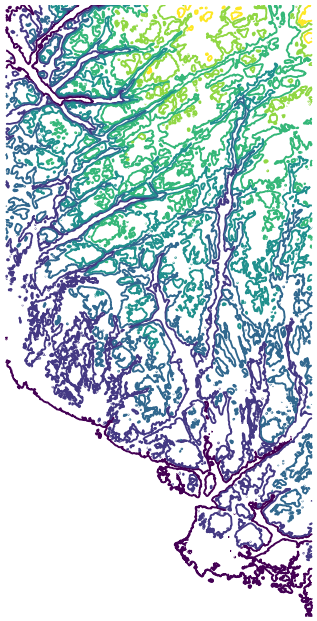
Regression Analysis and Resampling Methods
— Project 1 —
FYS-STK4155

Heine Aabø Halvard Sutterud

October 2018



Abstract

In this project, we study different methods for regression analysis, with focus on regularization and data resampling. We first try to fit the Franke function using polynomials of degree 5. To do this, routines for both Ordinary Least Squares (OLS) and Ridge Regression are implemented in python. Both K-fold validation and bootstrap is used to get better estimates of the mean squared error (MSE) and R2-score, and their variances. With such a simple model, ordinary least squares has the best performance, with an R2 score of 0.85 ± 0.06 for a sample size of $N = 2000$.

We then study the ridge regression and Lasso methods more specifically, and show the dependence of the parameters with respect to the penalty λ. Both methods force the parameters to zero as λ increases, as expected. The relative performance of our methods are also independent of the amount of noise. Stability analysis and bias variance decomposition show OLS breaks down when number of parameters relative to sample size becomes too large.

We then try to fit more complicated polynomials to terrain data. We fit some ridges near Stavanger with both Ordinary Least Squares, Ridge regression and Lasso regression. Unsuprisingly Ridge regression had the best performance. Our model obtained an R2 score of between 0.71 in the worst case and 0.98 in the best case. In addition, we obtain some visually pleasing fits to the terrain.

Contents

1 Introduction

As a data scientist, one typically wants to explain or predict data. One is interested in finding the best model, in the sense that it fits and predicts the data best given some underlying variables. This is a nontrivial task, as the amount of data available to train the model might be limited. However, we will show that, for the not too complicated tasks, regression analysis can be well suited for cases of medium sized data sets.

In this project, we will study three different regression methods, namely Ordinary Least Squares (OLS) regression, Ridge regression and Lasso regression. First we are going to use these to fit noisy data to the Franke function, given by a weighted sum of four exponentials, and then we will evaluate their performance. The performance of the model is be found by considering the decomposition of the error into variance, bias and data noise, and we estimate these using the resampling technique of K-fold-validation. The stability of the methods are also studied. Finally, after implementing the methods and testing on the Franke function, we will use the same methods on real terrain data.

This project consists of four sections after the introduction, where the first two introduce respectively the theory and the implementations of the methods used. Following this we will briefly state the results, before a more comprehensive discussion. Supplementary material such as extra figures, data and notebooks to reproduce the results can be found at https://github.com/halvarsu/FYS-STK4155.

2 Theory

2.1 Regression Analysis

Regression analysis is a method of fitting a statistical model $g(\hat{\beta})$ to some measured data $\hat{z} \in \mathbb{R}^n$. We typically wish to approximate some underlying function with a set of parameters $\hat{\beta}$ each corresponding to one column, or predictor, in the so called design matrix $X \in \mathbb{R}^{n \times p}$. The goal of the analysis is to explain new data points of this function, which can be written as

$$\hat{z} = \tilde{z} + \epsilon, \tag{1}$$

where ϵ is the error we want to minimize for all data, assumed to be normally distributed, and the $\tilde{z}$ is the predicted value,

$$\tilde{z} = X\hat{\beta}, \tag{2}$$

To train the model, we minimize the value of a cost function $C(\hat{z}, g(\hat{\beta}))$ with respect to β for a given training data set, which is a subset of the total data. One commonly used cost function is the least squares, which we will come back to in section 2.2.

2.2 Least squares regression

Rewriting eq. (1), the error can be written as $\hat{\epsilon} = \hat{z} - \tilde{z}$. A natural cost function is the sum of squares of the errors,

$$C(\hat{z}, g_{\mathcal{L}}(\hat{\beta})) = \sum_{i=1}^{p} (z_i - \tilde{z}_i)^2, \tag{3}$$

which gives us least squares regression. To minimize the error, we find the derivative of the cost function and set it to zero,

$$\frac{\partial C(\hat{z}, g_{\mathcal{L}}(\hat{\beta}))}{\partial \hat{\beta}} = \frac{\partial}{\partial \beta} \sum_{i=1}^{n} (z_i - \tilde{z}_i)^2 = 0. \tag{4}$$

This can be rewritten in the following way

$$\begin{aligned}
\frac{\partial C(\hat{z}, g_{\mathcal{L}}(\hat{\beta}))}{\partial \beta_i} &= \frac{\partial}{\partial \beta_i} \sum_{j=1}^{n} \left(z_j - \sum_{k=1}^{p} X_{jk}\beta_k \right)^2 && (5)\\
&= -2 \sum_{j=1}^{n} \left(z_j - \sum_{k=1}^{p} X_{jk}\beta_k \right) \frac{\partial}{\partial \beta_i} \sum_{l=1}^{p} X_{jl}\beta_l && (6)\\
&= -2 \sum_{j=1}^{n} X_{ji} \left(z_j - \sum_{k=1}^{p} X_{jk}\beta_k \right) && (7)\\
&\Rightarrow \hat{X}^T \left(\hat{z} - \hat{X}\hat{\beta} \right) = 0, && (8)
\end{aligned}$$

where the second line follows from the core rule, and the third line was recognized as the i'th element of the matrix-vector expression on the last line (times an insignificant constant). The last line is also called the normal equations. Solving for $\hat{\beta}$ gives the optimal parameters as

$$\hat{\beta} = (\hat{X}^T \hat{X})^{-1} \hat{X}^T \hat{z}, \tag{9}$$

which is a problem solvable on a computer through matrix inversion, as long as $X^T X$ is invertible.

