FYS-STK4155

Applied Data Analysis and Machine Learning

Regression Analysis and Resampling Methods

Authors:
Celine Marie Solberg & Ole Petter Maugsten

October, 2022

Table of Contents

List of Figures

Abstract

In this report, regression analysis was performed on elevation data of Møsvatn Austfjell. The regression methods used were ordinary least squares, ridge and LASSO. These methods were also performed with bootstrapping and cross-validation as resampling techniques. The models were evaluated with mean squared errors and R^2-scores. The bias-variance tradeoff of the regression methods were also studied. The best model found for a subset of the elevation data was an OLS fit with 10-fold cross-validation. The existence of a better model could not be ruled out.

Introduction

Regression analysis is a powerful statistical method that allows one to examine the relationship between variables. Linear regressions seeks linear relations between independent variables, sometimes called features, and a dependent variable. By defining an error metric, the error of a fit can be minimized. This definition of error gives rise to a variety of regression models like the ordinary least squares, ridge and LASSO. The different regression methods tackle the problem differently. Some methods, like the LASSO, can even remove features it deems redundant. After obtaining a fitted model to some data, the relationships between variables will allow us to predict future outcomes. Examples of predictions are probability of diseases, weather forecast or rise or fall of stocks. Predictions of variables is the core of all machine learning.

In this text, we will use different regressions methods to study elevation data of Møsvatn Austfjell, Norway. Resampling techniques will be used to improve the precision of our models. We will evaluate our model with mean squared errors and R^2-scores. This should give us better insight to the elevation data.

Theory

Linear Regression

Linear regression is a common type of predictive analysis. It is a linear approach to model the relationship between variables. Say one has obtained the data set $\boldsymbol{z}(\boldsymbol{x})$, and wants to model the relationship between $\boldsymbol{z}$ and $\boldsymbol{x}$. It is assumed that the data $\boldsymbol{z}$ is a function f that describes the relationship with some noise $\epsilon = N(0, \sigma^2)$,

$$\boldsymbol{z}(\boldsymbol{x}) = f(\boldsymbol{x}) + \epsilon. \tag{1}$$

The model $\tilde{\boldsymbol{z}}$ approximates the function $f(\boldsymbol{x})$. The model consists of a set of weighted features. These features are ordered as columns in a design matrix $\boldsymbol{X}$. Each feature is weighted by a parameter β. The model is a product of the design matrix and the parameter column vector,

$$\tilde{\boldsymbol{z}} = \boldsymbol{X}\beta. \tag{2}$$

The model is linear in the features. When performing a regression, the task is to find the optimal parameters $\hat{\beta}$. The optimal parameters are the ones that minimizes the error. The error metric can be defined in many ways, and each way optimize the parameters differently.

With the model $\tilde{\boldsymbol{z}}$ approximated for the function $f(\boldsymbol{x})$, the expected value of the data z_i at a given point is the model $\tilde{z}_i$ evaluated at the same point. The variance of the same data point is the variance of the noise ϵ. See appendix for this.

Ordinary Least Squares

The ordinary least squares (OLS) method is a type of linear regression. It minimizes the residual sum of squares between the data $\boldsymbol{z}$ and the model $\tilde{\boldsymbol{z}}$ and is given by

$$C(\beta) = \sum_{i=1}^{N} \left(z_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2. \tag{3}$$

Equation (3) is called a cost function. It can be rewritten into matrix form:

$$C(\beta) = (\boldsymbol{z} - \boldsymbol{X}\beta)^T(\boldsymbol{z} - \boldsymbol{X}\beta). \tag{4}$$

Setting the first derivative of equation(4) with respect to the parameters β equal to zero gives

$$\boldsymbol{X}^T(\boldsymbol{z} - \boldsymbol{X}\beta) = 0, \tag{5}$$

thus obtaining the unique solution

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

for the optimal parameters $\hat{\beta}$. It is assumed that $\boldsymbol{X}^T\boldsymbol{X}$ is positive definite.

The predicted values are then given as

$$\tilde{\boldsymbol{z}} = \boldsymbol{X}\hat{\beta} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{z}. \tag{7}$$

The expected value of the optimal parameters β. The variance can be derived from equation (6) and is given by

$$Var(\hat{\beta}) = \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1} \tag{8}$$

where σ^2 is the noise in the data. See appendix for this.

Mean Square Error and R^2 score

How does one evaluate a model? The mean square error (MSE) and R^2 score are two ways of measuring how much the predicted values deviates from the real values.

The MSE is defined as

$$MSE(\boldsymbol{z}, \tilde{\boldsymbol{z}}) = \frac{1}{n}\sum_{i=0}^{n-1}(z_i - \tilde{z}_i)^2 \tag{9}$$

where $\boldsymbol{z}$ is the real values and $\tilde{\boldsymbol{z}}$ is the predicted values. A high MSE is considered bad.

The R^2 score is defined as

$$R^2(\boldsymbol{z}, \tilde{\boldsymbol{z}}) = 1 - \frac{\Sigma_{i=0}^{n-1}(z_i - \tilde{z}_i)^2}{\Sigma_{i=0}^{n-1}(z_i - \bar{y})^2} \tag{10}$$

where $\bar{\boldsymbol{z}}$ is the mean value of $\boldsymbol{z}$. R^2-scores usually range from 0 to 1. A high R^2-score is considered good. An R^2-score of 1 indicates that the regression model fit the data perfectly.

Singular Value Decomposition

The singular value decomposition decomposes a matrix into three different components. Given a matrix X, the SVD gives

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}} \tag{11}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices with orthonomal eigenvectors of $\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}$ and $\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$ respectively [Hui 2022]. $\boldsymbol{\Sigma}$ is a diagonal matrix composed of singular values in decreasing order.

