FYS-STK4155 - Applied data analysis and machine learning
Project 1 - Regression analysis and resampling methods

Mäntysalo A. Visa, Nyberg Fredrik and Zariouh Osama

October 2019

The material for project 1 can all be found at: `https://github.com/OsamaZa/Project1`

Abstract

This project is a study on various regression methods such as the Ordinary Least Squares, ridge regression and lasso regression. These regression methods will be combined with k-fold cross-validation to resample the data, and the Bias-Variance trade off will also be studied in detail. The algorithms will first be tested on a dataset generated by a two-dimensional Franke's function before being applied on digital terrain data provided by the U.S. Geological Survey [2].

Introduction

Regression analysis is tremendously useful to gain knowledge about complicated and seemingly chaotic systems. In regression analysis one extracts and exploits statistical relationships between variables to create mathematical models to describe a system. This reduces the complexity of the problem and reasonable predictions can then be made. However there are many techniques for gaining these models.

In this project, algorithms are created for Ordinary Least Squares (OLS), ridge and lasso regression methods with python. Only the lasso algorithm will be based on a pre-made SciKitLearn-functionality, the OLS- and Ridge-algorithms are self-made. K-fold cross validation is implemented as a resampling technique to mimic novel data, when used in combination with the regression methods for training and testing the models.

These algorithms will firstly attempt to fit a fifth order polynomial approximation to datasets created by the two-dimensional Franke's function with an added noise term. Later the algorithms will be applied to real terrain data provided by the U.S. Geological Survey [2]. Finally, an evaluation and discussion will be had for which model best fits the data.

The different regression methods will be evaluated and discussed by interpreting various statistical quantities such as the Mean Square Error (MSE) and variance score (R2-score) for the different parameters. The Bias-Variance trade off will also be studied for this discussion.

Theory

K-fold cross validation

When it comes to making reliable predictions with regression analysis it is common practice in the discipline of machine learning to resample the available data into training and test data. When the data is split in this manner we artificially create a working environment where some of the data is used to train the model, and the remaining data is used to evaluate how well it predicts. The resampling technique used in this project is the k-fold cross validation (CV).

The k-fold CV algorithm randomly rearranges the set of data and then splits the data into k folds. Then the data from (k-1) of the folds are selected to train the model e.g. OLS, and the remaining fold is then used to benchmark the prediction performance of the model. This is repeated until all possible combinations of training and test data of grouped (k-1) and k folds have been used for training. The initial randomisation ensures that each group represents data from the whole span of data. This way of building a model is done in order to obtain the most reliable model which consistently can make decent predictions for novel data, based on the available data.

Ordinary Least Squares regression

The Ordinary Least Squares method, OLS, is a common regression technique and lays the foundation of this analysis. The method is all about minimizing the averaged squared error between the data, (x_i, y_i), and a general linear regression model. The linear regression model, [1] p.44, has the form:

$$f(X) = \beta_0 + \sum_j (X_j \beta_j) \tag{1}$$

Here β_j represents the regression parameters, X is a design matrix which contains information about the dependent variables in the model that is used for fitting e.g. polynomials up to 5th order.

The average sum of squared errors between the model, $f(x_i)$, and the data, y_i, is then given by the so-called cost function

$$C(\beta) = \frac{1}{n}\sum_i (y_i - f(x_i))^2 \tag{2}$$

$$C(\beta) = \frac{1}{n}\sum_i (y_i - \beta_0 - \sum_j (x_{ij}\beta_j))^2 \tag{3}$$

$$C(\beta) = (y - X\beta)^T(y - X\beta) \tag{4}$$

This leads to a minimization problem where the minima of $C(\beta)$ will yield the optimal values for the parameters β in the model we wish to describe the data with:

$$\frac{dC(\beta)}{d\beta} = -2X^T(y - X\beta) \tag{5}$$

$$\downarrow$$

$$X^T(y - X\beta) = 0 \tag{6}$$

$$\downarrow$$

$$\beta = (X^T X)^{-1} X^T y \tag{7}$$

This approach for fitting is rather pragmatic but effective, it makes no assumptions about the validity of the model and takes no stance on how the data actually arose in the first place. It simply yields best possible fit to the given conditions we set in the design matrix.

A typical problem, when solving the inversion operation in the calculation for beta, can occur when the matrix $X^T X$ is singular or near singular. One can not simply invert a singular matrix, and this challenge also applies for near singular matrices because these can computationally be interpreted as singular. A method for dealing with this is by doing a Singular Value Decomposition (SVD). However in the forthcoming section about ridge, an alternative way to bypass this issue will be presented.

