Applications of Regression Methods on Terrain-data

Christer Dreierstad, Hans Erlend Bakken Glad, Torbjørn Lode Gjerberg,* and Stig-Nicolai Foyn†
Institute of Physics, University of Oslo
(Dated: October 7, 2019)

Applying methods of regression in combination with resampling we studied the effects of increasing the complexity of our fitted model. The methods are applied on the Franke function with normal distributed datapoints and an image of a terrain. When increasing the complexity we observe overfitting for Ordinary Least Squares (OLS) for polynomial degree > 7. Further we apply Ridge and Lasso regression, which are methods that penalize the complexity proportionally the hyperparameter λ, yielding a more stable and/or smooth model for larger values of λ when the complexity is high or there is a high variance in the data points. For Ridge we observe overfitting for the same polynomial degree as OLS, where the overfit is related to the hyperparameter. For Lasso there is no observed overfitting. Applying our methods on the terrain data we observe that the R^2 score is greatly reduced compared to the Franke function, which is related to how noisy the terrain data is. For the terrain data the best MSE was found to be 0.008 using Ridge regression.

I. INTRODUCTION

By regression analysis with resampling we will predict a model for the Franke function applying normal distributed datapoints, this will serve as a benchmark for our methods. Having a model that behaves as one would expect for the case of the Franke function, we then apply our methods on a real dataset. Our real data will consist of random terrain data in the form of a grey-scale image, where the pixel value is related to the curvature of the terrain. The aim of this paper to make a polynomial fit or parametrization of our datasets. The datasets are two dimensional, which requires us to modify the design matrix such that a given polynomial degree is expressed for x, y and the combination of x and y.

For regression analysis we will consider the following methods; OLS, Ridge and Lasso, while our method for resampling is Cross-Validation. We will study the MSE as we increase the complexity, i.e. increase the polynomial degree of the fit, specifically we will compare the MSE when we apply our model on the train and test data. This leads to a discussion of the bias-variance tradeoff and overfitting of the model. Both these subjects will be a central discussion in this paper. We will study how the MSE depends on the hyperparameters in Ridge and Lasso, which penalize the complexity of the fitted model. Further the confidence intervals of the coefficients β will be discussed.

* Institute of Informatics, University of Oslo
† Institute of Geoscience, Univerity of Oslo

II. DATA

A. Data Generated using the Franke Function

We initially generate artificial data using the Franke function;

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

so that the height z of the terrain is

$$\boldsymbol{z} = f(\boldsymbol{x},\boldsymbol{y}) + \boldsymbol{\epsilon}, \quad \boldsymbol{x},\boldsymbol{y},\boldsymbol{z} \in \mathbb{R}^n$$

where $\boldsymbol{\epsilon}$ represents normal distributed stochastic noise, i.e. $\epsilon_i = \mathcal{N}(0,\sigma^2)$. This gives us a set of data that resembles terrain which we can use to test our regression methods on.

B. Real Terrain Data

After testing our algorithms on artificial data we will introduce real terrain data. This data is downloaded from [1] and represents an area in Norway. This dataset is 3601 by 1801 pixels, with each pixel taking a value that represents the terrain height.

Due to the size of the dataset we will need to downscale the data in order to get reasonable runtimes with our algorithms. We choose to scale down the data by a factor of 20 along both dimensions, resulting in a dataset that is 400 times smaller than the complete dataset. This appears to give a good balance between runtime and data resolution. The individual datapoints are also scaled so that x, y and z only take values between 0 and 1.

III. METHOD

A. Ordinary Least Squares

1. The Design Matrix

Approximating an unknown data set with regression requires that one has a set of independent explanatory variables. This is organised as a set of parameters in a design matrix. This way an unknown dataset can be approximated with a linear combination of the design matrix X and some vector with coefficients β, scaling our parameters. The goal is that the dataset can be represented, within acceptable accuracy, using this linear combination assuming there is some irreducible noise ϵ inherent to the dataset. The data can then be represented as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \epsilon. \tag{1}$$

To approximate the Franke function, and later the terrain dataset, we used a design matrix containing combinations of polynomials. The complexity of the model was specified by the maximum polynomial degree in the matrix. In this study we want to fit a model that predicts curvature of a function or terrain dataset at a given position in the xy-plane. The data points constructing the polynomials will be then be positional coordinates in the x, y-plane. In the final part of the paper when we analyze terrain data we will attempt to find a polynomial parameterization of the data.