2.3 SVD

If the model involves many parameters, the chances increase that some columns in the design matrix may become linearly dependent, or at least very close to it. In that case, $X^T X$ will become singular, which makes eq. (18) break down. This is especially the case on a computer, where the ability to represent numbers is limited to a certain precision. When the size of the determinant becomes very small, a matrix which is supposed to be singular can then instead be represented as a small non-zero value, giving nonsense values on the other side. One way to solve this problem is to use the SVD decomposition. We present here the main results of SVD which we need for our algorithm, see [2] for a more thorough review of the topic.

For any matrix $A \in \mathbb{C}^{n \times m}$, an SVD decomposition exists such that

$$A = U \Sigma V^T, \tag{10}$$

where U and V are unitary matrices whose columns are the orthonormal eigenvectors of AA^T and $A^T A$ respectively, and

$$\Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \in \mathbb{C}^{m \times n}, \tag{11}$$

is a block matrix, where the non-zero block is a diagonal matrix $\Sigma_{1,ij} = \sigma_i \delta_{ij}$ consisting of the singular values σ_i. The σ_i's are ordered such that $\sigma_1 \geqslant \cdots \geqslant \sigma_r > \sigma_{r+1} = \cdots = \sigma_{\min(m,n)} = 0$, where r is the rank of A. Converting this to our case, consider the matrix part of the right hand side of the normal equations in eq. (18),

$$\left(\hat{X}^T \hat{X} \right)^{-1} \hat{X}^T. \tag{12}$$

Inserting eq. (10) with $A = X$, we use the unitarity of U and V to get

$$\begin{aligned}
\left(\hat{X}^T \hat{X} \right)^{-1} \hat{X}^T &= \left(V \Sigma^T U^T U \Sigma V^T \right)^{-1} V \Sigma^T U^T \\
&= V^{-1} \Sigma^{-1} (\Sigma^T)^{-1} V^{-1} V \Sigma^T U^T \\
&= \left(U^T \right)^{-1}. && (13)
\end{aligned}$$

We can now use eq. (11), which tells us that the only part of the expression contributing to the optimal $\hat{X}$ is the first r columns of U and V. Decomposing them as $U = [U_1, U_2]$, $U_1 \in \mathbb{R}^{m \times r}$ and $V = [V_1, V_2]$, $V_1 \in \mathbb{R}^{n \times r}$, we see that the orthogonality now translates into

$$U_1 U_1^T = \begin{bmatrix} \mathbb{I}_r & 0 \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{m \times m} \tag{14}$$

and similarly for V_1. Combining this with eqs. (13) and (18) gives us a simplified expression for the optimal parameters when the rank of A is less than its dimensionality

$$\hat{\beta} = V_1 \sigma_1^{-1} U_1^T \hat{z}. \tag{15}$$

2.4 Ridge and lasso regression

Overfitting and underfitting occurs in least-squares regression often as a result of the parameters being too large. To avoid this a regularization term is added in whats called ridge regression and lasso regression, where the cost function is given by eq. (16) and eq. (17), respectivly.

$$C(\hat{z}, g_{\mathcal{L}}(\hat{\beta})) = \sum_{i=1}^{n} (z_i - \beta_0 - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_i|^2 \tag{16}$$

$$C(\hat{z}, g_{\mathcal{L}}(\hat{\beta})) = \sum_{i=1}^{n} (z_i - \beta_0 - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_i| \tag{17}$$

where X_{ij} is the elements of the design matrix.

Using the same approach as in section 2.2, the parameters β_{ridge} is given as

$$\hat{\beta}_{ridge} = (\hat{X}^T \hat{X} + \lambda \hat{I})^{-1} \hat{X}^T \hat{z}, \tag{18}$$

The main feature of lasso is that it has the ability to change parameters to zero for low values of λ and get rid of the ones that are redundant. However this can also lead to loss of precision.

2.5 Variance in the parameters

In the linear regression model, we find the variance in the parameters analytically by the expression given in the lecture notes [1],

$$\text{var}(\beta) = \left(\hat{X}^T \hat{X}\right)^{-1} \sigma^2 \tag{19}$$

with

$$\sigma^2 = \frac{1}{N - p - 1} \sum_{i=1}^{N} (z_i - \tilde{z}_i)^2. \tag{20}$$

Since the formula for β_{ridge} is different than for β_{ols}, the formula for variance will be different as well. A good resource on this is [4], which states

$$\text{var}(\beta_{ridge}) = \sigma^2 \hat{W}_\lambda \left(\hat{X}^T \hat{X}\right)^{-1} \hat{W}_\lambda^T \tag{21}$$

where

$$\hat{W}_\lambda = \left(\hat{X}^T \hat{X} + \lambda \hat{I}\right)^{-1} \hat{X}^T \hat{X} \tag{22}$$

2.6 Resampling methods

To evaluate the performance of a fit, it is often desired to test the prediction abilities of the model. When the available data consist of a restricted amount of samples, a good way to do this is by resampling the data set. This gives us multiple different data sets on which the model can be tested. Some statistic related to the model can be made from the average of each fit. Usually this is the mean square error or the R^2-score, given by

$$MSE(\hat{z}, \hat{\tilde{z}}) = \frac{1}{n} \sum_{i=0}^{n-1} (z_i - \tilde{z}_i)^2 \tag{23}$$

$$R^2(\hat{z}, \hat{\tilde{z}}) = 1 - \frac{\sum_{i=0}^{n-1} (z_i - \tilde{z}_i)^2}{\sum_{i=0}^{n-1} (z_i - \bar{z}_i)^2} \tag{24}$$

The score R^2 has a maximum value 1.0 and is a measure of how good the model can predict future samples. $R^2 = 0$ indicates a linear fit equal to the mean. While negative R^2-values indicates a fit that do not follow the trend of the data.

2.6.1 K-fold cross validation

A common resampling method is K-fold cross validation. This involves dividing the data set into k subsets of approximately equal size. The different subsets then iteratively plays the role as a test set, while the rest is assembled into a training set. After training the model, the mean square error and R^2-score can be found by fitting the test data.

2.6.2 Bootstrap

Another common method is the bootstrap. It picks a set of random samples from the data set and uses it as the training set, while the rest of the samples become the test set. Since the samples are picked randomly, it is possible for the same sample to appear multiple times in the training set, which is called sampling with replacement. The advantage of this method is that the training set is not a fixed partition of the original data set, and so the statistics are not the average of local statistics.