If the matrix $\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$ is a singular matrix, then it can not be inverted and thus creates a problem when trying to perform a linear regression. SVD can be used to approximate the inverse by setting $1/\sigma_i = 0$ where $\sigma_i = 0$, allowing linear regression to be successfully performed.

Ridge Regression

The ridge regression method reduces the complexity of a model by shrinking the parameters β [Team 2022]. The shrinkage is done by imposing a penalty on the parameter's size [Hastie et al. 2017]. The method can be used to solve issues of multicollinearity and overfitting. In other words, the ridge regression performs a penalized residual sum of squares,

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}}\Big\{\sum_{i=1}^{N}(z_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}\beta_j^2\Big\}. \tag{12}$$

In matrix form:

$$\hat{\beta}^{ridge} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{z}. \tag{13}$$

The ridge parameters is a scaled version of the OLS parameters,

$$\hat{\beta}^{ridge} = \frac{1}{1+\lambda}\hat{\beta}^{ols} \tag{14}$$

LASSO Regression

The LASSO regression method is similar to ridge regression in the way that it imposes a penalty on the parameters β. The lasso estimate is defined as

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}}\Big\{\frac{1}{2}\sum_{i=1}^{N}(z_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}|\beta_j|\Big\}. \tag{15}$$

While ridge regression uses the L1-norm in the penalty term, LASSO uses the L2-norm. This means that ridge can shrink the parameters towards zero, while LASSO can force the parameters to be equal to zero. In other terms, the LASSO can remove features it deems redundant.

Resampling Methods

The fundamentals of creating a machine learning algorithm involve splitting data into train and test data. A model is made using the training data, and later applied on the test data. Sometimes a model can give good results because of a lucky choice of train and test data. To verify that a model is actually a good model regardless of the choice of train and test data, resampling methods can be applied to give a more accurate estimate of the model's performance.

Bootstrap

The main consept of the boostrap method is to randomly sample the data we have, find the parameters β and calculate the MSE, repeatedly. The number of bootstraps (repetitions) has to be defined. For each bootstrap, datapoints are randomly drawn from the original dataset, meaning that a data point can reappear in future drawn samples as well. Figure 1 shows an illustration of the bootstrap method.

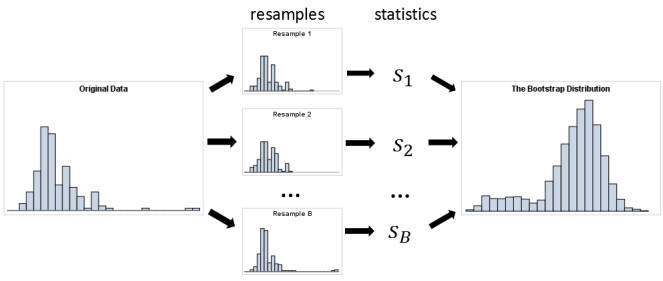

Figure 1: An illustration of the bootstrap method. Each resample contains randomly drawn datapoints from the orignal data. Resample S_1 to S_B represents the number of bootstraps.

Cross-validation

Cross-validation is another way of evaluating how good an algorithm/model is. The fundamentals of the cross-validation method involve shuffling the original data into a random order and splitting it into k number of folds. For each iteration, where the total number of iterations equals k number of folds, one of the folds is chosen as the test data set. The rest is concatenated to a training data set. Each iteration gives an estimated error (MSE). The total error is given as the average of all the iteration errors. Figure 2 illustrates the cross-validation method with 4 folds.

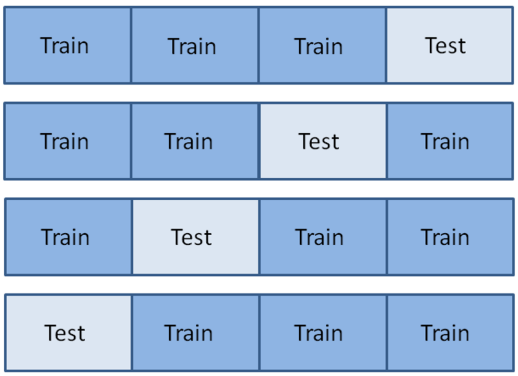

Figure 2: An illustration of the cross-validation method. In this example, each model consists of 4 folds (3 train and 1 test fold). Each model gives an estimated error. The average error is taken of the models.

Franke function

The Franke function is a weighted sum of four exponentials,

$$\begin{aligned} f(x,y) = &\frac{3}{4}\exp\Big(-\frac{(9x-2)^2}{4}-\frac{(9y-2)^2}{4}\Big) \\ &+\frac{3}{4}\exp\Big(-\frac{(9x+1)^2}{49}-\frac{(9y+1)}{10}\Big) \\ &+\frac{1}{2}\exp\Big(-\frac{(9x-7)^2}{4}-\frac{(9y-3)^2}{4}\Big) \\ &-\frac{1}{5}\exp\Big(-(9x-4)^2-(9y-7)^2\Big) \end{aligned} \tag{16}$$

The Franke function is used as a test function for regression methods.

Method

The regression analysis was performed on a dataset generated from the Franke-function (16) and on real terrain data retrieved from the website earthexplorer.usgs.gov. The data generated from the Franke-function was used to verify that the code was working, and that results were as expected. With the verified code, the real terrain data was analysed with the different regression methods and resampling techniques.

Three different regression methods were used: ordinary least squares (OLS), ridge and LASSO. Furthermore, bootstrap and cross-validation were used as resampling methods. In each regression method, a design matrix was made with x and y variables on the form $[x, y, x^2, y^2, xy, ...]$.

