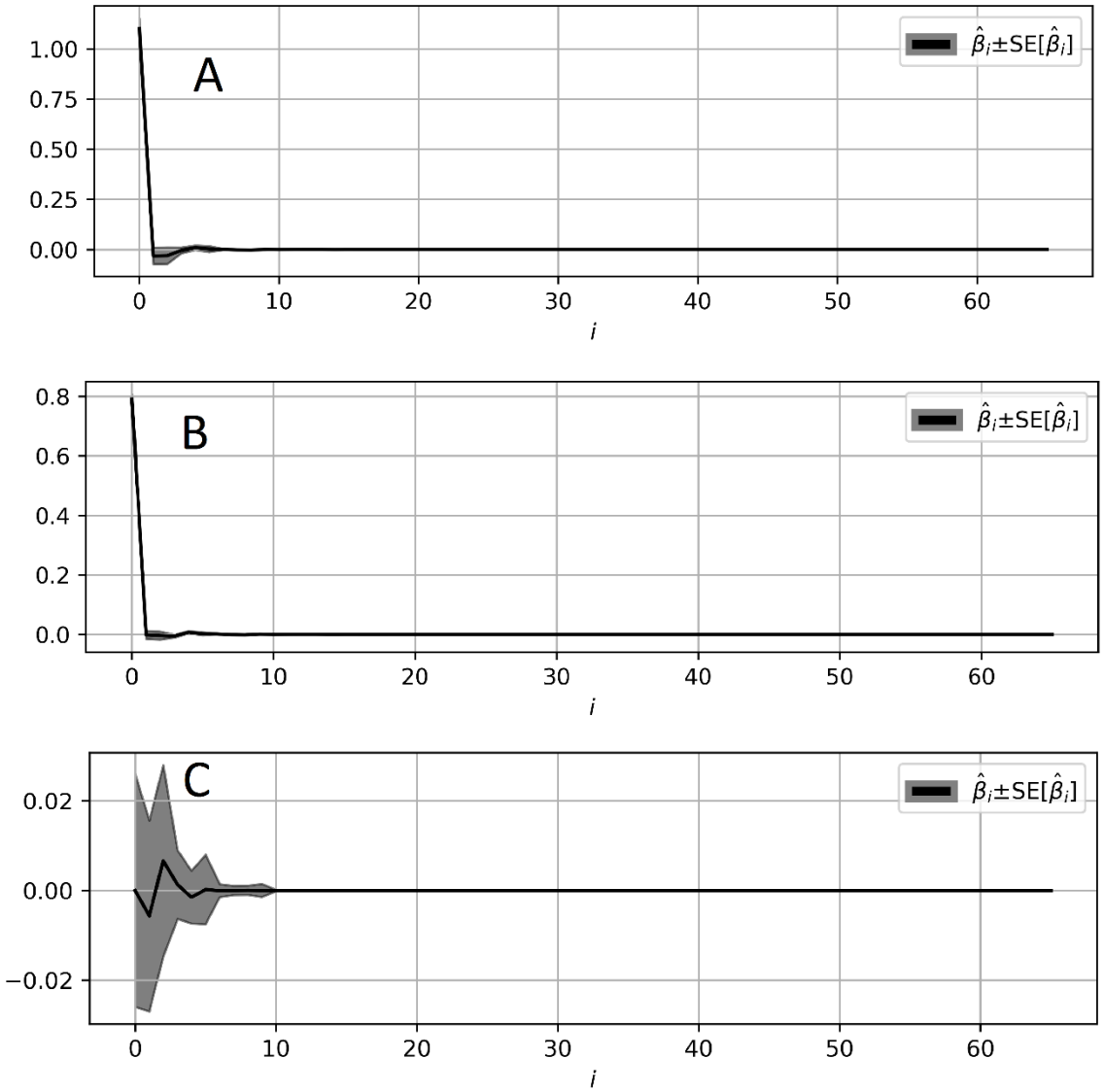

Figure 2: The $\widetilde{\boldsymbol{\beta}}$ parameters from OLS (A), Ridge (B) and Lasso (C) and their corresponding confidence interval. The confidence intervals are much narrower than we observed in the parameterization of Franke's function. This is due to the spatial range, which we defined the topographic surface.