2.7 Bias and variance

A central concept in modeling is the decomposition of the error of our prediction into bias and variance. A model with high variance typically gives larger fluctuations around the true value, while higher bias corresponds to a larger error in the average of the estimates. As an expression for the error in the model, consider the expectation value of the cost function $C(\hat{z}, g_{\mathcal{L}}(\hat{\beta})$ over both the noise ε and different training sets $\mathcal{L}$,

$$E_{\mathcal{L},\varepsilon}[C(\hat{z}, g_{\mathcal{L}}(\hat{\beta}))] = E_{\mathcal{L},\varepsilon}\left[\sum_i (z_i - \tilde{z}_i)^2\right]. \tag{25}$$

Under the assumption that the error is normally distributed with zero mean and variance σ_ε this can be decomposed as (derived from [3])

$$E_{\mathcal{L},\varepsilon}[C(\hat{z}, g_{\mathcal{L}}(\hat{\beta})] = \sum_i \sigma_\varepsilon^2 + \sum_i \left(f(\vec{r}_i) - E_{\mathcal{L}}\left[g_{\mathcal{L}}(x_i, y_i)\right]\right)^2 + \sum_i E_{\mathcal{L}}\left[\left(g_{\mathcal{L}}(x_i, y_i) - E_{\mathcal{L}}[g_{\mathcal{L}}(x_i, y_i)]\right)^2\right]. \tag{26}$$

The noise in the data is completely contained in the first term, while the next two is the bias and variance of the model $g_{\mathcal{L}}$.

To estimate these values, we can in general not access the values of $f(\vec{x})$ directly, and averaging over all data sets may be unfeasible (especially when even a single calculation is very expensive). We will overcome this to some degree, respectively by using measured data instead of ground truth, and using resampling methods.

TODO: Fiks notasjonen i teori så vi har konsistent notasjon for datasett (x, y), utdata z, predikert data $\hat{z}$, osv.

3 Methods and implementation

3.1 Function and model

The test our models and statistical tools, we used a function $f(x, y)$ for $x, y \in [0, 1]$, with observed values $z_i = f(x_i, y_i) + \varepsilon$, where ε is noise generated from a normal distribution $N(0, \sigma_\varepsilon)$. We used the Franke function, seen in fig. 1, which is given by a weighted sum of exponentials as

$$\begin{aligned} f(x,y) = &\frac{3}{4}\exp\left(\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) \\ &+ \frac{3}{4}\exp\left(\frac{(9x+1)^2}{49} - \frac{(9y+1)^2}{10}\right) \\ &+ \frac{1}{2}\exp\left(\frac{(9x-7)^2}{4} - \frac{(9y-3)^4}{2}\right) \\ &- \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right). \end{aligned} \tag{27}$$

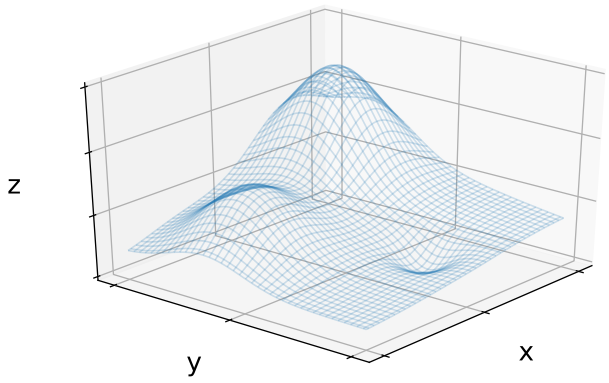


Figure 1: The Franke function, for $x, y \in [0, 1]$

The model $g(\hat{\beta})$ chosen was a two-dimensional polynomial with terms of degree up to k in x, y and products of the two. We mainly used $k = 5$, which yields $p = (k+1)(k+2)/2 = 21$ independent parameters β from the two predictors.

3.2 Implementation

In each numerical experiment, we first generated n points (x_i, y_i) for $i = 1, \ldots, n$ from a uniform distribution, and found the corresponding observed values. This was considered as out total data set. The design matrix was then set up as $X_{i,j} = x_i^{n_j} y_i^{m_j}$, for n_j and m_j such that all polynomial degrees are included in increasing order. $\hat{X}$ was then used to train model and provide the best fits of the parameters $\hat{\beta}$, and a $\hat{X}'$ made from a distinct data set was used to predict new values $\tilde{y}' = \hat{X}'\hat{\beta}$ for testing purposes.

We created a python class named `Regression` in the module `tools.py`. It works similarly as the regression classes in the python library `sklearn.linear_model`. Given a X and $\hat{z}$, it calculates the best fit parameters $\hat{\beta}$ using matrix inversion, for any model defined by X. If provided with a penalty λ the method used is the Ridge regression, else ordinary least squares is used. Also included in the class are methods to estimate the variance in $\hat{\beta}$, and for calculating the in-sample error (the error of the prediction of the training data versus the observed data) in the form of MSE (mean squared error) and R2 score. Both matrix inversion and SVD-decomposition was used to solve the linear problem, as described in **??**. To perform the Lasso regression, we used the `Lasso`-method found in `sklearn`.

To test the model, we resampled using K-fold validation and bootstrap methods implemented in the file `tools.py`. As we had access to ground truth data, we also resampled directly from $f(x, y)$ with noise, as this let us evaluate the resampling techniques directly. We varied the size of the data set to study the dependency of in-sample error and out-of-sample error on both penalty λ and training set size N. Bias and variance was then calculated, using the theory of section 2.7.

The methods for the terrain data was performed in much

as similar way. We used the data included in the course Github repository [1], over an area near Stavanger, Norway. From this data set, 6 random patches of dimensions 100×50 were selected, which we used for fitting the polynomials. To estimate the best parameters, we looped over polynomials $p \in 5, 6, \ldots, 17$, and estimated MSE and R2 with k-fold validation with $k = 10$ for OLS for each patch. For Ridge and Lasso we also looped over penalties $\lambda = 10^n, \quad n \in -7, \ldots, -1$. The original data and selected patches can be seen in fig. 2.

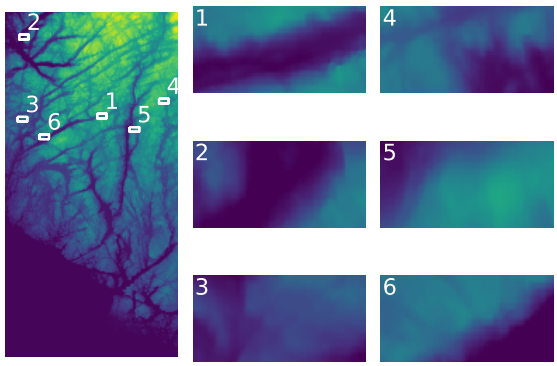


Figure 2: The terrain data used, an area near Stavanger in Norway, with 6 randomly selected patches.