2. The Least Squares Approximation

Given a design matrix with parameters, we would like to find the optimal β to scale our model, so that it best fits the dataset. Given that the matrix consists of polynomials it can be written in terms of the parameters x_i, which multiplied with the coefficients β forms our model

$$\tilde{\boldsymbol{y}} = \begin{bmatrix} 1 & x_0^1 & \dots & x_0^{n-1} \\ 1 & x_1^1 & \dots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m-1}^1 & \dots & x_{m-1}^{n-1} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{m-1} \end{bmatrix},$$

for the one dimensional case. Expanding to two dimensions all polynomial degree combinations of x and y must be present in the design matrix, that is $x^i y^j$ $i, j \in [0, n-1]$, where $n-1$ is the maximum polynomial degree. To find β we minimize the cost function $C(\beta)$, which in this case is the MSE. The MSE is given by the average of the sum over the errors in our model

$$C(\beta) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2.$$

The above equation can be written in terms of the model we introduced in Eq. (1), with this we can derive the cost function and find the minimum of the function;

$$C(\beta) = \frac{1}{n}\sum_{i=0}^{n-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}).$$

The minimum is found where the derivative of the function is equal to zero;

$$\begin{aligned} \frac{\partial C(\beta)}{\partial \beta_j} &= \frac{1}{n}\sum_{i=0}^{n-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \\ &= -\frac{2}{n}\left[\sum_{i=0}^{n-1} x_{ij}(y_i - \beta_0 x_{i0} \cdots - \beta_{p-1}x_{ip-1})^2\right] \\ &= -\frac{2}{n}\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) = 0. \end{aligned}$$

The equation for β is now given by:

$$\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{2}$$

The last step to find the β is to invert the matrix $\boldsymbol{X}^T\boldsymbol{X}$. For the case when $\boldsymbol{X}^T\boldsymbol{X}$ is close to singular there may be stability issues when inverting the matrix. In our case we tackle the problem where the matrix is close to singular by Moore-Penrose semi-inversion, which utilizes the singular value decomposition when inverting the matrix. We then end up with the following equation for the predicted model:

$$\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}.$$

With the above equation a fit can be made using polynomial combinations of the positional parameters to benchmark our model on the Franke function. Later OLS is used to fit a model with the same types of parameters to the terrain data to attempt to make a parameterization of the terrain.

3. Confidence Intervals

It is natural when we design a model to be interested in how well the model fits the real data. In particular it is relevant to review which parts of the model that are most important and which parts that are less relevant. To perform such an analysis we need to find the confidence interval of the β values that we found in Eq. (2). The confidence interval of a normal distribution can be found by[1]:

$$\left(\bar{X} - z_{\alpha/2}\frac{\sigma}{\sqrt{n}}, \bar{X} + z_{\alpha/2}\frac{\sigma}{\sqrt{n}}\right),$$

[1] See [2] page 387 Eq. (8.5)

where $\bar{X}$ in our case is β and $\frac{\sigma}{\sqrt{n}}$ is the sample standard deviation of β, s_β. If the confidence interval of some model has a small span, one can be confident within some probability (e.g. 95%) that the model is close to the real solution. The variance of our coefficients can be found by reviewing the variance in $\boldsymbol{\beta}$, given by

$$\begin{aligned} Var(\hat{\boldsymbol{\beta}}) &= Var((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}(\boldsymbol{\beta}+\epsilon)) \\ &= \mathbb{E}[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}(\boldsymbol{\beta}+\epsilon)^T \\ &\times (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}(\boldsymbol{\beta}+\epsilon)], \end{aligned}$$

when using equations (1) and (2). The terms containing β will cancel, leaving

$$\begin{aligned} &= \mathbb{E}[((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\epsilon)^T(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}\epsilon)] \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}[\epsilon\epsilon^T]\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\sigma^2. \end{aligned}$$

With the above equation we can write the confidence interval for our model as:

$$(\boldsymbol{\beta} - \sqrt{(\boldsymbol{X}^T\boldsymbol{X})^{-1}\sigma^2}, \boldsymbol{\beta} + \sqrt{(\boldsymbol{X}^T\boldsymbol{X})^{-1}\sigma^2}).$$

B. Ridge and Lasso Regression Methods

Ridge and Lasso are both classified as shrinkage methods. They minimize the residual sum of squares with a penalty term that adds an explicit size constraint on the parameters[2]. The same ideas that are utilized in Ridge regression can be found in the "ad hoc" way of solving singularities in matrices in linear algebra. The idea is to add some hyperparameter λ to avoid zeros along the diagonal of the matrix, thus being able to give an estimate to the inverted singular matrix. Using the hyperparameter results in an estimate that can be considered reasonable for the determinant of the matrix. Ridge uses a hyperparameter λ to penalize the β coefficients of our model. This is done by adding the λ to the diagonal of the matrix that we are inverting. In some cases where the noise of the dataset is high, Ridge can help reduce overfitting.

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{3}$$

Ridge penalizes the coefficients of the fitted model by uniformly shrinking them, while Lasso translates each coefficient by λ, truncating at zero[3]. In the case of Ridge the coefficients go to 0 as $\lambda \to \infty$. For Lasso we used scikit-learn, since we were more interested in the methods results rather than the method itself.