The goal was to find parameters β that gave the best fit for the data in terms of the smallest error. The parameters were found based on training data and applied on test data.

To show what the real data looked like, all the data points were used to plot the heatmap of the original terrain data, the OLS fit without resampling and the difference between the two plots. A surface plot and the corresponding fit was also made by using all data points. It was only done once because of the large computation time. Further on, the real data was downsampled in order to decrease computational time.

Generating data with the Franke function

The Franke function was dependent on two independent variables $\boldsymbol{x}$ and $\boldsymbol{y}$. Thus we defined a domain $D = [x] \times [y]$ where $x \in [0, 1]$ and $y \in [0, 1]$. The function returned the calculated function values $\boldsymbol{z}$.

Scaling the data

Before any regression was done, the data was scaled. Since features in the design matrix can have different units and values with different ranges, it is common practice to scale them into dimensionless form. That way, features might not be 'overrated' because the values are large. It also increases numerical stability. An example is that overflow might occur for design matrices with polynomials of high degree if the features are not scaled. In some cases, scaled features are easier to interpret.

The features in X were normalized to a scaled matrix X^*. This made all the columns range from 0 to 1. A scaled column vector $X^*_{i,*}$ is given as

$$X^*_{i,*} = \frac{X_{i,*} - \min X_{i,*}}{\max X_{i,*} - \min X_{i,*}}.$$

The elevation data z was standardized into

$$z^* = \frac{z - \overline{z}}{\sigma_z}.$$

That means the data was centered by subtracting the mean value, and scaled by the standard deviation. This

means that the scaled elevation data had zero mean and unit variance. The design matrix was made without an intercept as the data could be rescaled by the original mean.

Regression with OLS

First, OLS regression was tried to find the best fit. Regressions were done with polynomial degrees varying from 1 to 10. No resampling was used for the these regressions.

For each polynomial degree, the method created a design matrix. The matrix was split into train and test data where the test data constituted 20% of the original data.

The parameters $\beta = \{\beta_0, \beta_1, ..., \beta_{p-1}\}$ where p is the number of parameters, were calculated using the SVD algorithm,

```
def svd_algorithm(X, z):

    U, Sigma, Vt = np.linalg.svd(X,
    ↪  full_matrices=False)
    betas = Vt.transpose() @ np.diag(1/Sigma)
    ↪  @ U.transpose() @ z

    return betas
```

The predicted values $\tilde{\boldsymbol{z}}_{train}$ for the training data was found by performing matrix multiplication on the training data and the betas. Then, the same betas were applied on the test data giving us the predicted values $\tilde{\boldsymbol{z}}_{test}$ for the test data.

Next up, the MSE, R^2 and β values were calculated and studied.

At last, the OLS regression was performed using bootstrap and cross-validation as resampling methods. The bias-variance tradeoff was analysed for the bootstrap method. Then a comparison of the MSE for bootstrap, cross-validation and no resampling for varying polynomial degree was done.

Regression with ridge

The ridge regression method used the hyperparameter λ which constraints the penalty term in equation (13). Again, regressions with varying polynomial degree were done.

For each polynomial degree, the method created a design matrix and split it into train and test data. The parameters β were found using the ridge solver,

```
def ridge_solver(X, z, lmd):

    XTX = X.transpose() @ X
    betas = np.linalg.pinv(XTX +
    ↪  lmd*np.eye(len(XTX))) @ X.transpose()
    ↪  @ z

    return betas
```

Regressions with varying polynomial degree and hyperparameter λ was made to find the best combination of the two. The hyperparameter ranged from 10^{-8} to 10^2. The results were collected in a heatmap of the MSE. This heatmap was made 100 times and a new heatmap of the average was made so to reduce the variance of the results. After chosing the best polynomial degree, regressions with varying hyperparameter were done to study its effect on the MSE. The bias-variance tradeoff was also studied by varying hyperparameter.

Regression with LASSO

LASSO regression was studied in the exact same manner as for the Ridge regression. The parameters were found using the LASSO solver,

```
def LASSO_solver(X, z, lmd):

    RegLASSO = linear_model.LASSO(lmd,
    ↪  fit_intercept=False)
    RegLASSO.fit(X,z)
    betas = RegLASSO.coef_.reshape(-1,1)

    return betas
```

The LASSO solver used a package called scikit-learn to perform the calculations.

Results

Terrain Data

The following calculations were done using all the datapoints (roughly 6.5 million) in the terrain data. The method performed was OLS without resampling up to polynomial degree 10.

Figure 3 shows a surface plot of the terrain data and the corresponding model. The model looks similar to the original data, only with less variations in the terrain/surface.

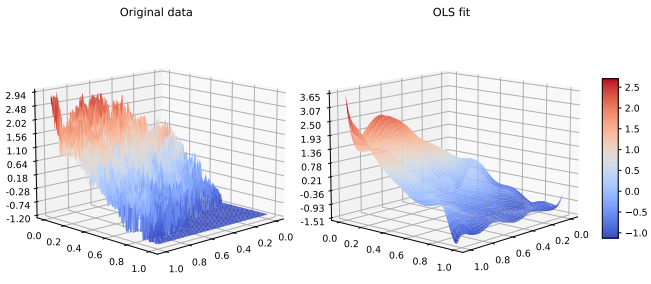

Figure 3: Surface plots for the original terrain data and the OLS fit.

Figure 6, 7, 4 shows three heatmaps: the original data, the OLS fit and the difference between the two. The figure shows that the OLS fit is a smoothed version of the terrain data and does not include abrupt transition in terrain in terms of elevation.