4 Results

We now state our results.

4.1 Regression analysis of Franke function

Everything in this section is done with fifth degree polynomials for $N = 2000$ points.

4.1.1 OLS

A selection of the parameters is listed in table 1, with an uncertainty representing the 95% confidence interval of each parameters. For OLS and Ridge regression this uncertainty is the standard deviation of the parameters calculated using the theory in section 2.5.

β_j	Confidence interval
$j = 0$	0.3 ± 0.1
$j = 5$	-40.0 ± 4.9
$j = 10$	28.9 ± 10.7
$j = 15$	-16.4 ± 4.1
$j = 20$	2.7 ± 4.3

Table 1: Selection of parameters of the OLS regression with uncertainty.

The mean squared error and the $R2$-score of the model is listed in table 2. Here $k = 10$ in the k-fold, and in the bootstrap the sample-size is equal to N.

	No resampling	k-fold	Bootstrap
MSE	0.0122	0.013 ± 0.003	0.0127 ± 0.0006
R2	0.8734	0.85 ± 0.06	0.85 ± 0.01

Table 2: MSE and $R2$-score of OLS regression

4.1.2 Ridge

To find an optimal value for λ, the mean square error and $R2$-score was plotted in fig. 3 as a function of λ.

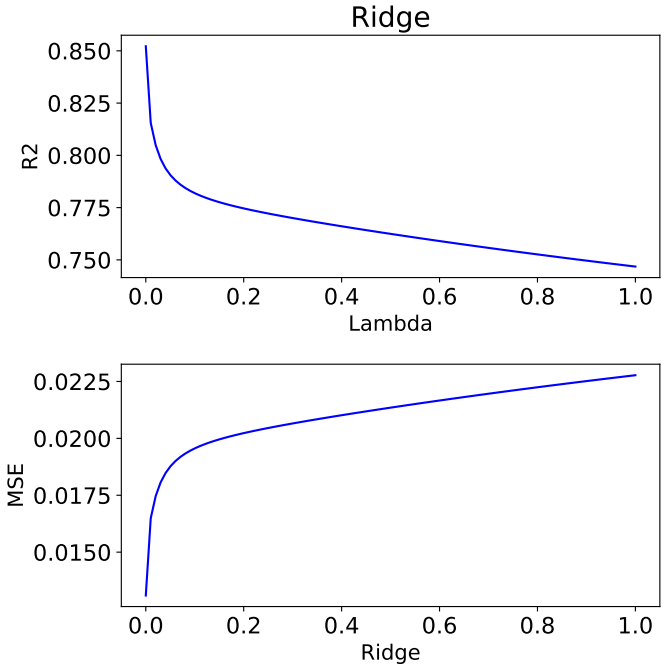


Figure 3: $R2$-score and MSE as a function of λ.

This gives an indication that low values of λ will give the best fit. In table 3 these values are given with and without resampling for low values of λ, where the uncertainties are the standard deviation after resampling. Again, $k = 10$ in the k-fold, and the sample-size is equal to N in the bootstrap.

	No resampling	k-fold	Bootstrap
$\lambda = 0.0001$			
MSE	0.0124	0.013 ± 0.004	0.0127 ± 0.0006
R2	0.8719	0.85 ± 0.06	0.848 ± 0.009
$\lambda = 0.001$			
MSE	0.0134	0.014 ± 0.005	0.0136 ± 0.0006
R2	0.8619	0.84 ± 0.07	0.83 ± 0.01
$\lambda = 0.01$			
MSE	0.0149	0.015 ± 0.005	0.0152 ± 0.0006
R2	0.8464	0.83 ± 0.07	0.81 ± 0.01

Table 3: MSE and $R2$-score of the ridge regression for three different values of λ.

$\lambda = 0.0001$ gives the best results, and a selection of its parameters is given table 4. The parameters are found with a different formula than OLS, so the variance also has a different formula, given by eq. (21).

β_j	Confidence interval
$j = 0$	0.49 ± 0.06
$j = 5$	-25.7 ± 3.0
$j = 10$	21.5 ± 7.0
$j = 15$	-12.8 ± 2.9
$j = 20$	-1.8 ± 2.9

Table 4: Selection of parameters of the ridge regression with confidence intervals.

The evolution of the parameters as a logaritmic function of λ is plotted in fig. 4 for small values, since they just approach 0 as $\lambda \to \infty$.

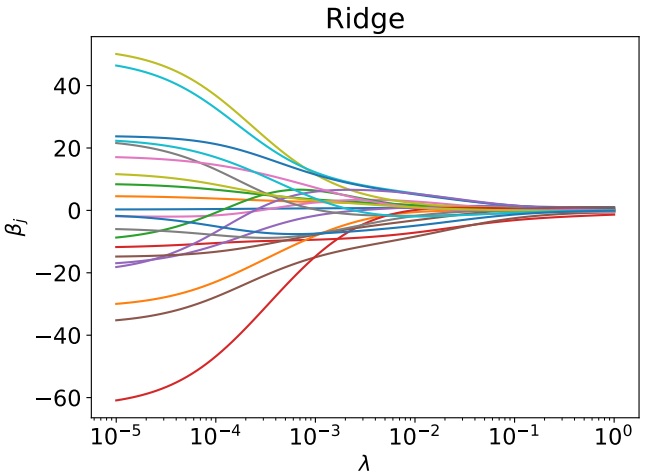


Figure 4: The parameters of the ridge regression as a function of λ, where $10^-5 \leqslant \lambda \leqslant 1$.

4.1.3 Lasso

Again the the mean squared error and $R2$-score was plotted as a function of λ.