[2] [3], 62-63

[3] See The Elements of Statistical Learning[3] page 69

C. K-Fold Cross-Validation

When working with scarce data one can use a resampling method for validating the prediction. By splitting the data into subsets; a test and a training set, we performs K validations of the predictive model. For K-fold CV we split the data into K folds, where one of the folds serve as the test set and the rest is used to train the model. This is repeated K times until all the folds have been used as a test set once. Each repetition results in a slightly different model where the evaluation scores are retained, e.g. MSE or R^2 score, which can be used to decide on the best model.

A typical choice of folds is 5 or 10. A lower number of folds K will return a smaller variance, but a larger bias as less data is used for training. With very small datasets this bias is likely to affect the outcome[4]. This bias and variance is inherent to the cross-validation itself for a given number of parameters.

An added bonus of performing a Cross-Validation is that we can scale down the dataset for the terrain data, thus reducing the dataset, to reduce calculations in our algorithm and improve run times, while still producing a satisfactory model.

D. Estimating Bias and Variance

The full prediction error consists of a variance, bias and an irreducible error. When few parameters are used

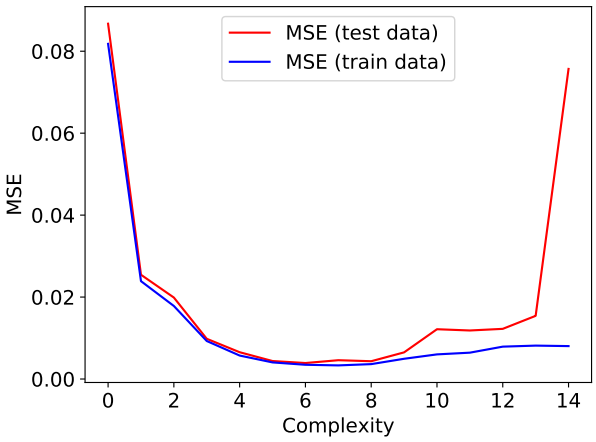


FIG. 1: Train and test error as a function of complexity using the Franke dataset. Method used to predict is OLS. The grid is 50x50 large and the error is normal distributed around 0 with a scale of 0.5, this is done to enhance the bias-variance tradeoff.

[4] See The Elements of Statistical Learning[3] page 243

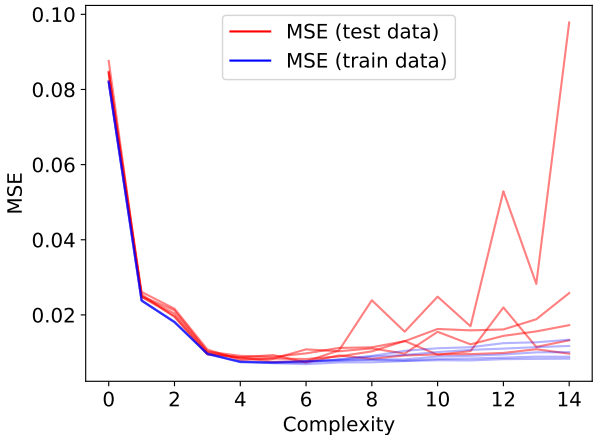


FIG. 2: Train and test error as a function of complexity using Ridge regression on the Franke dataset. The noise is normally distributed around 0 and has a variance of 0.65. The error is tweaked slightly from Fig. 1 in order to illustrate the dependence on λ better. $\lambda = \{1.00\cdot10^{-10}, 3.16\cdot10^{-9}, 1.00\cdot10^{-7}, 3.16\cdot10^{-6}, 1.00\cdot10^{-4}\}$, in order of decreasing test MSE at complexity 14.

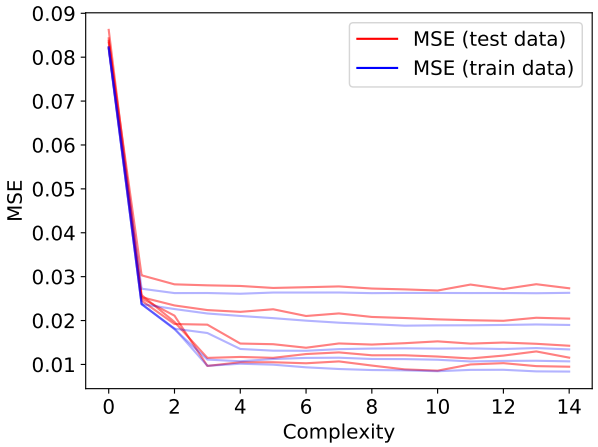


FIG. 3: Train and test error as a function of complexity using Lasso regression on the Franke dataset. The noise is normally distributed around 0 and has a variance of 0.65. The error is tweaked slightly from Fig. 1 in order to illustrate the dependence on λ better. $\lambda = \{5.00\cdot10^{-5}, 1.88\cdot10^{-4}, 7.07\cdot10^{-4}, 2.66\cdot10^{-3}, 1.00\cdot10^{-2}\}$ in order of decreasing test MSE at complexity 14.