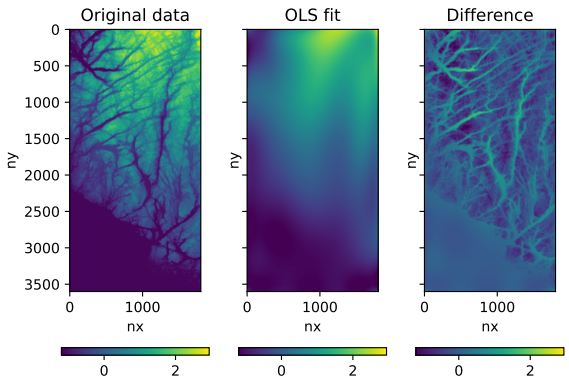

Figure 4: Heatmaps showing the original data, the fit made by the OLS method and the difference between the two plots. The x and y axis represents the number of the data point.

Figure 5 shows the MSE against polynomial degrees for both train and test data. The minimal MSE value was obtained at polynomial degree 10 and was approximately equal to 0.15.

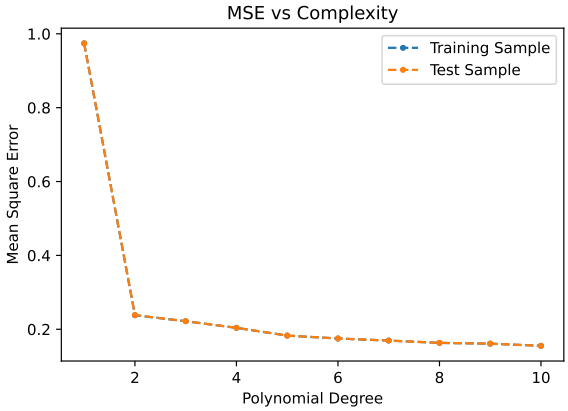

Figure 5: Calculated MSE against polynomial degrees using OLS without resampling for the terrain data. The blue and orange line represents the train and test data, respectively.

Down-sampled Terrain Data

The following calculations were done using a down-sampled dataset of the terrain data.

The train and test sets were split randomly every run, so the scaled data sets' range changed for each run. Typically, they ranged from -1.1 to 2.6.

Figure 8, 9 and 10 were produced from the same code run. They were generated using OLS without resampling. The rest of the figures were produced one by one in different code runs. The specific methods used are defined in the texts for each plot.

Figure 6 shows a surface plot of the terrain data and the corresponding model. The model looks similar to the original data, only with less variations in the terrain/surface.

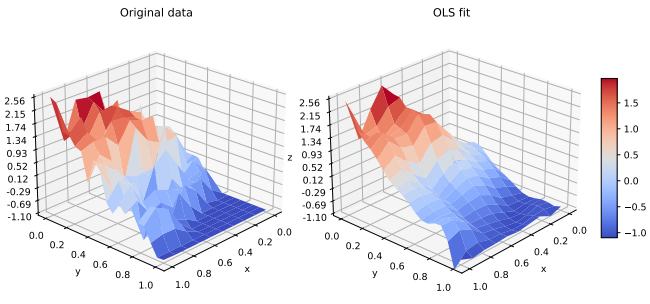

Figure 6: Surface plots for the downsampled terrain data and the OLS fit.

Figure 7 shows three heatmaps: the (downsampled) original data, the OLS fit and the difference between the two. The figure shows that the OLS fit is a smoothed

version of the terrain data and does not include abrupt transition in terrain in terms of elevation.

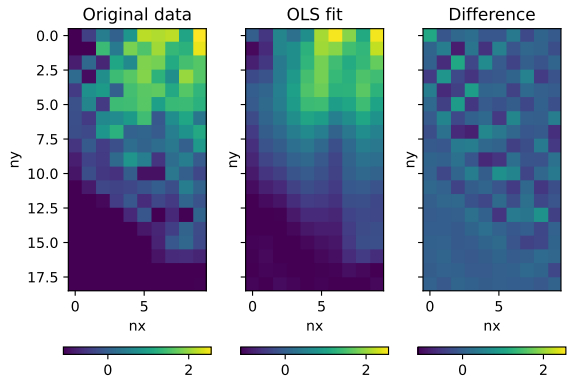

Figure 7: Heatmaps showing the original data, the fit made by the OLS method and the difference between the two plots. The x and y axis represents the number of the data point.

Figure 8 shows the MSE against polynomial degrees for both train and test data. After passing polynomial degree 7, the MSE for the test data starts to rise.

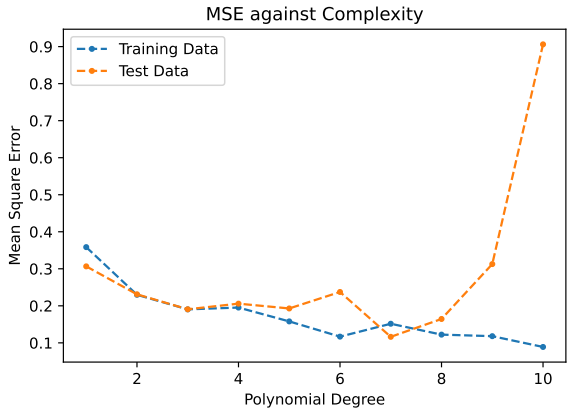

Figure 8: Calculated MSE against polynomial degrees using OLS without resampling for the terrain data. The blue and orange line represents the train and test data, respectively.

Figure 9 shows the R^2 score of complexity for both train and test data. After passing polynomial degree 6, the MSE for the test data starts to fall.