Ridge regression

OLS regression doesn't differentiate between the important and unimportant predictors in a model. It includes all of them as equally important. This may lead to overfitting of the training data and failing in finding a good model for future data. In ridge regression this is avoided by introducing a ridge penalty parameter, lambda, to the standard OLS method. The parameters β_{ridge} are obtained by minimizing the cost function L, [3] p.16:

$$L_{ridge}(\beta;\lambda) = ||Y - X\beta||_2^2 + \lambda||\beta||_2^2 \tag{8}$$

$$\downarrow$$

$$L_{ridge}(\beta;\lambda) = \sum_i (Y_i - X_{i*}\beta_j^2) + \lambda \sum j\beta_j^2 \tag{9}$$

From SVD we know that $X = UEV^T$ where E is a matrix containing the eigenvalues in a diagonal descending order. The estimator β with SVD implemented is $\beta = (X^T X + \lambda I)^{-1} X^T y = UD(D^2 + \lambda I)^{-1} DU^T y$. The matrix D is the squared diagonal matrix. In the brackets the λ controls how much the eigenvalues squared, i.e. parameters β, are shrunken.

This shrinking punishes the parameters β after their usefulness for the model based on the terms defined in X. Useful terms in X tend to have higher values for the beta-parameter than less useful terms. As λ increases, the relatively unimportant estimators converge towards zero and the important estimators still remain large enough to play a crucial role in the model. Therefore, the λ has to be chosen carefully such that the estimators β_i are shrunk sufficiently, and that the estimator β potentially gets closer to the actual true estimator when attempting to predict novel data. Therefore by shrinking the irrelevant estimators β_i, more bias is introduced to the model in return for a potentially better MSE. An easy way to implement this λ is to simply add it to the diagonal of the matrix X^TX.

$$X^TX \rightarrow X^TX + \lambda I \tag{10}$$

As mentioned in the OLS section, singular or near singular matrices can be a challenge when implementing a standard OLS method. If the columns in matrix X^TX are linearly dependent, then the matrix X^TX is singular. This will make the matrix rank to be deficient and make it impossible to invert the matrix X^TX. A byproduct of ridge is that it fixes this problem by making the X^TX matrix invertible. The estimator $\beta = (X^TX)^{-1}X^Ty$ is defined if $(X^TX)^{-1}$ exists. An easy way to go around this is to add a small diagonal component to the X^TX matrix. This makes the X^TX a full rank matrix or a non-singular matrix.

Lasso regression

Lasso regression is an alternative method for punishing less useful parameters in the estimator. The ridge and lasso methods are quite similar but have some important differences. The loss function, L_{Lasso} takes the l_1 norm instead of the Euclidean norm found in ridge and minimizes to find the estimators, [3] p.79.

$$L_{Lasso}(\beta;\lambda) = ||Y - X\beta||_2^2 + \lambda||\beta|| \tag{11}$$

$$\downarrow$$

$$L_{Lasso}(\beta;\lambda) = \sum_i (Y_i - X_{i*}\beta_j)^2 + \lambda\sum_j |\beta_j| \tag{12}$$

This results in a different shrinking effect than the one found in ridge. In ridge regression the irrelevant parameters in the estimator reduce the slope asymptotically to 0, while lasso can reduce these same parameters all the way to zero. Which can potentially exclude irrelevant variables more effectively.

Lasso regression method is implemented via the SciKitLearn functionality, which uses a Stochastic Gradient Descent (SGD) algorithm. The method uses the gradient of the cost function to converge to local minimum. Stochastic means that there are multiple random initial points where gradient direction is calculated, so convergence is reached faster. Learning rate γ can be defined by looking at an approximate function of how GD is used on the estimators β.

$$\beta^{i+1} = \beta^i + \gamma\frac{dC}{d\beta} \tag{13}$$

The size of γ dictates how the β converges. If γ is too large, the algorithm might not convergence toward the true result. However if the γ is too small the algorithm will need more iterations to achieve convergence and therefore use more computational time.

Bias-variance trade off

The cost function can be written as

$$E[(y - \tilde{y})^2] \tag{14}$$

where $y = f + \epsilon$ and $\tilde{y}$ is our fitted function and $E[x] = \frac{1}{n}\sum_i(x_i)$ if there is n x_i's. In the derivation under $E[\epsilon] = 0$ because ϵ is normally distributed.

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

$$\begin{aligned}
&= E\big[(f-E[\hat{f}])^2\big]+E[\varepsilon^2]+E\big[(E[\hat{f}]-\hat{f})^2\big]+2E\big[(f-E[\hat{f}])\varepsilon\big] \\
&\quad + 2E\big[\varepsilon(E[\hat{f}]-\hat{f})\big] + 2E\big[(E[\hat{f}]-\hat{f})(f-E[\hat{f}])\big]
\end{aligned}$$

$$\begin{aligned}
&= E\big[(f-E[\hat{f}])^2\big]+E[\varepsilon^2]+E\big[(E[\hat{f}]-\hat{f})^2\big]+2(f-E[\hat{f}])E[\varepsilon] \\
&\quad + 2E[\varepsilon]E\big[E[\hat{f}]-\hat{f}\big] + 2E\big[E[\hat{f}]-\hat{f}\big](f-E[\hat{f}])
\end{aligned}$$

$$\begin{aligned}
&= E\big[(f-E[\hat{f}])^2\big]+E[\varepsilon^2]+E\big[(E[\hat{f}]-\hat{f})^2\big] \\
&= \frac{1}{n}\sum_i(f_i-E[\tilde{y}])^2+\frac{1}{n}\sum_i \tilde{y}_i-E[\tilde{y}])^2+\sigma^2
\end{aligned} \tag{15}$$