Conclusion

In this project, we have implemented regression methods for parameterization of synthetic- and real data using Franke's function and topographic data from southern Norway. The regression analysis included least squares (OLS), Ridge and Lasso regression. The different models were evaluated using known metrics such as mean squared error (MSE) and R^2-score. In addition, we analyzed and evaluated the model performance using a resampling method known as bootstrap.

Using OLS estimation and bootstrap, the best-fit model was found to be using 8^{th} order polynomial fit for Franke's function, and a 10^{th} order for the real topographic data. Using Ridge and Lasso, we concluded that a using a penalty parameter equal to 10^{-5} was reasonable value. The OLS and Ridge estimations showed in overall a better model performance than Lasso, implying that these two methods performed better on this type of data.

References

Hastie, T., Tibshirani, R., Friedman, J. (2001). *The Elements of Statistical Learning*. New York, NY, USA: Springer New York Inc.

James G., Witten D., Hastie T., Tibshirani R. (2013). *An introduction to statistical learning: with applications in R*. New York :Springer

van Wieringen W. N. (2015). *Lecture notes on ridge regression*, arXiv:1509.09169.

Appendix A

The algorithm for computing the regression solutions, as presented in (5) and (13) in the theory section. The Lasso algorithm is provided by the Scikit learn toolbox linear_model:

```
def least_square(x,y,method='OLS',lamb=0):

    # exctract the shape of the model
    shape = x.shape
    n     = shape[0]
    p     = shape[1]

    # include bias within the data
    x0      = np.ones([n,1])

    X = np.concatenate((x0,x),axis=1)

    Xt   = np.transpose(X)

    if method == 'OLS':

        Hat  = np.dot(np.linalg.inv(np.dot(Xt,X)),Xt)
        beta = np.dot(Hat,y)
        y_   = np.dot(X,beta)

    if method == 'Ridge':
        I = np.eye(p+1)
        # No regularization on the the bias parameter
        I[0,0] = 0
        # Define hat matrix
        Hat  = np.dot(np.linalg.inv(np.dot(Xt,X) + lamb*I),Xt)
        beta = np.dot(Hat,y)
        y_   = np.dot(X,beta)

    if method == 'Lasso':
        Lasso_reg = linear_model.Lasso(lamb, fit_intercept=True, max_iter=150000)
        Lasso_reg.fit(X,y)
        beta = Lasso_reg.coef_.reshape(p+1,1)
        y_ = Lasso_reg.predict(X)

    return y_, beta
```

The algorithm for computing the MSE metric, as presented in (15) in the theory section:

```
def MSE_metric(y, y_):

    # extract the shape (dimensions) of the model
    shape = y.shape
    n     = shape[0]

    # compute the MSE score
    Err   = np.dot(np.transpose((y - y_)),(y - y_))/n

    return Err
```

The algorithm for computing the R^2 metric, as presented in (16) in the theory section:

```
def R2_metric(y ,y_):

    # extract the shape (dimensions) of the model
    shape = y.shape
    n     = shape[0]

    # compute the mean and store it as a vector
    y_m  = np.mean(y)
    y_mu = y_m*np.ones([n,1])

    A = np.dot(np.transpose((y - y_)),(y-y_))
    B = np.dot(np.transpose((y - y_mu)),(y-y_mu))

    # compute the R2 score
    R2 = 1 - A/B

    return R2
```

The algorithm for computing the variance of the $\tilde{\boldsymbol{\beta}}$ parameter, as presented in (9) in the theory section:

```
def Metrics_param(beta, y, y_, Design_matrix,compute_var=True):

    # Computing the variance and standard error of the parameters

    # if the variance is not assumed to be 1
    if compute_var == True:
        shapey_ = y_.shape
        shapeb  = beta.shape
        A       = 1/(shapey_[0] - shapeb[0] - 1)
        B       = np.dot((y - y_).T,(y - y_))
        Var     = np.multiply(A,B)
        M       = np.dot(Design_matrix.T,Design_matrix)
        VarBeta = np.multiply(np.linalg.inv(M),Var)
        VarBeta = np.diag(VarBeta).reshape(shapeb)
        StdBeta = np.sqrt(VarBeta).reshape(shapeb)

    # if the variance is assumed to be 1
    if compute_var == False:
        shapeb  = beta.shape
        M       = np.dot(Design_matrix.T,Design_matrix)
        VarBeta = np.linalg.inv(M)
        VarBeta = np.diag(VarBeta).reshape(shapeb)
        StdBeta = np.sqrt(VarBeta).reshape(shapeb)

    return VarBeta, StdBeta
```