in a regression the bias becomes high. If complexity is increased by adding more parameters, i.e. increasing polynomial degree of the fit, the bias will decrease. On the other hand low complexity means there will be a lower variance and increasing complexity will increase the variance. This is the bias and variance of a model which can be derived from the cost function:

$$C(\boldsymbol{X},\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i) = \mathbb{E}[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2],$$

where $y = f(x) + \epsilon$. To find expressions for the bias, variance and irreducible error we add and subtract $\mathbb{E}$ and insert the expression for y

$$\begin{aligned}
&= \mathbb{E}[(f + \epsilon - \tilde{y} + \mathbb{E}[\tilde{y}] - \mathbb{E}[\tilde{y}])^2] \\
&= \mathbb{E}[(f - \mathbb{E}[\tilde{y}])^2] + \mathbb{E}[\epsilon] + \mathbb{E}[(\mathbb{E}[\tilde{y}] - \tilde{y})^2] \\
&= \frac{1}{n}\sum_i (f_i - \mathbb{E}[\tilde{y}])^2 + \frac{1}{n}\sum_i (\mathbb{E}[\tilde{y}] - \tilde{y})^2 + \sigma^2.
\end{aligned}$$

The first term in the above equation is the square bias of the model, and it is the difference between the true function behind the data and the expectation of the predicted model. The second term term is the variance and it tells us how much the given model varies around the mean. A model with high variance follows the training data closely and will not adapt well to new data. The final term is the irreducible error.

IV. RESULTS

In Fig. 1, Fig. 2 and Fig. 3 we can see the MSE for the train and test data applied on our model as a function of complexity, for the Franke dataset, for OLS, Ridge and Lasso respectively. While for the terrain dataset we can see the MSE for train and test applied on the model in Fig. 6.

Fig. 5 shows the confidence intervals for β up to polynomial degree 5.

In table II and III we can see the best polynomial degree and corresponding hyperparameter for the models for Franke and the terrain data respectively.

V. DISCUSSION

We observe a higher R^2 score and lower MSE for the Franke data with a standard deviation > 1. Comparing the Franke function with the terrain data we would expect the methods to perform better on the former, due to the smoothness of the data. Increasing the noise of the

TABLE I: R^2 and MSE for both datasets. For Franke we have used 50x50 grid and a normal Gaussian distributed noise with standard deviation 0.5. The model is trained with noisy data, but the error is found by the true Franke function.