In the derivation above $E[\epsilon^2] = Var[\epsilon] + (E[\epsilon])^2 = \sigma^2$

The term $\frac{1}{n}\sum_i(f_i - E[\tilde{y}])^2$ is the mean of the bias squared and the term $\frac{1}{n}\sum_i(\tilde{y}_i - E[\tilde{y}])^2$ is the mean variance. The bias is the difference between the mean of our modeled function and the true function. The variance is the difference between the mean of our modeled function and our modeled function.

The training error is $E[(y_t raining - \tilde{y})^2]$ where $y_t raining$ is the datapoints from the training set.

Statistical tools

Mean Squared Error

The Mean Squared Error (MSE) is a method of quantifying how well a model predicts the data. It is a measure of the mean squared deviance between the predicted values of the model $f(x_i)$ and the actual measurement y_i.

$$MSE(x_i) = \frac{1}{n}\sum_i [(y_i - f(x_i)^2)] \tag{16}$$

This is useful when evaluating comparing different models and evaluating which best fits the test data for each specific case of groupings of test data and training data.

R2-score

The variance-score, or the R2-score, is a method of quantifying the mean spread of a random variable from its expected value.

$$Var(y) = \frac{1}{n}\sum_i [(y_i - E(y))^2] \tag{17}$$

Confidence interval

Confidence interval (CV) is an estimate on the certainty of a parameter. The standard practice of reporting CV amounts to approximate of 95%. It is going to be used in this project to evaluate the estimators β certainty. From [1] p.49, the CV of β can be found with the formula:

$$Min(\hat{\beta} - z^{1-\alpha} v_j^{\frac{1}{2}} \hat{\sigma}), Max(\hat{\beta} + z^{1-\alpha} v_j^{\frac{1}{2}} \hat{\sigma}) \tag{18}$$

where $z = 1.96$ is the percentile of the normal distribution which make $\alpha = 0.05$ and v_j is the jth diagonal element of $(X^T X)^{-1}$.

Method

The two different datasets used in this project was artificial data generated by the Franke's function and real data imported from the US Geological Survey. The Franke's function is given by the following expression

$$\begin{aligned} f(x,y) &= \frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) \\ &+ \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) \\ &+ \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) \\ &- \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right) \end{aligned} \tag{19}$$

and is defined for x,y $\in [0,1]$.

Noise was added with the error term ϵ $N(0,\sigma^2)$ such that $z = f(x,y) + \epsilon$. The z values were computed with a grid of x and y values ranging from 0 to 1 with steps of 0.05 giving a total of 400 data points. This function was used to generate the first dataset where the methods and algorithms were established and tested.

The real data can be imported from *U.S. Geological Survey* [2]. The chosen real data in this project consists of a 3601x1801 grid with topographical points of Oslofjorden, Norway. The heat map visualization for the topography was implemented with imshow from matplotlib.

Regression and statistical analysis

A five-fold CV resampling algorithm was implemented before each fitting. The arrays of data were shuffled by creating a randomly shuffled index vector divided into 5 parts. The shuffled index vector was then used on the design matrix and the datapoints in a loop to yield five different combinations of training and test data. A regression was done for each of the training sets and evaluated on the corresponding test set. This was done for each of the three regression methods done here, and for both the datasets.

For the fitting algorithm for the OLS regression method, standard linear algebra operations following eq.7 were implemented with the numpy-functionalities. For ridge regression, the algorithm for OLS was slightly altered by simply adding $\lambda * I$-matrix as shown in eq.10.

However, SciKitLearn-functionality was used for doing the lasso-regression.

The design matrix, X, had the same polynomial degree when attempting to fit both of the datasets. X, was defined to consist of polynomials in x and y up to fifth order.

In order to find the optimal λ for ridge and lasso, a tuning algorithm was implemented. This was done by running the ridge and lasso methods for a list of several lambdas. The lambda-list for artificial data points defined as logspace(-7,-5,15) for ridge and logspace(-8,-6,5) for lasso. The lambda-list for real data points defined as logspace(-3,-1.5,5) for ridge and logspace(-6,-2,5) for lasso. The reason for shorter length of lambda-vector for the lasso calculations was due to reduction in computation time. The lower and upper value for both intervals was chosen by attempting several values, until the interval was sufficiently produced a distinct minima for the mean MSE for each of the two regression methods. The fitting with the λ which produced the MSE minima was chosen to represent the method in the final comparison between OLS, ridge and lasso.