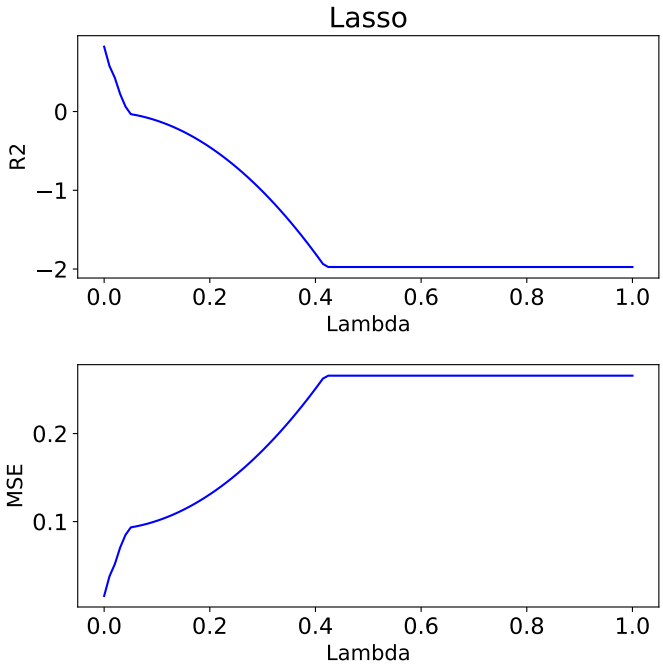


Figure 5: $R2$-score and MSE as a function of λ.

As for ridge, low values of λ seems to give the best fit. The mean squared errors and $R2$-scores in table 5 are given for the same values of λ as in section 4.1.2. With $k = 10$ in the k-fold, and the sample-size is equal to N in the bootstrap.

	No resampling	k-fold	Bootstrap
$\lambda = 0.0001$			
MSE	0.0195	0.019 ± 0.007	0.020 ± 0.001
R2	0.7988	0.78 ± 0.09	0.74 ± 0.01
$\lambda = 0.001$			
MSE	0.0236	0.024 ± 0.009	0.024 ± 0.001
R2	0.7562	0.74 ± 0.09	0.63 ± 0.02
$\lambda = 0.01$			
MSE	0.0389	0.04 ± 0.02	0.039 ± 0.002
R2	0.5981	0.58 ± 0.09	-0.1 ± 0.1

Table 5: MSE and $R2$-score of the lasso regression for three different values of λ.

The class `sklearn.linear_model.Lasso` do not include a method for the variance in the parameters. The variances can be found analytically, e.g. by taking the standard deviation of the parameters after k-fold validation with a reasonable k. In table 6 the uncertainties was found using $k = 10$.

β_j	Confidence interval
$j = 0$	0.70 ± 0.01
$j = 5$	-0.13 ± 0.04
$j = 10$	0
$j = 15$	0
$j = 20$	0

Table 6: Selection of parameters of the lasso regression with uncertainty.

The evolution of the parameters as a function of λ is shown in fig. 6.

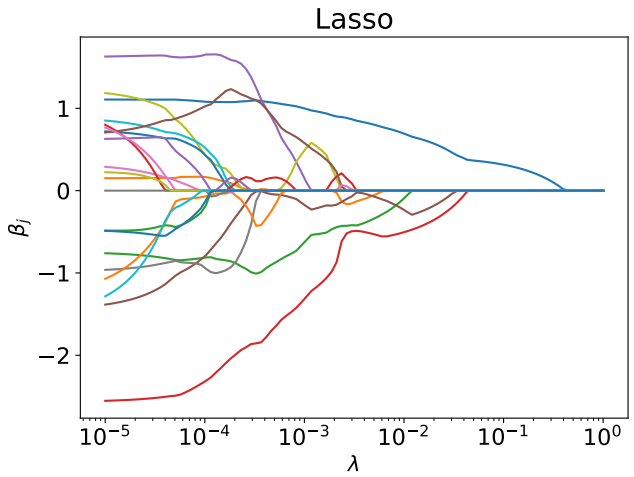


Figure 6: The parameters of the lasso regression as a function of λ, where $10^-5 \leqslant \lambda \leqslant 1$.

4.2 Noise dependency

To further evaluate any differences between the methods, the mean squared error were plotted as a function of noise (added to the Franke function). The noises in figs. 7 and 9 range from 0.0001 to 2.

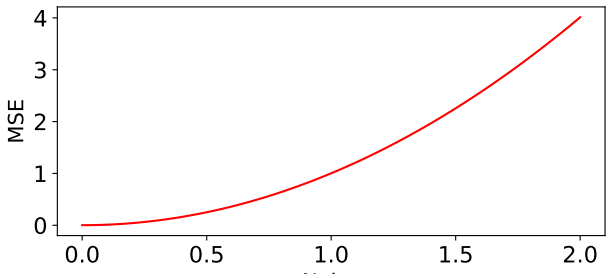


Figure 7: OLS regression. MSE as a function of noise

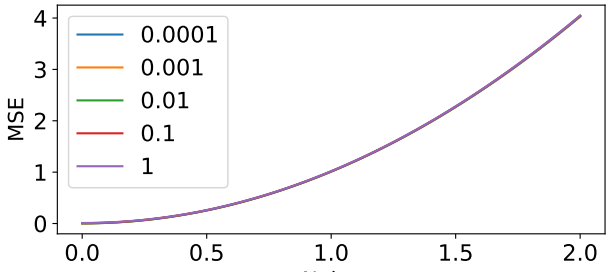


Figure 8: Ridge regression. MSE as a function of noise, for different values of λ.

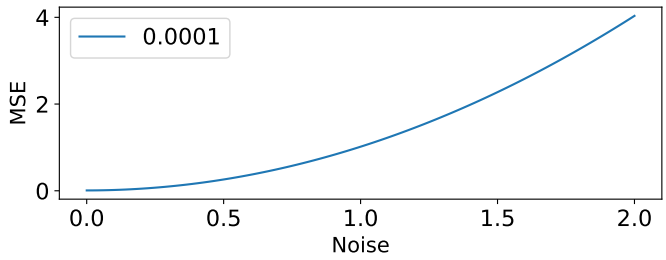


Figure 9: Lasso regression. MSE as a function of noise, for different values of λ.

4.3 Covariance in the OLS parameters

The covariance matrix for the $\hat{\beta}$ calculated analytically for the Franke function with the expression given in section 2.5 is shown in fig. 10.

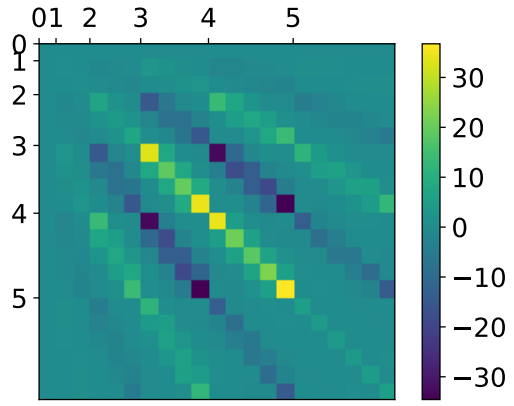


Figure 10: Representation of the covariance matrix of the parameters, sorted by polynomial degree along the x- and y-axis.