Dataset	R^2 Score	MSE
Franke	0.942	0.004
Terrain	0.539	0.012

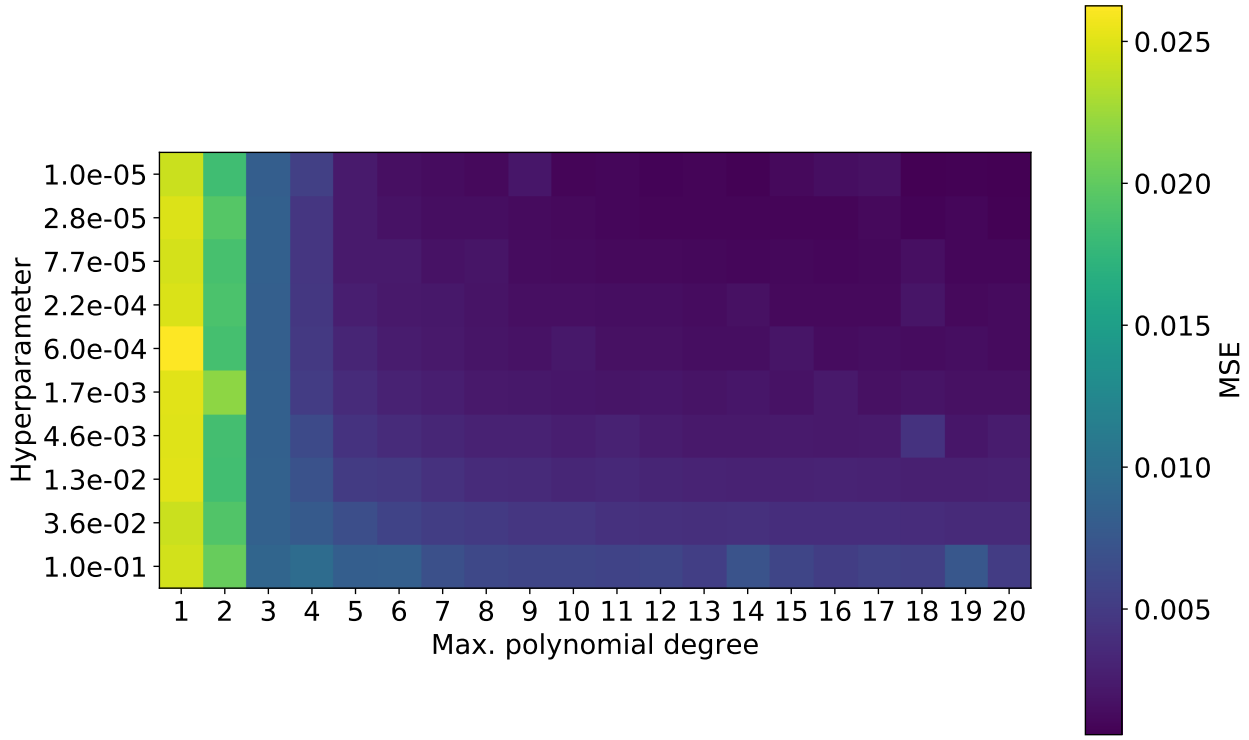


FIG. 4: Test MSE from using Ridge regression to fit a model on dataset generated by the Franke function.

Franke function increases the MSE and reduces the R^2 score, mimicking the noise in the terrain data. Choosing a flatter terrain to model could increase the R^2 score, as when reducing the noise of the Franke function.

A. Confidence Interval of Regression Coefficients

The 95% confidence intervals (CI) for $\boldsymbol{\beta}$ were calculated for the case where the highest polynomial degree was 5 in x and y. The intervals are represented in Fig. 5. We see that the lowest order polynomials have confidence intervals with a narrow range of values, suggesting that we are close to the true values of these coefficients. As the polynomial degree increases, we see that the CI widens, until around degree 15 where the CI starts narrowing again. The coefficients for the highest order polynomials appear to be closer to their true values than the polynomials around order 10, but not as close as the lowest order polynomials.

B. Comparison of Regression Methods

The Franke function is often used to test methods in interpolation problems, and it can help us review the performance of our methods. When reviewing the applicability of the Franke function as a way to benchmark the regression methods, we observed that the methods were not equally applicable in parameterizing the terrain data. Because the noise was uniformly distributed the methods predicted the true Franke function quite easily even for complexities like polynomial degree five. If by contrast we used polynomial degree five on the terrain

TABLE II: Best polynomial degree and hyperparameter for Franke dataset for each model. We have used 50x50 grid and a normal Gaussian distributed noise with standard deviation 0.5.

Dataset	OLS	Ridge	Lasso
Degree	6	7	10
Hyperparameter	N/A	$1.274 \cdot 10^{-4}$	$5 \cdot 10^{-5}$
MSE	0.005	0.004	0.007
R^2	0.935	0.934	0.902

TABLE III: Best polynomial degree and hyperparameter for terrain dataset for each model.

	OLS	Ridge	Lasso
Degree	38	47	45
Hyperparameter	N/A	$4.642 \cdot 10^{-11}$	$3 \cdot 10^{-6}$
MSE	0.009	0.008	0.012
R^2	0.689	0.700	0.548

data, we would not get a good estimate for any of the features in the terrain.

Ridge outperformed OLS for high model complexities (e.g. degree 12 for Franke or degree 40-60 for terrain) where Ridge had a score of $R^2 = 0.952$ and OLS had $R^2 = 0.924$ (with 100x100 grid of datapoints) on the Franke dataset, this is due to how Ridge penalizes the overfitting, by the hyperparameter. Lasso performs worst on the Franke dataset, this is due to some parameters being truncated to zero. Lasso may perform better for higher complexities since it is less likely to overfit than both OLS and Ridge. For the complexities evaluated in the study Lasso did not perform better than OLS or Ridge.

None of the methods were able to predict the features of the terrain data as seen in Fig. 7. All regression methods returned a smooth average over the area with a clear loss of features. The loss of features are because we are fitting the model using polynomials, which will yield smooth functions. The terrain data have a lot of differences in height between points. When using our models to predict the data an average is produced and a lot of information about height is lost. If the models took all this variability into account they would grossly overfit. New predicted points will simply be distributed around the mean within variance. None of the tested regression method preformed particularly well on the the terrain data. Ridge preformed best with an $R^2 = 0.70$. OLS and Lasso were both outperformed by ridge (see table III).

C. Bias-Variance Tradeoff

As we increase the complexity we risk that the model gets biased towards the dataset it has been trained on. In Fig. 1 we see how the MSE for the test data increases relative to the increase in complexity, this corresponds to when the model gets biased and overfitting occurs. We would expect the MSE for the training data to decrease to zero as the complexity increases, but as we see in Fig. 1 the MSE increases along with the test data. The increase of training MSE is because we train the model with the Franke function with added stochastic noise, while applying the true data on model.