The R2-score and MSE were computed for each of the fittings, in order to find the best regression method for each data set. Each regression method had five fittings as a consequence of the five-fold CV. This yields five calculated values for the R2-score and MSE. To give an overall score for the predictability of a method the mean of the five R2-scores and MSE's for each experiment were used to compare the methods. The method with the lowest mean MSE's was then concluded to be the best regression method for the dataset.

The confidence interval was computed for the real data. For ridge and lasso, the mean betas from the five fittings for the optimal λ were used. The betas confidence interval and values were plotted in figures.

Bias-Variance trade off

The bias, variance and error were studied as a function of varying complexity for the design matrix for the real data. The design matrices X_n was defined for a maximum polynomial order ranging from 0th to n-th order for OLS and 2nd to n-th order for ridge and lasso.

Figures were constructed to illustrate how the mean bias, variance and MSE for the five folds changed with complexity. This was only done for the artificial data. The highest complexity for X was chosen to be 12th order. How these figures changed for different lambdas were also studied for the ridge and lasso method.

The bias, variance and error were studied as a function of the amount of data points used too for the real data. For this the design matrix was fixed to a fifth degree polynomial as usual. The error and the training error were also plotted in their own graph as a function of complexity for the real data.

Results

The main results for the regression and statistical analysis can be found in table 1 and 2. The confidence intervals for β in addition to all figures created for this project can be found in the GitHub link in the beginning of the document. However, relevant figures for our discussions will be brought up here when needed.

Franke's function	OLS	Ridge	Lasso
R2-score	0.946	0.969	0.965
MSE	0.00431	0.00247	0.00283

Table 1: MSE and R2-score for data made with Franke's function where $\lambda_{Ridge} = 3.73 * 10^{-6}$ and $\lambda_{Lasso} = 3.16 * 10^{-8}$.

Real data	OLS	Ridge	Lasso
R2-score	0.6486	0.6487	0.6366
MSE	6023.1	6023.1	6231.1

Table 2: MSE and R2-score for real data where $\lambda_{Ridge} = 2.37 * 10^{-3}$ and $\lambda_{Lasso} = 1.00 * 10^{-3}$.

From the results for the mean MSE and R2 with artificial data, the ridge regression outperformed OLS and showed ever so slightly better performance than lasso for predicting the test data. For the real data ridge barely outperformed OLS, but this time OLS outperformed lasso. It is however important to mention that maximum iterations for lasso was set to 100000 for the artificial and 1000 the real data. This led to non convergence of the estimators to save computational time.

It is important to be aware of that the compatibility between X and the dataset can pre-determine whether ridge or lasso is favoured. The choice of X is limited to the observers knowledge of the system. If the observer defines more useful terms overall in X, ridge can be more favoured because the lasso penalty is more strict on the relatively less useful ones. However, if X contains useless terms with respect to the data, these terms might be eliminated with lasso which can yield a better fitting than ridge, because these useless terms only converge to zero but are not necessarily eliminated. And if all the terms are more or less equally important, OLS or ridge will probably be the best fitting. Since ridge had the best fit for both datasets, the design matrix probably had some less important terms in the design matrix in both cases. However, as mentioned, the lasso method did not converge, so it is hard to say if it would have ended at a better approximation if more computational time was given.

The bias can be interpreted as the error because of our model. If the bias is high, our model is too simple, so it does not care enough about the training data. Therefore the bias is expected to increase as the polynomial degree of the fitting function gets smaller. The variance on the other hand, can be interpreted as the error resulting from the scatter of the training data. If the variance is high our model is too complicated, it cares too much about the training data and is expected to increase as the polynomial degree of the fitted function gets higher.

From figure 1 it is seen that the test error at first gets smaller when increasing the complexity, but then bigger again as the complexity increases for OLS. This is because as the complexity increases, the polynomial will be able to fit the training data better and better as also shown in figure 1. This will predict the test data better and better until the complexity is so high that it accounts for the random fluctuations in the training data. At this point the polynomial no longer gives a curve that describes the general trend. The same trend does not occur for the ridge and lasso method as shown in figure 3 and 5 respectively. This is probably because ridge and lasso shrink unimportant terms. Ridge and lasso makes the bias higher, in order to maybe fit future data better.

As seen from figure 2, 4 and 6, the error is not bigger than the bias plus the variance as expected for any of the methods. For OLS the variance follows the right trend, but the bias should be more like the training error in figure 1. The variance also crosses the error, which is very weird. For ridge and lasso, both bias and variance does not follow the expected trend. It is very hard to tell why, since the derivations shows that the error should be strictly bigger than or equal to the bias plus the variance. Note! The bias term used in the calculations might not be correct.