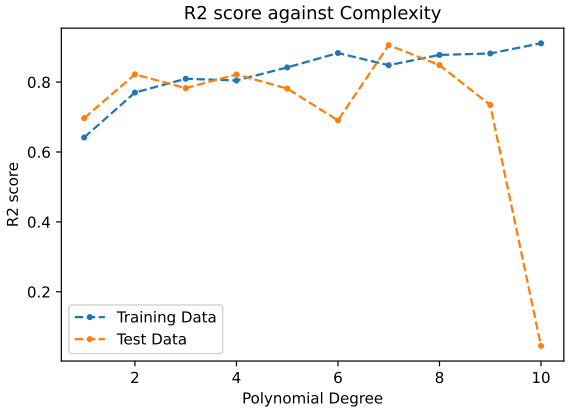

Figure 9: Calculated R^2 score against polynomial degrees using OLS without resampling for the terrain data. The blue and orange line represents the train and test data, respectively.

Figure 10 shows the values of the β parameters against polynomial degree. The figure shows that the variance of the set of parameters increases with model complexity. The regression was run up to polynomial degree five.

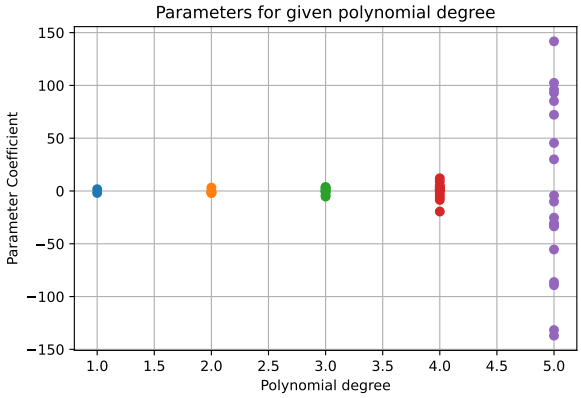

Figure 10: Parameter values against polynomial degree. Regression was performed using OLS with no resampling up to polynomial degree five.

Figure 11 shows a comparison of the MSE against polynomial degree for the different resampling methods including no resampling at all. The MSE was calculated for the test data. OLS regression was used. The lowest MSE was 0.19 and was obtained at polynomial degree 6.

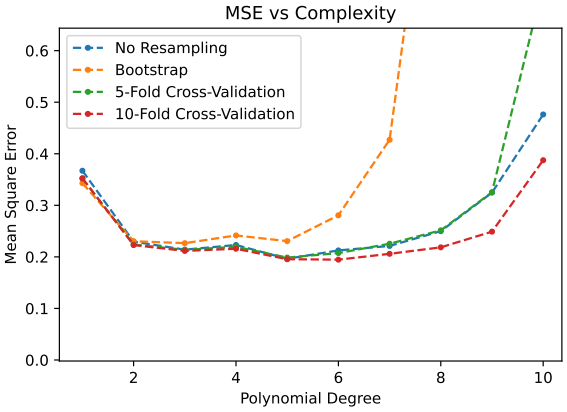

Figure 11: MSE of models with different resampling methods. The MSE was calculated on the test data 100 times and averaged. The figure shows models with no resampling, 100 bootstraps, 5-fold cross-validation and 10-fold cross-validation.

Figure 12 shows a heatmap of the MSE for varying polynomial degree and common logarithm of hyperparameters λ. The plot is an average of 100 heatmaps. The lowest value was -0.727 at $\log\lambda = -6$ and polynomial degree 7.

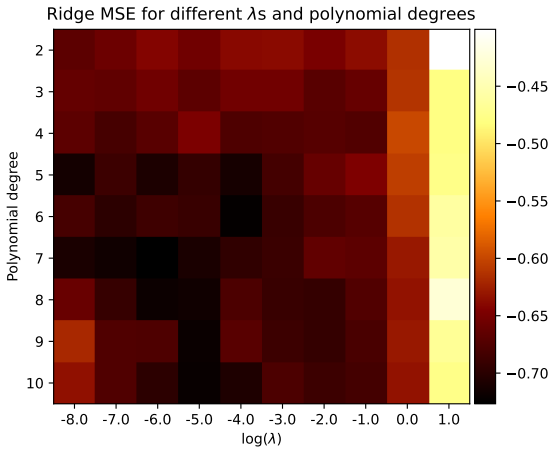

Figure 12: The figure shows a heatmap of the MSE of different Ridge regressions. Regressions were made for different values of the hyperparameter λ and different polynomial degrees. Lowest value = -0.727 at $\log\lambda = -6$ and polydeg=7.

Figure 13 shows a heatmap of the MSE for varying polynomial degree and common logarithm of hyperparameters λ. The plot is an average of 100 heatmaps. The lowest value was -0.693 at $\log\lambda = -3$ and polynomial degree 7.

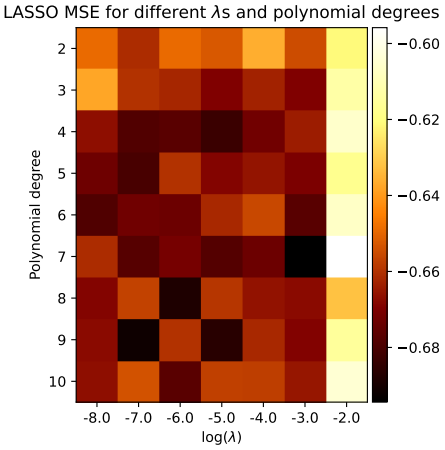

Figure 13: The figure shows a heatmap of the MSE of different LASSO regressions. Regressions were made for different values of the hyperparameter λ and different polynomial degrees. Lowest value = -0.693 at $\log\lambda = -3$ and polydeg=7.

Figure 14 shows a comparison of the MSE against varying hyperparameter λ for different regression methods with bootstrap and 10-fold cross-validation as resampling methods. The polynomial degree was set to 7. The original terrain data was downsampled with $200 \times 200 = 40.000$ skips, meaning that every 40.000 datapoint was skipped.

