A Study of Datapreprocessing, Regression and Resampling Techniques Applied to Franke's Function and Terrain Data

René Ask, Kaspara Skovli Gåsvær & Maria Linea Horgen
(Dated: October 1, 2020)

In this report we implement three different solvers based on methods in Linear Regression, namely the Ordinary Least Squares method (OLS), Ridge regression and LASSO regression. Resampling techniques are used to discuss the bias-variance trade-off for the OLS method. The three methods are critically compared and contrasted using a dataset generated with Franke's function. Lastly we use Ridge regression on terrain data and find that the fitted models are highly data-dependent. Furthermore, the models' generalizability is not properly measured and computed performance metrics are questionable due to the biased approach to model-fitting.

I. INTRODUCTION

Machine Learning (ML) is a field of science which studies how to learn from data and make prediction on never-before-seen data. Nowadays, large datasets flood the internet and sifting through these datasets to identify important features manually is intractable and the field of ML provides methods to circumvent this problem and automatically learn which features matter and provide means to predict future data in a meaningful way. Yet, using it right can be difficult, with pitfalls such as under- and overfitting and human bias affecting the resulting models. The key measurement of a model's generalizability is how well it predicts unseen data, not how well it fits some particular set. This is the key differentiator ML has from classical statistics and is precisely what makes ML complicated.

In this article we present three regression methods. ordinary least-squares, Ridge and LASSO. The mathematical ideas underlying these are developed and and tested on two different datasets that both fit into the same framework. We discuss data preprocessing, the bias-variance trade-off and resampling techniques such as bootstrapping and k-fold cross-validation. The codes used in this article and be found here [3].

II. FORMALISM

A. Preliminaries

Suppose we're given a dataset of $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{z})$ of length n, where $\boldsymbol{X}$ contain the measured features and $\boldsymbol{z}$ is the response variable.

In this article, we assume the data is given by

$$z = f(x, y) + \epsilon, \tag{1}$$

where z is the measured data, $f(x, y)$ is the ground-truth and $\epsilon \sim \mathcal{N}(0, \sigma)$ is the noise with zero mean and standard deviation σ.

We assume a model class of polynomials of degree at most d

$$\hat{f}(x, y) = \sum_{i=0}^{d} \sum_{j=0}^{i} c_{i-j,j} x^{i-j} y^{j}, \tag{2}$$

which may be rewritten as

$$\hat{f}(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{w}, \tag{3}$$

with $\boldsymbol{x}^T$ given by the row vector

$$\boldsymbol{x}^T = \left[1, x, y, x^2, xy, y^2, \cdots, x^d, x^{d-1}y, \cdots, y^d\right], \tag{4}$$

and $\boldsymbol{w}$ contain the corresponding weights

$$\boldsymbol{w} = [w_1, w_2, \cdots, w_p]^T, \tag{5}$$

Note the appearance of p, which denotes the number of *features* of the model. In the case of our model, the number of features are related to the polynomial degree d by

$$p = \frac{(d+1)(d+2)}{2}, \tag{6}$$

which can easily be verified by induction. The number of features corresponds with the model complexity, and the terms will be used interchangeably throughout this report.

Given the dataset $\mathcal{D}$, we can form the so-called *design matrix* by stacking row-vectors containing measured features on-top of each others, resulting in the following generic matrix for our model

$$X = \begin{bmatrix} 1 & x_1 & y_1 & x_1^2 & x_1y_1 & \cdots & x_1^d & x_1^{d-1}y_1 & \cdots & y_2^d \\ 1 & x_2 & y_2 & x_2^2 & x_2y_2 & \cdots & x_2^d & x_2^{d-1}y_2 & \cdots & y_2^d \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ldots & \vdots & \vdots & \ldots & \vdots \\ 1 & x_n & y_n & x_n^2 & x_ny_n & \cdots & x_n^d & x_n^{d-1}y_n & \cdots & y_n^d \end{bmatrix}, \tag{7}$$

where n denotes the number of datapoints. This allows us to reformulate the model as a matrix equation of the form

$$\hat{\boldsymbol{f}}(X) = X\boldsymbol{w}. \tag{8}$$

B. Feature scaling

Raw data in datasets may vary widely, and machine learning algorithms may not work as intended without proper scaling. To remedy this, we introduce a type of *feature scaling* sometimes called the standard score. For the features, this means

$$\boldsymbol{x}' = \frac{\boldsymbol{x} - \langle \boldsymbol{x} \rangle}{\boldsymbol{\sigma_x}}, \tag{9}$$

and for the response variables

$$\boldsymbol{z}' = \frac{\boldsymbol{z} - \langle \boldsymbol{z} \rangle}{\boldsymbol{\sigma_z}}, \tag{10}$$

where the expressions are understood as *element-wise*.

C. Splitting the data

The dataset $\mathcal{D}$ contains all the data. To evaluate the performance of the model, the dataset in split into to smaller partitions: a training set $\mathcal{D}_{\text{train}}$ and a test set $\mathcal{D}_{\text{test}}$. This can be done with many different ratios, and in this article we split it so that 20% of the data is testing data and 80% is training data. The datasets are also reshuffled randomly to eliminate any human bias with respect to its ordering. Partitioning the dataset in this way provides a less biased estimate of the performance of the model.