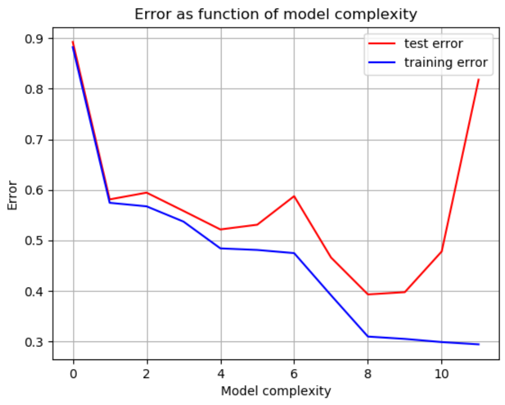

Figure 1: Error vs Complexity for OLS

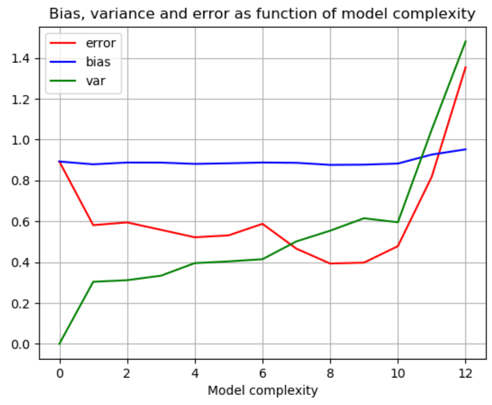

Figure 2: Bias, variance and error for OLS

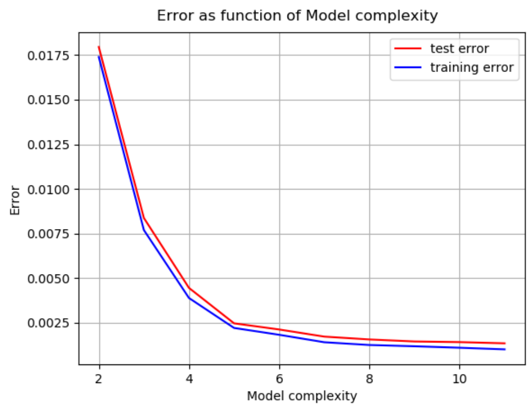

Figure 3: Error vs Complexity for ridge

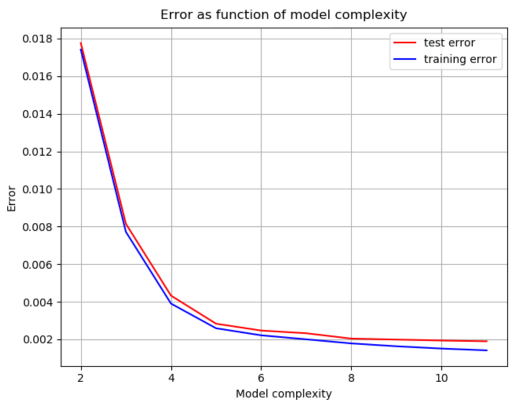

Figure 5: Error vs Complexity for lasso

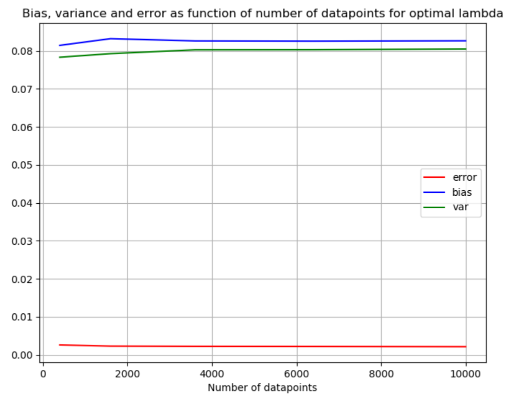

Figure 4: Bias, variance and error for ridge

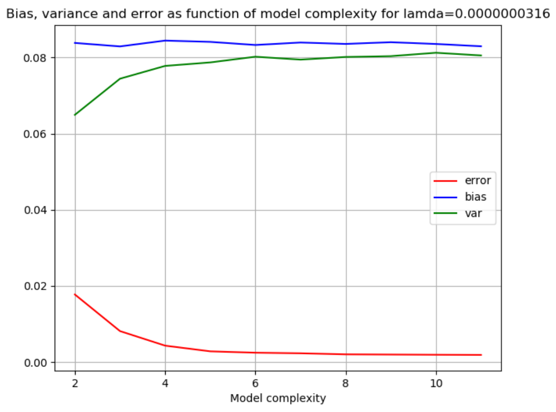

Figure 6: Bias, variance and error for lasso

By analysing the estimators β, from figure 1-3, for each of the regression method the importance of the shrinking is visualised. In OLS the estimators have a larger value then both ridge and lasso regression. The difference between OLS and ridge looks to be only a difference in magnitude of estimators. While lasso penalises the unnecessary estimators much more and changes the β both in magnitude and how the individually β_i are in relation to each other.