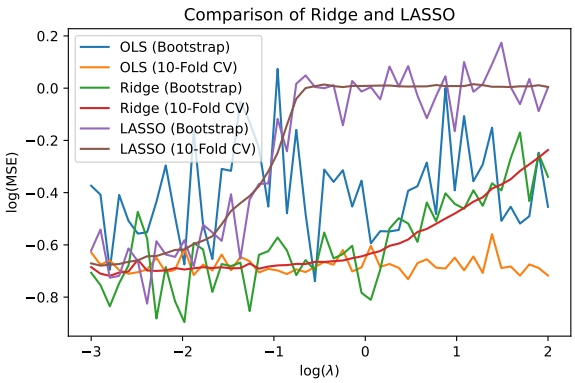

Figure 14: Comparison of the MSE for different regression and resampling methods. Skips 200x200 datapoints.

Figure 15 shows a comparison of the MSE against varying hyperparameter λ for different regression methods with bootstrap and 10-fold cross-validation as resampling methods. The polynomial degree was set to 7. The original terrain data was downsampled with $50 \times 50 = 2.500$ skips, meaning that every 2.500 datapoint was skipped.

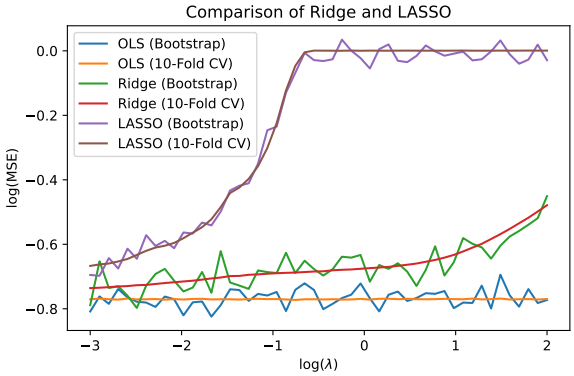

Figure 15: Comparison of the MSE for different regression and resampling methods. Skips 50x50 datapoints.

Figure 16 shows the bias-variance tradeoff with varying polynomial degree using OLS with bootstrap as resampling method. The second axis shows the $\log_{10}$ of the error. The intercept between the bias and the variance occurs somewhere between polynomial degree 7 and 8. The lines are interpolated between the polynomial degrees.

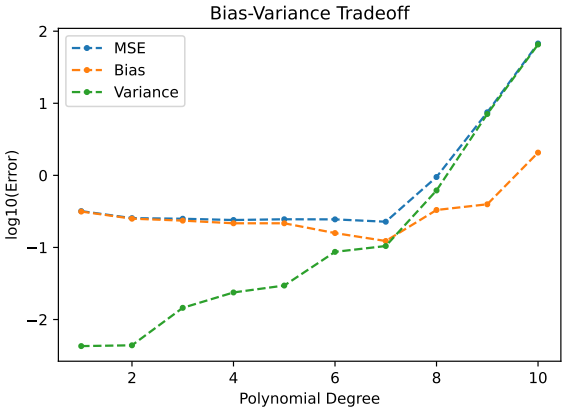

Figure 16: The figure shows the common logarithm of errors against the models polynomial degree. The errors shown are the MSE, bias and variance of the predicted values on the test data.

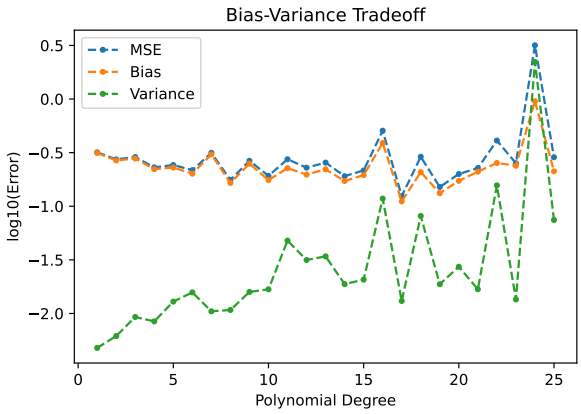

Figure 17: The figure shows the MSE, bias and variance of Ridge regressions with hyperparameter λ=0.01 for varying polynomial degree. The regressions was done with 100 bootstraps. Downsampled data.

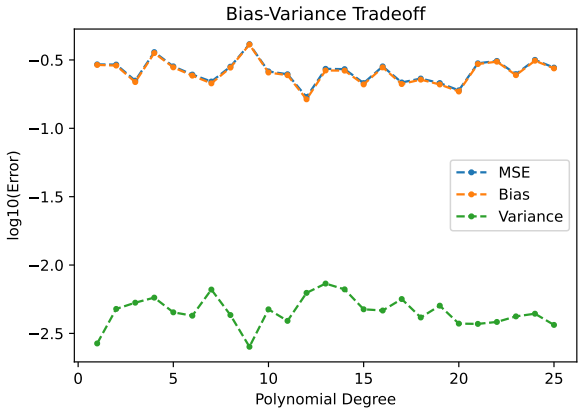

Figure 18: The figure shows the MSE, bias and variance of LASSO regressions with hyperparameter λ=0.01 for varying polynomial degree. The regressions was done with 100 bootstraps. Downsampled data.

Discussion

Amount of Data

The surface plots and heatmaps (fig. 3, 6, 4 and 7) shows that the resolution of the terrain is higher when using all data points. Figure 7 shows a very pixelated version of the terrain. The downsampled dataset does not catch the abrupt variations in elevation in the terrain we are trying to model. In other words, the data we are using to fit the model is less representative for the terrain the less data points we use.