Increasing the polynomial degree will reduce the variance, as the model can fit closer to each datapoint. For a certain polynymial degree the model becomes biased to the data, and we observe overfitting, which results in a poor model.

Applying the Ridge regression method to the same data we see a similar trend. Even for different penalty terms λ the test MSE is at its lowest around polynomial degree 4-6, as seen in Fig. 2. We see that Ridge is less likely to overfit and we observe that as the penalty increases, the MSE increases. There is an overfitting for the largest λ, but we consider this is a statistical outlier, as it performs worse than OLS. We would not expect Ridge to perform worse than OLS, as with the penalty term $\lambda = 0$ Ridge is equal to OLS.

Lasso regression on the same data yields MSEs for both test and training data that decreases with complexity. In Fig. 3 the choice of λ has a larger impact on the MSE than the complexity. For Lasso we observe no overfit for the chosen interval, this is due to the method shrinking some of the coefficients to zero. Even for complexity much larger than for both OLS and Ridge regression the MSE becomes smaller.

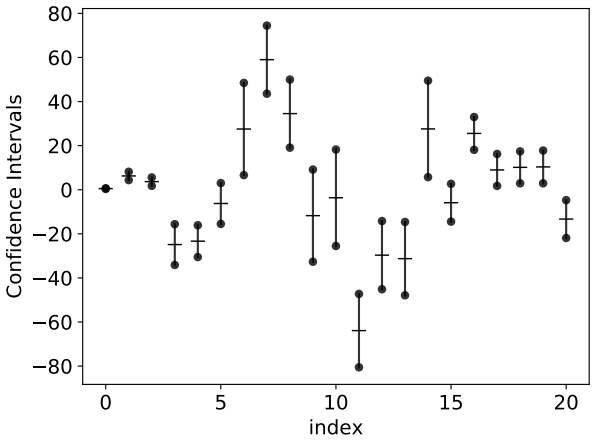


FIG. 5: 95% Confidence intervals for $\boldsymbol{\beta}$ with complexity $p = 5$, using a dataset generated by the Franke function. The error is normally distributed around 0 with variance 0.5.

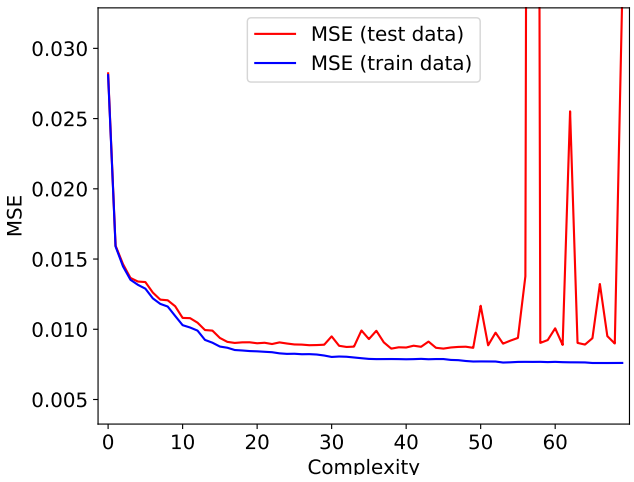
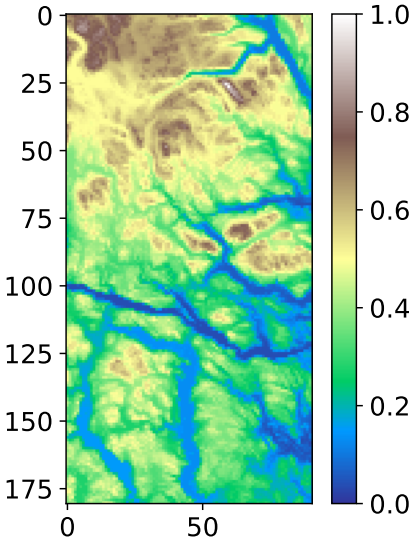
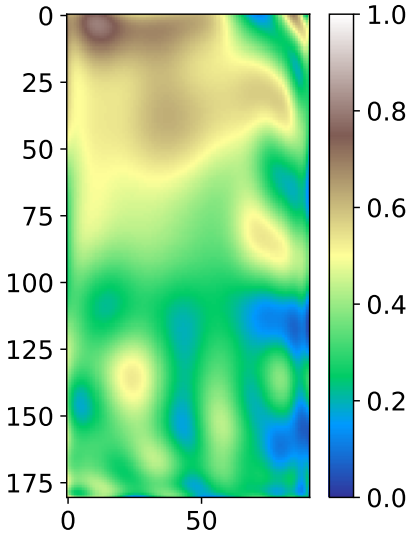


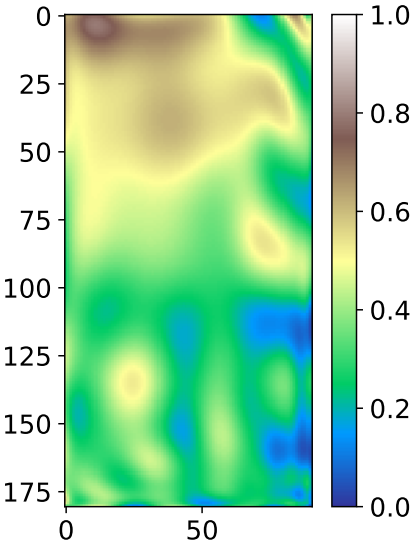
FIG. 6: Train and test error as a function of complexity, using the ordinary least squares regression method on (scaled) terrain data.