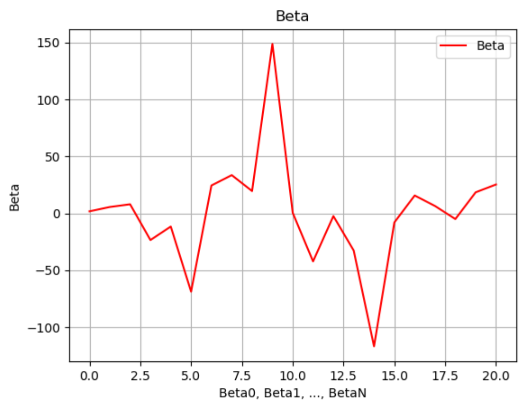

Figure 7: Betas of the OLS

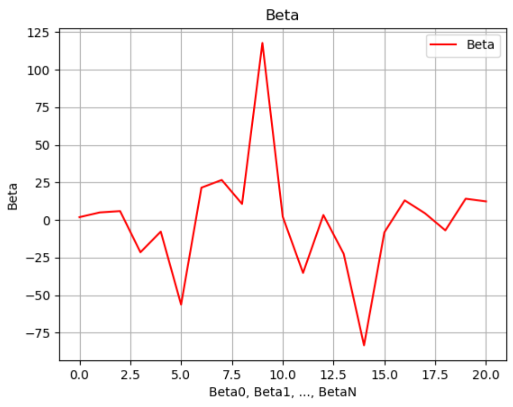

Figure 8: Betas of the ridge

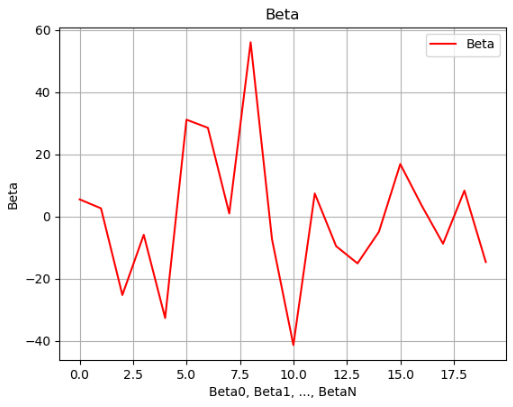

Figure 9: Betas of the lasso

The confidence interval of the estimator for each regression method are presented and with the CV the certainty of each β_i can be estimated. This have to do with the importance of the estimators β_i. However this can in combination with the figures for β determine reliability of each β. E.g. if an estimator is small and the confidence interval is large this estimator can be an important one. Which means that the estimator is not well defined. The confidence interval plots are listen in the appendix.

In figure 10 the training and test MSE and variance for all the regression methods was plotted against the number of datapoints used. This was done for a fixed X complexity of 5th order and with a noise $\in$[0,1]. The fluctuations of these quantities tends to get smaller with increasing numbers of data. This plot was done for number of datapoints = (400, 1600, 3600, 6400, 10000, 14400, 19600, 25600, 32400, 40000, 48400, 57600). This can probably indicate that after a certain number of points the model tends to become stable. It could have been interesting to investigate this for even more datapoints to see if this actually converges. This kind of plot could probably be exploited to determine the least amount of points needed, to potentially save computation time, for a given dataset to have a converged and stable regression model.

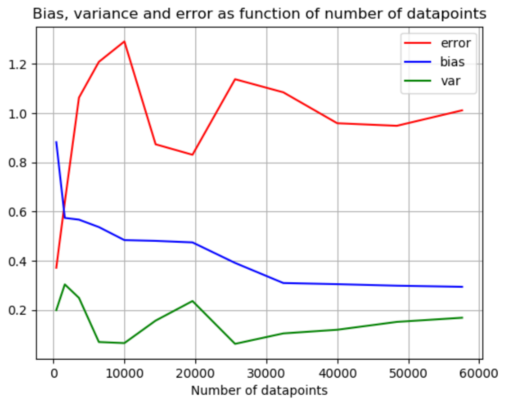

Figure 10: Bias, variance and error plotted against number of datapoints for OLS

Conclusion

So for these datasets the ridge method turned out to be the best regression based on the mean squared error and the R2-score. The OLS method makes bad predictions when the complexity is too large because it fits the training data too good. The ridge and lasso methods is not so affected by the complexity of the design matrix since it is able to neglect unimportant terms. From the confidence intervals, the betas calculated from the ridge and lasso method seems more certain than the betas from the OLS.