4.4 Error as function of sample size

In fig. 11, we see the estimated in-sample and out-of-sample error of the model for different training data sizes for OLS and Ridge, compared with the size of the determinant of the inverted matrix. This shows the dependency of the models on the amount of samples for the given model complexity, and gives also a good comparison between the Ridge and Lasso as the number of data points is reduced.

part which was different when the linear problem was solved with SVD-decomposition (see section 2.3). The notable difference is the in-sample error which stays at zero for low N.

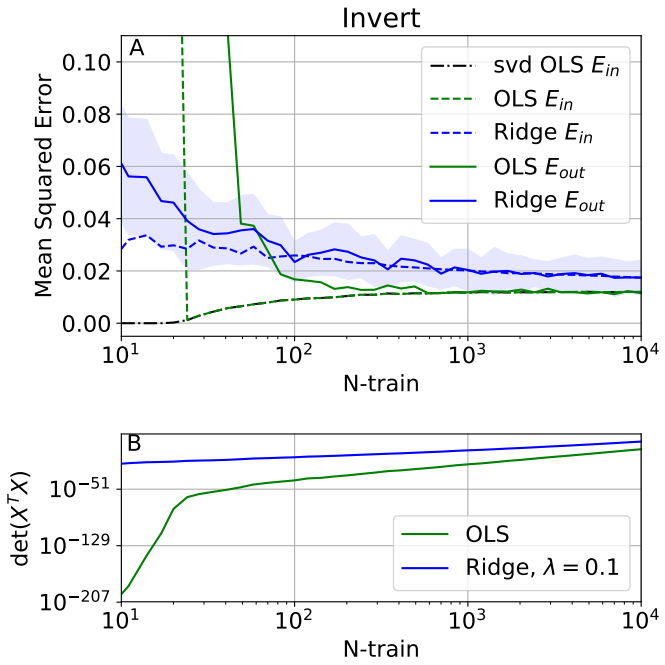


Figure 11: **A:** The in-sample and out-of-sample errors of our models for different sizes of the training set, calculated by generating points in $n_{rep} = 40$ batches, and training for each batch. OLS and Ridge with a penalty of $\lambda = 0.1$ was used. The standard deviation over the batches is shown as a shaded area for the out-of-sample error. The linear problems are solved by inverting X^TX (see eq. (18)). Also shown is the in-error of the OLS when SVD-decomposition was used. **B:** The determinant of the matrix which is inverted in OLS and Ridge respectively in the upper figure. Note the correspondence between instabilities in the error and small values of the determinant.

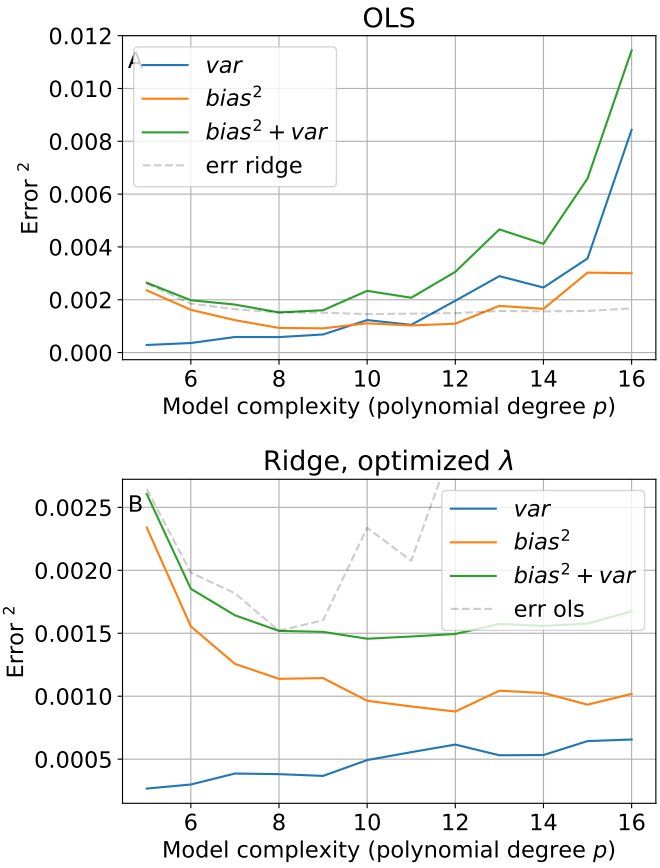


Figure 12: The reducible error of the model decomposed into bias and variance for ordinary least squares regression in **A**, and Ridge regression in **B** with penalty $\lambda = 10^{-5}$ in most of the cases, found by searching through several values of λ. This is plotted as function of model complexity, represented by polynomial degree p. Sample size was set to $N = 1000$. We used SVD, as inverting a matrix gives singular matrices above $p = 10$.

4.5 Decomposition of error

The decomposition of the error of our model into variance and bias can be seen in fig. 12 for training data set of fixed size $N = 1000$, as a function of model complexity (polynomial degree).

4.6 Terrain data

We found the Ridge regression for each of the terrain patches, as it gave the highest R2 score when the optimal parameters found were used. See fig. 13. In addition, we found that Lasso performed worst in all three cases, even after optimizing the λ. Figure 14 shows that the variance is also greater in Lasso and OLS than in Ridge. In fig. 15, the actual values of the penalty λ are plotted against the optimal polynomial degree p for Ridge and Lasso.

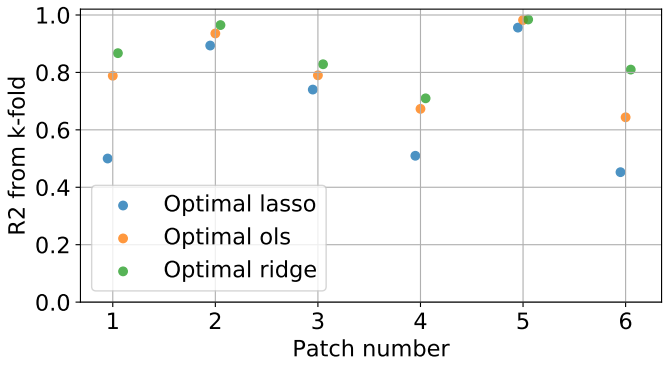


Figure 13: The R2 values of the optimal fit for OLS, Ridge and Lasso over the six terrain patches, calculated using K-fold with $k = 10$.