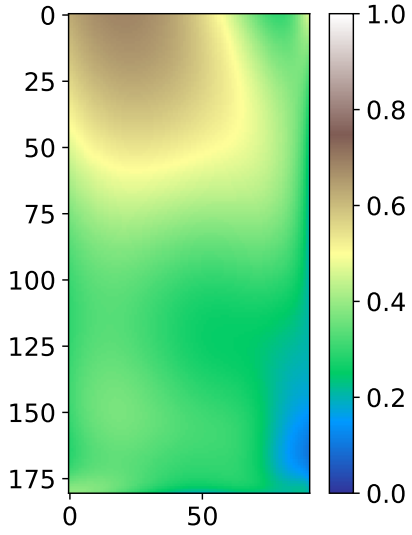
(a) Original terrain.



(b) Terrain as predicted using OLS. Highest polynomial degree in x and y is 38.



(c) Terrain as predicted using Ridge. Highest polynomial degree in x and y is 47 and the hyperparameter $\lambda = 4.642 \cdot 10^{-11}$.



(d) Terrain as predicted using Lasso. Highest polynomial degree in x and y is 45 and hyperparameter $\lambda = 5 \cdot 10^{-6}$.

FIG. 7: Terrain data visualized with colors representing terrain height.

On the terrain data OLS regression gives a similar result to the Franke function data. Fig. 6 shows that train error becomes a quite smooth slope. The test MSE becomes jagged and unstable for high complexity and is at risk of overfitting.

D. Results and Applicability of Regression Models

When it comes to the Franke function our methods all made models with reasonable scores in relation to the true function. This changed if we split the dataset into train and test data, and used the test data to verify our model. We interpreted this as reasonable since the noise

was so large in scale compared to the rest of the dataset. When we switched to the terraindata we soon realized that our methods were not really that good at representing the terrain we wanted to parameterize. Since the "noise" in the terrain could in many cases be significant features of interest like areas with water, we needed to increase the complexity of our model by a lot. This meant in turn that the model would be bad at predicting unknown data, however for the most part we were more interested in replicating that one spesific terrain. Even when increasing the complexity we experienced a large loss of features as seen in Fig. 7, where none of the models seemed to be able to give good results.

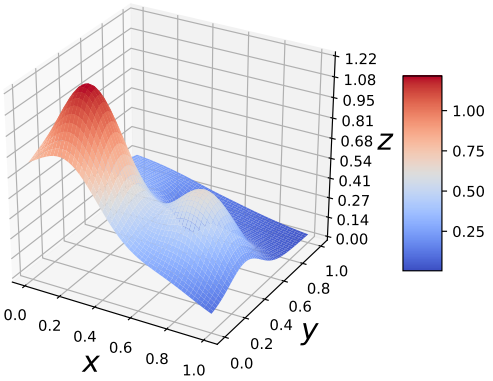
E. Conclusions

Summing up the analysis of this study we see that our regression methods perform reasonable when predicting a function like the Franke function. However when our model performs poorly when attempting to parameterize a varied terrain. This is due to the model being based on polynomials which cannot represent the varied and near discontinuous changes in the terrain. Since the polynomials are smooth we experience a loss of features in the terrain, rendering our model useless at accurately representing the data. Furthermore the model also does not perform well at predicting terrain other than the terrain we have built our model upon. One interpretation is that if we want a model that is general enough for this purpose, the loss of features from the current dataset would be even more severe. Again due to the limiting factors of building the model on polynomials. The types of datasets we could predict would tend to be smoother than the Norwegian terrain that we attempted to parameterize in this paper.

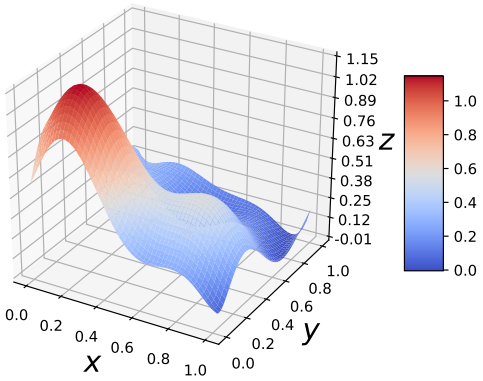
Appendix A: Figures

Figures 8 and 9 are not discussed in the report, but are included since they may be interesting to the reader.