Appendix

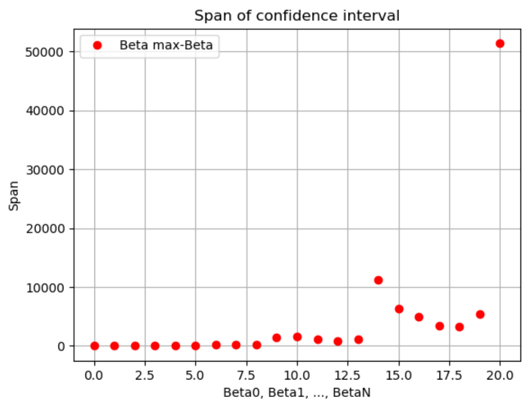

Figure 11: Span of confidence interval for OLS

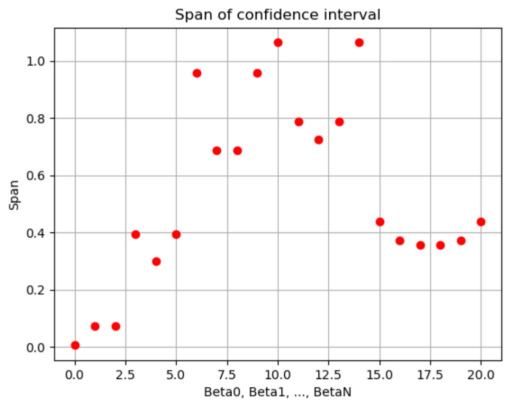

Figure 12: Span of confidence interval for ridge

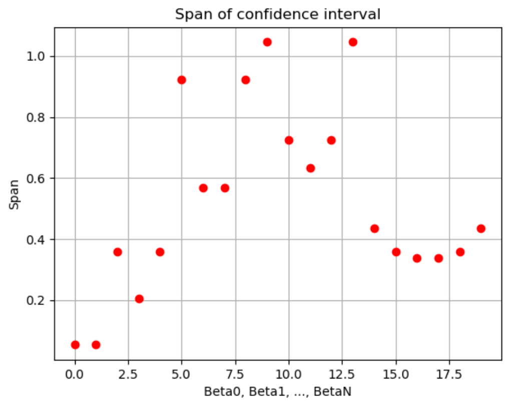

Figure 13: Span of confidence interval for lasso

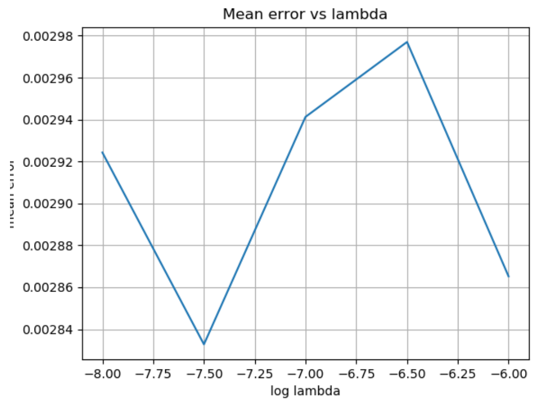

Figure 14: Plots of optimal lambda for ridge Franke function

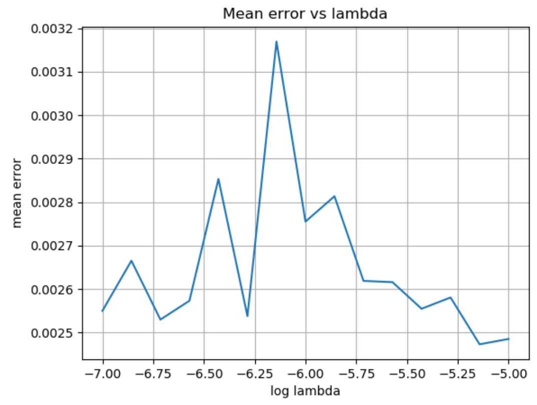

Figure 15: Plots of optimal lambda for lasso Franke function

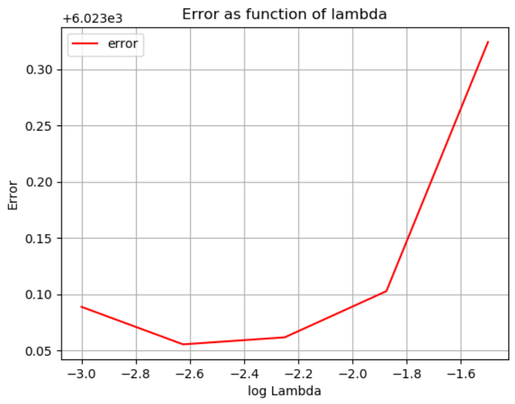

Figure 16: Plots of optimal lambda for ridge real data

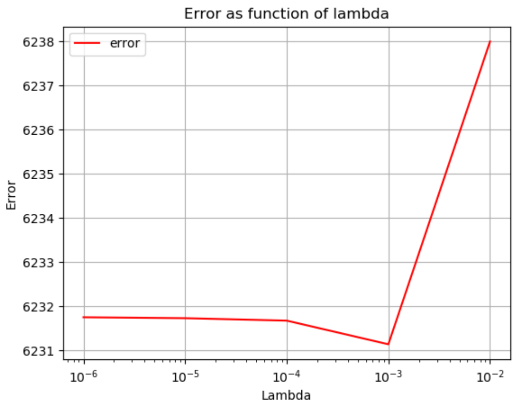

Figure 17: Plots of optimal lambda for lasso real data

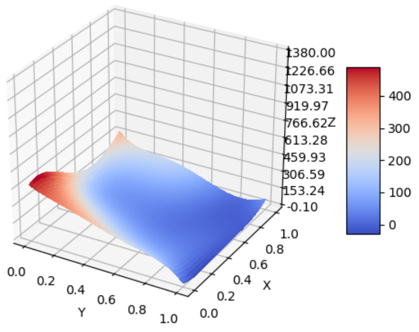

Figure 18: Plots of approximated terrain of Oslofjord OLS.

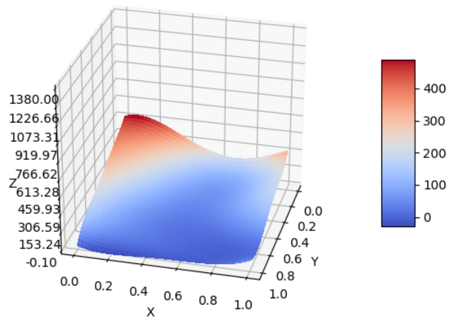

Figure 19: Plots of approximated terrain of Oslofjord ridge.