Figure 5 and 8 shows the calculated MSE using all data and a set of downsampled data, respectively. When using all data points, the MSE decreases all the way up

to polynomial degree 10, which was the highest degree used in the calculations. When calculating the MSE using the downsampled data set, the MSE decreases until it reaches polynomial degree 7, and starts increasing significantly after that. This means that the model is most likely overfitted for polynomial degrees higher than 7, thus giving a big error for the test data.

The R^2 score was another way to represent how the good the model was. A score of 1 would be the perfect model. Figure 9 shows that the model gets better and better until it reaches polynomial degree 7 and likely enters a region of overfitting.

Our resampling of the downsampled data set gave a big variance of MSE between runs. This made it hard to interpret the results in figure 14. The figure shows that the MSE given by the bootstrap method was very unstable. When increasing the number of data points, this variance of MSE decreased. The result, given by figure 15, was now easier to interpret.

The reason why we chose to work with the downsampled data was first of all because of the computation time, but also because we wanted to show the consequences of not having enough data.

Computation Time

As observed in figure 5, the MSE kept decreasing for each polynomial degree. A question would be how many polynomial degrees we could use to decrease MSE even further before encountering overfitting. If we could decrease the MSE even more, we would have a better model. The only problem is the computation time. Increasing polynomial degree means increasing number of iterations in the code which demands a longer computation time. If resampling methods were to be used as well, the computation time would increase significantly and the code could take several hours to run on a commercial laptop.

Scaling the Data

The x values in the terrain data was initially ranging from 0 to 3600, giving significantly big values in the design matrix, and potentially leading to overflow error. Therefore both x and y were normalized.

We suspect this happened before we decided to scale the design matrix because the MSE for the training data started to increase after polynomial degree 7 for OLS regression (see figure 8). The MSE for the training data is in theory only supposed to decrease with increasing complexity. After scaling the data, the problem did not occur, and the training MSE kept decreasing towards zero.

The maximum value in $\boldsymbol{z}$ was 1203. When the variables x and y are small, the parameters would need to be big to be able to predict values like 1203. If the parameters gets too big, they can also lead to overflow errors. Therefore the terrain data was standardized.

After scaling, the terrain data typically ranged from -1.1 to 2.6. These are values that could practically never give overflow problems for the parameters $\hat{\beta}$ as they are not big enough.

Resampling Techniques

To improve the precision of the predicted values, resampling techniques were used in all regression methods. Figure 11 shows the MSE for OLS regressions with varying polynomial degrees. The resampling techniques used were bootstrapping with 100 repetitions, 5-fold cross-validation and 10-fold cross-validation. OLS without resampling was also plotted for comparison. Since the variance of the calculated MSEs was so big, each regression was repeated 100 times and averaged for a clearer picture on which resampling method worked best.

In the region of polynomial degrees between 5 and 8, where the regression gave the best MSEs, the 10-fold cross-validation gave the smallest MSE for all degrees. This means that the 10-fold cross-validation was on average the best resampling method to use for our downsampled dataset. The 5-fold cross-validation did on average not improve the precision of the model compared to using no resampling. For some polynomial degrees it did better, and for others it did worse. The regressions with bootstrapping did on average worse than the regressions without resampling. This is likely because of the downsampling of the data. With the downsampling, the design matrix might be so scarce that each bootstrapped matrix becomes so limited that the regression works against it's purpose. This problem could be resolved by including more datapoints in the dataset. Including more data could also improve the precision of the regressions with cross-validation.

Ridge and LASSO Regression

A difference between OLS and the ridge and LASSO methods is that ridge and LASSO utilises the tuning parameter λ. To figure out what λ and polynomial degree that gave the best model in terms of lowest MSE, a heatmap was made. Since the variance of the calculated MSEs was big, each regression was repeated 100 times and averaged to get a more valid result.

Figure 12 shows that the best model was found using polynomial degree 7 and $\lambda = 10^{-6}$. Figure 13 shows that the best model was found using polynomial degree 7 and $\lambda = 10^{-3}$. While it may seem odd that the λ that

gave the smallest MSE for LASSO was 10^3 times bigger than the λ found for the ridge model, these values are so close to zero that their size doesn't really matter in this case. What they actually show is that the ridge and LASSO methods are no better than OLS in this particular case because a λ equal to zero gives the same solution as the OLS regression. The different best λ values found for ridge and LASSO would change from computation to computation because of the variance of the model.

Figure 14 shows that OLS gave the best fit over all in terms of smallest MSE. The ridge and LASSO methods gave better results with decreasing λ values. The figure also shows that LASSO gave the worst fit. The reason why LASSO was so bad is probably because the penalty on the parameters was too high. Another important notice is that figure 15 used polynomial degree 7. OLS did not give overfitting until the polynomial degree was bigger than 7.

So why bother with ridge and LASSO at all? One option we did no research is if there exists a polynomial degree (higher than what we studied) and a λ value where ridge or LASSO actually makes a better model than OLS. One thing we know for sure is that ridge and LASSO can be very useful when dealing with multicollinearity between parameters and when overfitting is a problem because they shrink the parameters closer to each other and closer to zero.

Bias-Variance Tradeoff

Figure 16 shows the tradeoff between the bias and variance of OLS regression fits as the polynomial degree increases. As the models' complexity increases, the bias goes down and variance goes up. The bias and variance intercept at polynomial degree 7. This signals the point where the models goes into the region of overfitting. From the same graph, we see that the MSE of the models keeps increasing with polynomial degrees, underlining the claim of the modeling going into the overfitting region.