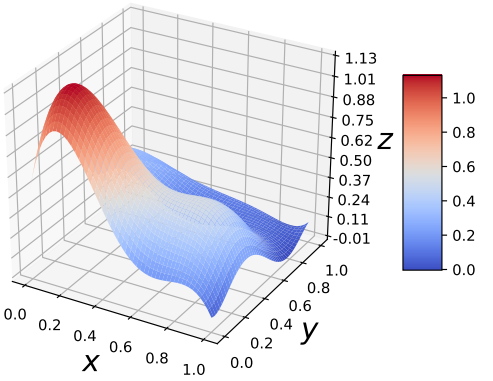
[1] U. S. Government, United states geological survey (2019).
[2] J. L. Devore and K. N. Berk, *Modern Mathematical Statistics with Applications* (Springer, 2012).
[3] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Springer, 2017).



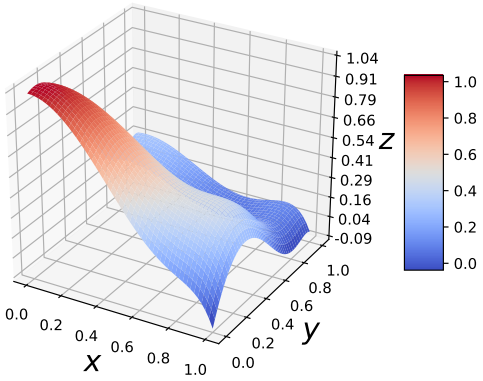
(a) Visualization of terrain generated by the Franke function, using 50x50 points and no noise.



(b) Visualization of OLS prediction on dataset as defined in Fig. (a).



(c) Visualization of Ridge prediction ($\lambda = 1.247e \cdot 10^{-4}$) on dataset as defined in Fig. (a).



(d) Visualization of Lasso prediction ($\lambda = 5 \cdot 10^{-5}$ on dataset as defined in Fig. (a).

FIG. 8: Different types of regression on the Franke function.

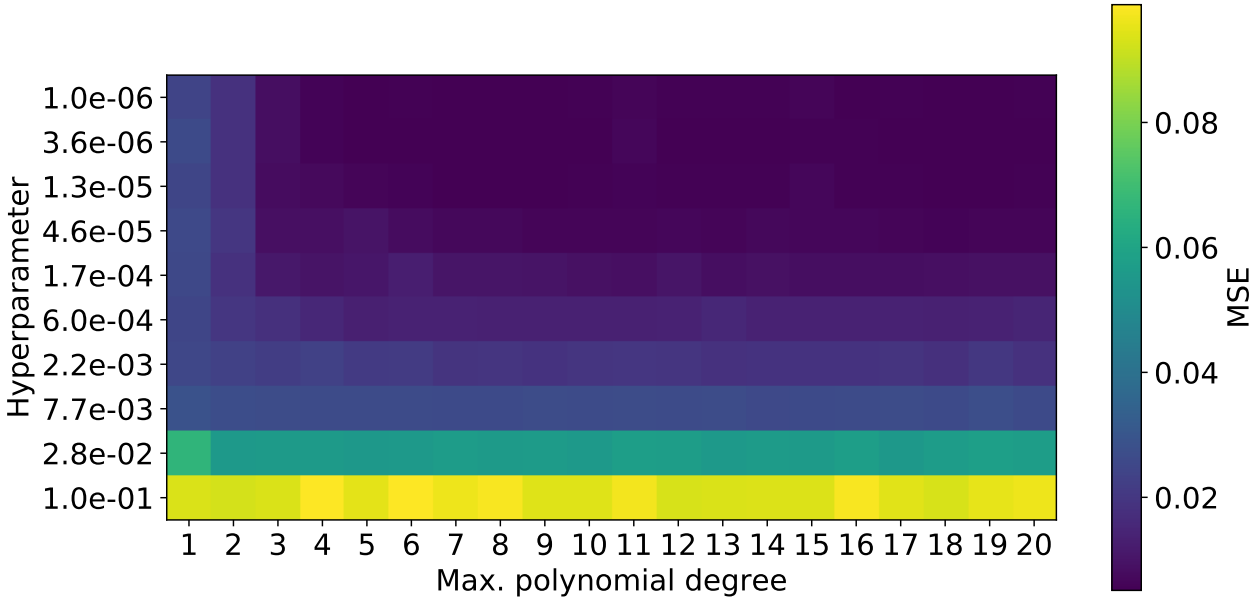


FIG. 9: Test MSE from using Lasso regression with $\lambda = 1.274 \cdot 10^{-4}$ to fit a model on dataset generated by the Franke function.