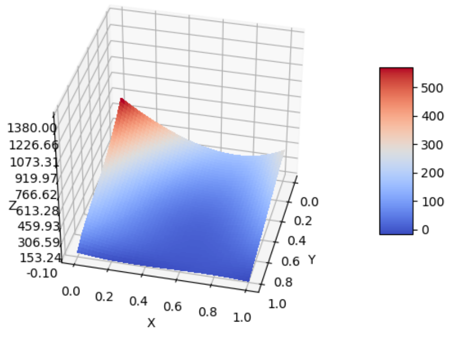

Figure 20: Plots of approximated terrain of Oslofjord lasso.

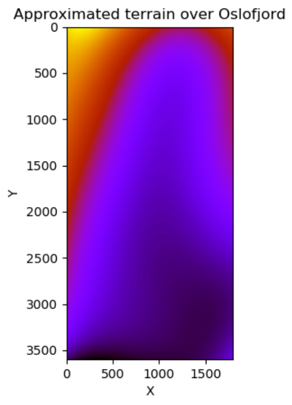

Figure 21: Approximated surface of Oslofjord OLS.

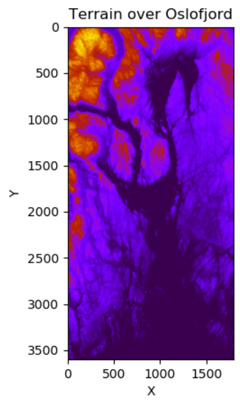

Figure 24: Surface of Oslofjord.

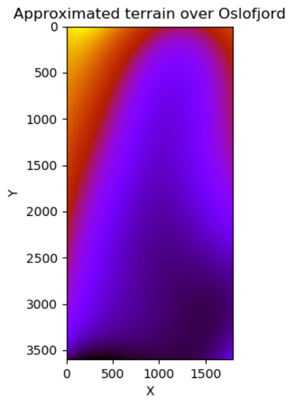

Figure 22: Approximated surface of Oslofjord ridge.

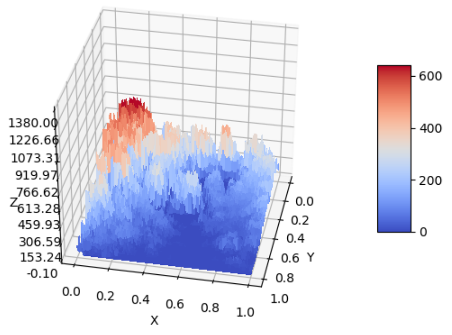

Figure 25: Surfaceplot of Oslofjord.

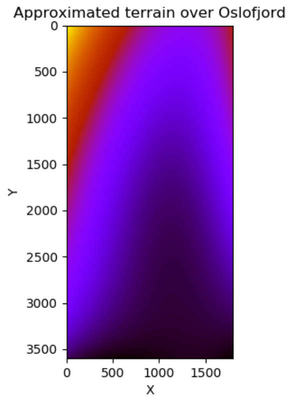

Figure 23: Approximated surface of Oslofjord lasso.

References

[1] Jerome Friedman Trevor Hastie Robert Tibshirani. *The Elements of Statistical Learning.* Springer-Verlag New York, 2009. ISBN: 978-0-387-84857-0.

[2] *U.S. Geological Survey.* URL: `https://earthexplorer.usgs.gov/`.

[3] Wessel N. van Wieringen. "Lecture notes on ridge regression". In: *arXiv® - Cornell University* (2019).