Figure 17 shows us the tradeoff between the bias and variance of ridge regression fits as the polynomial degree increases. The hyperparameter λ is set to 0.01. This time, the intercept between the bias and the variance occur first for a fit with polynomial degree 24. We know that the ridge parameters are scaled OLS parameters by a factor of $1/(1+\lambda)$ from equation (14). This dampening factor keeps the variance of the model down and the bias up. This explains why the intercept between the bias and variance is delayed to a higher polynomial degree than for the OLS fit.

Figure 18 shows us the tradeoff between the bias and variance of ridge regression fits as the polynomial degree increases. The hyperparameter λ is set to 0.01. This time, the intercept between the bias and the variance doesn't occur for regressions with polynomial degrees under 26. The LASSO is an even more biased regression method than ridge. As LASSO can set parameters equal to zero, it keeps the variance of the models down. This explains why the intercept between the bias and the variance doesn't appear on the graph. It most likely occurs for models with a much higher polynomial degree.

Conclusion

From all fitted models with polynomial degree up to 10, the ordinary least squares regression of 7th order polynomial with 10-fold cross-validation returned the lowest mean squared error on the downsampled dataset. As we did not preform Ridge and LASSO regressions for all polynomial degrees before they went into the region of overfitting, due to computation time, we cannot rule out that a better fit might exist for the same dataset. We know that the Ridge and LASSO regression did not go into the overfitting region since the bias-variance diagrams told us so. The best resampling technique for the 7th order OLS fit was the 10-fold cross-validation. The resampling techniques are sensitive to scarce data, so a new comparison would be needed if the data was not to be downsampled as much. To further study the data, less downsampling is suggested. As is a new comparison of the regression methods with higher polynomial degrees. Note that this might increase computation time significantly.

Bibliography

Hastie, Tibshirani and Friedman (2017). *The Elements of Statistical Learning*. Springer.

Hui, Jonathan (2022). *Machine Learning — Singular Value Decomposition (SVD) Principal Component Analysis (PCA)*. URL: https://jonathan-hui.medium.com/machine-learning-singular-value-decomposition-svd-principal-component-%20analysis-pca-1d45e885e491 (visited on 11th Oct. 2022).

Team, Great Learning (2022). *What is Ridge Regression?* URL: https://www.mygreatlearning.com/blog/what-is-ridge-regression/ (visited on 11th Oct. 2022).

Appendix

Expectation values and variances in linear regression

With our model $\tilde{z}$ as an approximation for the function f, the expectation value $E[z]$ of the data becomes

$$\begin{aligned} E[z_i] &= E[\tilde{z}_i + \epsilon_i] \\ &= E[\tilde{z}_i] + E[\epsilon_i]. \end{aligned}$$

The noise ϵ is Gaussian distributed, so the expectation value is zero. With $E[\epsilon_i] = 0$ we write

$$\begin{aligned} E[z_i] &= E[\tilde{z}_i] + 0 \\ &= E\big[\sum_j X_{ij}\beta_j\big] \\ &= \sum_j X_{ij}\beta_j \\ &= \tilde{z}_i. \end{aligned}$$

The variance of the data becomes

$$\begin{aligned} var(z_i) &= E[(z_i - E[y_i])^2] \\ &= E[(\tilde{z}_i + \epsilon_i - \tilde{z}_i)^2] \\ &= E[\epsilon_i^2] \\ &= \sigma^2. \end{aligned}$$

It is now evident that the variance of the data is the same as the variance σ^2 of the random noise ϵ.

To calculate the expectation value of the optimal parameters $\hat{\beta}$, let us first look at the following reformulation of the them using equation (?) for OLS regression.

$$\begin{aligned} \hat{\beta} &= (X^T X)^{-1} X^T y \\ &= (X^T X)^{-1} X^T (\tilde{z} + \epsilon) \\ &= (X^T X)^{-1} X^T (X\beta + \epsilon) \\ &= (X^T X)^{-1} X^T X\beta + (X^T X)^{-1} X^T \epsilon \\ &= \beta + (X^T X)^{-1} X^T \epsilon \end{aligned}$$

When we use the last expression for the optimal parameters when calculating the expectation value, it becomes clear that

$$\begin{aligned} E[\hat{\beta}] &= E[\beta + (X^T X)^{-1} X^T \epsilon] \\ &= E[\beta] + E[(X^T X)^{-1} X^T \epsilon] \\ &= \beta + 0 \\ &= \beta. \end{aligned}$$

We calculate the variance of the optimal parameters $var(\hat{\beta})$ as

$$\begin{aligned}
var(\hat{\beta}) &= E[(\hat{\beta} - E[\hat{\beta}])^2] \\
&= E[(\hat{\beta} - E[\hat{\beta}])(\hat{\beta} - E[\hat{\beta}])^T] \\
&= E[(\hat{\beta} - \beta)(\hat{\beta} - \beta)^T] \\
&= E[((X^TX)^{-1}X^T\epsilon)((X^TX)^{-1}X^T\epsilon)^T] \\
&= E[(X^TX)^{-1}X^T\epsilon\epsilon^TX(X^TX)^{-1}] \\
&= (X^TX)^{-1}X^TE[\epsilon\epsilon^T]X(X^TX)^{-1} \\
&= (X^TX)^{-1}X^T(\sigma^2I)X(X^TX)^{-1} \\
&= \sigma^2(X^TX)^{-1}X^TX(X^TX)^{-1} \\
&= \sigma^2(X^TX)^{-1}
\end{aligned}$$

Bias-Variance

The mean squared error $E[(z - \tilde{z})^2]$ can be decomposed into the model's squared bias $(E[\tilde{z}] - f)^2$, variance $E[(\tilde{z} - E[\tilde{z}])^2]$, and the variance σ^2 of the noise.

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