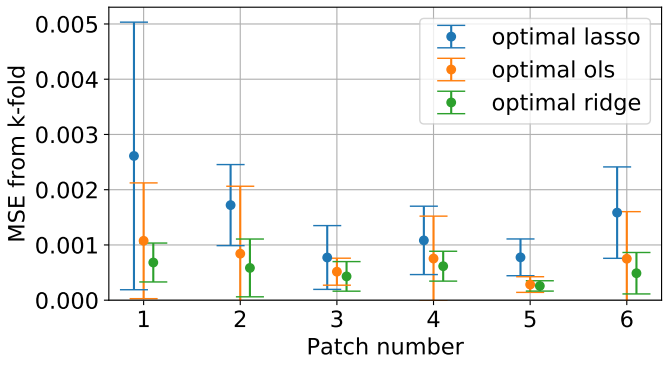


Figure 14: The MSE of the optimal fit for OLS, Ridge and Lasso over the six terrain patches, with errors. Mean value and std of the errors are extracted from K-fold with $k = 10$.

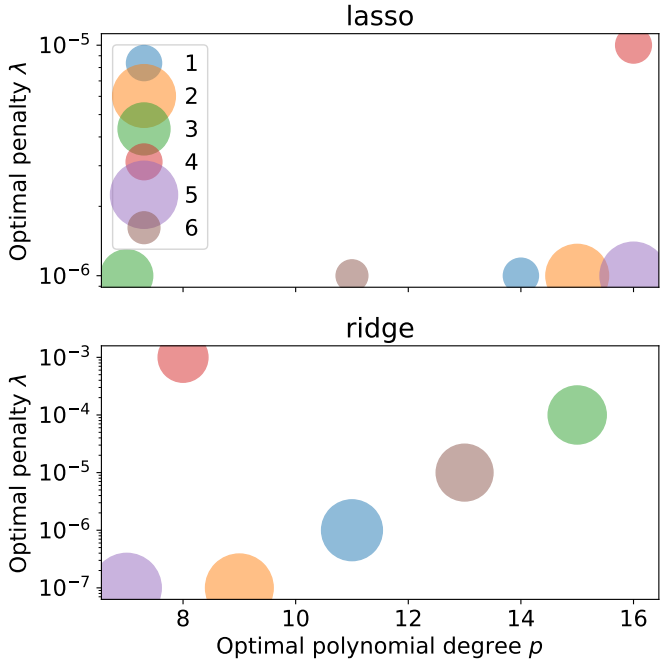


Figure 15: The values of the penalty λ versus polynomial degree p for Ridge and Lasso, for the six different terrain patches (colored equally between plots). The markers are scaled by the R2 score.

Finally, the best fits (all using Ridge regression) are visualized in fig. 16.

5 Discussion

5.1 Regression analysis of the Franke function

It is apparent from section 4.2 that the MSE of all methods are equal for low values of λ, and behaves like the noise squared, σ_ε^2. Higher values of λ seems to be proportional to this. Going back to eq. (23) and eq. (24) it is obvious that the $R2$-score will behave the same for all methods; it will be very good for low values of noise, and go towards 0 as the noise increases. So all methods handle noise in the franke function in the same way, and perform much better for low values of noise. As a result, there is no need to consider the noise when comparing the performance of the methods.

One obvious difference between the methods is the size of the parameters. The OLS stands out by both having bigger parameters than ridge and lasso, while also having larger confidence intervals and derby larger variance. This is expected as the goal of the penalty λ is to lower the variance. As a consequence they will not grow too big. This is also some of the reason that Ridge and Lasso performs worse than OLS in some cases, because the parameters are suppressed they don't take large enough values to represent all the changes in the data set.

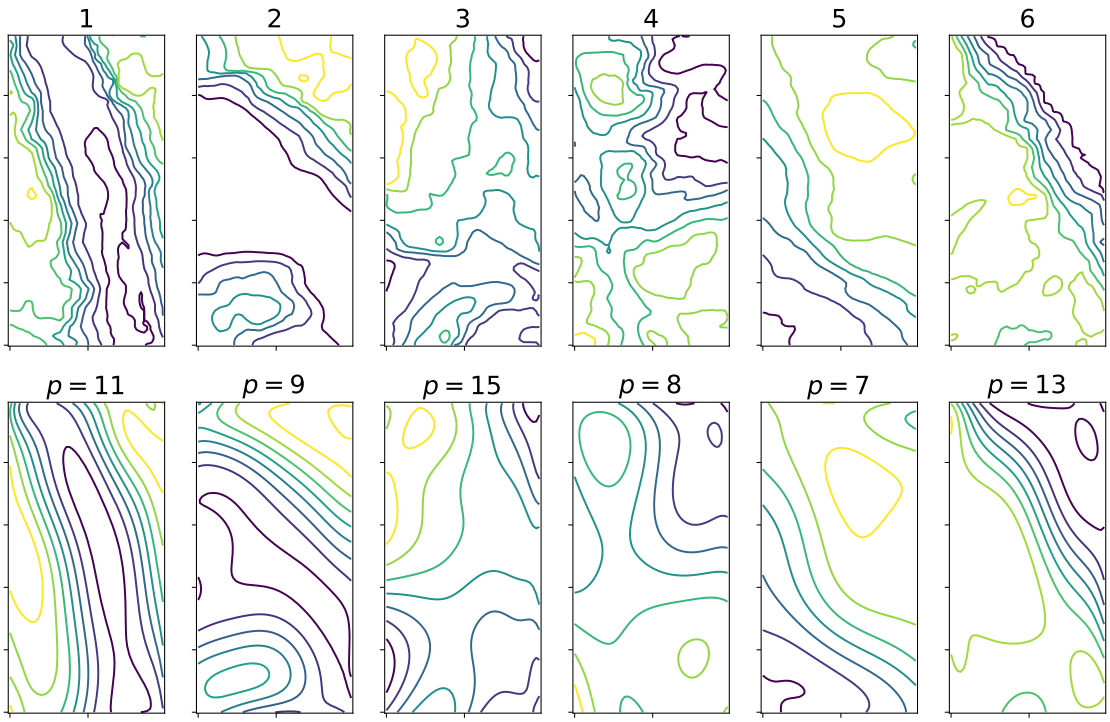


Figure 16: **Upper:** Contour lines of the terrain patches. **Lower:** Our polynomial fits to the terrain patches, all using Ridge regression as this was found to give the lowest error.

Regarding table 6, several parameters is zero, which is expected from the lasso. This makes the method stand out the most. A quick look at table 5 indicates that this method is not the best for predicting values of the Franke function. One possible reason for this might be that all the parameters of the model are important, and by tuning most of them to zero the fit will not become accurate.

The lasso performs bad as λ gets bigger, the same as for ridge. The method uses, `sklearn.linear_model.Lasso` which caused precision problems for low values of λ, when the number of iterations was insufficient. The algorithm uses coordinate descent and did not always converge for low values of λ. A high number of iterations was necessary for it to converge. We tested this for 100000 iterations, which got very different parameters from the standard that was used in section 4.1.3 (1000 iterations). In addition, it needed a lot of time to compute and did not give any better results. Perhaps an even higher number would give better results, but this would make lasso way too inefficient in comparison to ridge and OLS.

In table 3 and table 5 the performance of the regression gets better as λ gets smaller. There is a clear indication that the performance of the regression will get better as $\lambda \to 0$. This is supported by the fact that OLS (see table 2) gives the best performance, with lowest MSE and highest $R2$-score.

Finally, it is worth commenting the results in section 4.5, where the bias-variance decomposition is plotted as a function of model complexity. For OLS the error is lowest for degree $p = 8$ and $p = 10$ for ridge. The MSE and $R2$-score of both methods were found for these degrees. For $P = 10$ ridge got a 0.0001 better $R2$-score than OLS with the same MSE. However, OLS had an even better for $p = 8$.

5.2 Bias and variance

Typically more complex models have more variance, but less bias. The more degrees of freedom a model has, the more parameters, and the greater is the chance of the overfit; it might perfectly fit the training data but fail to predict new values. This is especially the case with Ordinary Least Squares, which uses all possible parameters to reduce the error with respect to the training data it has at hand. Regularization methods on the other hand introduce some sort of penalty on the number of degrees of freedom the model utilizes, which cause them to reduce the value of the least important parameters. This has the effect of lowering the number of effective degrees of freedom, giving less overfit and thus sess variance.

To calculate the bias and variance, we realized that the expression for the bias involve ground truth data without noise, which makes it hard to calculate in real data scenarios. Using the Franke function we could generate new data, and as seen in fig. 12 we manage to replicate figure 5 of [3], showing that there indeed exists a tradeoff between bias and variance in our polynomial model. This is also apparent for low amounts of training data in fig. 11, where OLS gives a large gap between in-sample and out-of-sample error while Ridge is more

stable.

5.3 Why use Ridge and lasso?

For fewer training data/more complex model, the symmetric $H = X^T X$-matrix can be close to singular/its determinant can be very small. This leads to numerically unstable OLS-solutions if they are built around matrix inversion, as the inverse of H is used. This is very obvious in fig. 11, when the number of data points approach the number of parameters, 21, the method fails to converge to the proper value. This is because the computer represents the singular matrix with some machine error, and the algorithms (`scipy.linalg.inv` in this case) thinks the matrix is invertible. A tiny fluctuation due to machine precision then becomes a gigantic number in the parameters. The same applies when the number of parameters in the model exceed some threshold (can be demonstrated using the supplementary material, by switching to matrix inverting in the Regression object). Ridge and Lasso do not have this problem, because both methods add elements along the diagonal of H. Using SVD also solves the problem, because one can determine a rank tolerance and remove the part of the data orthogonal to X.

Another known problem in linear solvers is highly correlated columns in the design matrix. This can cause OLS to choose unstable solutions where two parameters are set to extreme values, as they might cancel each other. This is solved by regularization, as one of the two correlated values can be suppressed by the algorithm. In this case, we first believed that Lasso would be more effective at suppressing the really bad columns of X, as values can be set to exactly zero, but our usage of Lasso did not show any advantages. Ridge can also set parameters to exactly zero, but this does not happen before $\lambda \to \infty$.

5.4 Terrain

Unfortunately, we forgot to randomize our data before resampling with K-fold. The data used was instead ordered in a C-ordered matrix-fashion. This probably caused a larger out-of-sample error, as the convex sets of all the test data never contained any training data. As such, our estimates for the MSE and R2 and their deviations in the case of the terrain data might be biased upward. We did not have time to do new simulations at the time of realizing this mistake.

Given our results though, we consider Ridge regression to be best suited to the terrain data. It need some optimization to find the best λ, but gives better results than the two other methods, as seen in fig. 13. We don't know if this fixing our mistake would change this, but it can be argued that the mistake should not affect the performance of the methods that much.

Figure 15 shows that there is a certain trend of increasing optimal λ for increasing model complexity. This makes sense with respect to what we earlier discussed, as when the number of parameters increases, we want to increase the penalty to exclude the less important ones. We see that Lasso tends to give better results for complex polynomials, while ridge gives a larger spread among the different values. Comparing with fig. 16, we also see that the terrain data which required the highest order polynomials were the ones with large fluctuations, i.e. patch number 6 which has a steep cliff stooping down into a flat fjord, or the ones with complicated shapes with gradients pointing in all directions such as patch number 3.

5.5 Conclusion and further work

The best fit to the Franke function was made with the OLS regression model, this was mainly tested for polynomials of fifth degree, but higher order polynomials gave the same results. Ridge regression gave good results and as $\lambda \to 0$ the MSE and $R2$-score got very close the OLS. Lasso however did not perform as well as the other, which

Our specific terrain data was fit best with Ridge regression

We also showed that OLS has its limitations when the model complexity increases relative to the number of sample points, the variance then

Our evaluation using Lasso failed to show its strengths. It might perform better on data where the

References

[1] Course web page. `https://github.com/CompPhysics/MachineLearning`, October 2018. original-date: 2017-09-18T20:11:45Z.

[2] Dmitriy Leykekhman. MATH 3795 Lecture 9. Linear Least Squares. Using SVD Decomposition.

[3] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G. R. Day, Clint Richardson, Charles K. Fisher, and David J. Schwab. A high-bias, low-variance introduction to Machine Learning for physicists. *arXiv:1803.08823 [cond-mat, physics:physics, stat]*, March 2018. arXiv: 1803.08823.

[4] Wessel N. van Wieringen. Lecture notes on ridge regression. *arXiv:1509.09169 [stat]*, September 2015. arXiv: 1509.09169.