D. Performance metrics

The mean-squared error, further denoted as MSE, of a finite sample of size n is given by

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(z_i - \hat{f}(\boldsymbol{x}_i))^2. \tag{11}$$

We'll refer to the *in-sample* error as the MSE computed on training data and the *out-of-sample* error as the MSE computed on test data.

Another performance metric which provides a simpler interpretation on performance is the so-called R^2-score which is defined as

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(z_i - \hat{f}(\boldsymbol{x}_i))}{\sum_{i=1}^{n}(z_i - \langle z \rangle)}, \tag{12}$$

with

$$\langle z \rangle \equiv \frac{1}{n}\sum_{i=1}^{n} z_i. \tag{13}$$

This performance metric is a perfect score if $R^2 = 1$.

E. Bias-variance tradeoff

We define a cost-function

$$\mathcal{C}(\boldsymbol{z}, \hat{f}(\boldsymbol{X})) = \sum_i (z_i - \hat{f}(\boldsymbol{x_i}))^2 \tag{14}$$

This cost-function is known as the residual squared error, RSS. The expected residual squared error $\mathbb{E}_{\mathcal{D}}\left[\mathcal{C}(\boldsymbol{z}, \hat{f}(\boldsymbol{X}))\right]$ on all possible datasets is

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\left[\mathcal{C}(\boldsymbol{z}, \hat{f}(\boldsymbol{X}))\right] &= \sum_i \Big(f(\boldsymbol{x}_i) - \mathbb{E}_{\mathcal{D}}[\hat{f}(\boldsymbol{x}_i)]\Big)^2 \\ &\quad + \sum_i \mathbb{E}_{\mathcal{D}}\left[\Big(\hat{f}(\boldsymbol{x}_i) - \mathbb{E}_{\mathcal{D}}[\hat{f}(\boldsymbol{x}_i)]\Big)^2\right] \\ &\quad + \sum_i \sigma^2,\end{aligned} \tag{15}$$

which is derived in the appendix. The first term is known as the squared *bias*. It's a measurement of the best possible out-of-sample error in the infinite data-limit. The second term is coined *variance*, and measures the degree to which the model fluctuates due to finite sample sizes. The third term is the *irreducible error* which is the variance the noise produce. For finite-samples, the bias will generally decrease as a function of model complexity (i.e number of features), while the variance increases. Put another way, a low model complexity gives a high bias and low variance, which results in *underfitting*. Increasing the model complexity too far will yield a low bias and high variance which results in *overfitting*. Somewhere in-between these extremes, there's an optimal choice of complexity that yields the lowest possible out-of-sample error.

F. Ordinary Least Squares (OLS)

The method of ordinary least squares (OLS), seeks to minimize the L_2-norm $\|\boldsymbol{z} - X\boldsymbol{w}\|_2^2$. Formally, this means we seek a solution vector $\hat{\boldsymbol{w}}_{\text{LS}}$ such that

$$\hat{\boldsymbol{w}}_{\text{LS}} = \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \|\boldsymbol{z} - X\boldsymbol{w}\|_2^2. \tag{16}$$

Minimizing the L_2-norm with respect to the weights can be done in a straight-forward manner:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{w}}\|\boldsymbol{z} - X\boldsymbol{w}\|_2^2 &= \frac{\partial}{\partial \boldsymbol{w}}\left(\boldsymbol{z}^T X\boldsymbol{w} - \boldsymbol{w}^T X^T \boldsymbol{z} + \boldsymbol{w}^T X^T X\boldsymbol{w}\right) \\ &= -2X^T\boldsymbol{z} + 2X^T X\boldsymbol{w} \\ &= 0,\end{aligned} \tag{17}$$

which implies that

$$\hat{\boldsymbol{w}}_{\text{LS}} = \left(X^T X\right)^{-1} X^T \boldsymbol{z}. \tag{18}$$

In the last equation, we've assumed $X^T X$ to be invertible which is usually the case when the number of datapoints n exceeds the number of features p such that $n > p$.

G. Ridge Regression

In this section, we develop the idea of Ridge regression, which adds a *regularization* parameter λ which acts as a penalty on the squared L_2-norm $\|\boldsymbol{w}\|_2^2$ of the solution vector. More formally,

$$\hat{\boldsymbol{w}}_{\text{Ridge}} = \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \left(\|\boldsymbol{z} - X\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2 \right). \tag{19}$$

Again, we minimize the L_2-norm with respect to the weights but, as is obvious, the only difference from the minimization of the L_2-norm in the case of OLS is that we have to add one more term

$$\lambda \frac{\partial}{\partial \boldsymbol{w}} \|\boldsymbol{w}\|_2^2 = 2\lambda\boldsymbol{w}, \tag{20}$$

which added to the terms we got from OLS and solving for $\boldsymbol{w}$ gives us

$$\hat{\boldsymbol{w}}_{\text{Ridge}} = \left(X^T X + \lambda \boldsymbol{I}_{p\times p}\right)^{-1} X^T \boldsymbol{z}. \tag{21}$$

It can be shown that the Ridge predictor, $\hat{\boldsymbol{f}}_{\text{Ridge}}$, is upper-bounded by the OLS predictor,

$$\hat{\boldsymbol{f}}_{\text{Ridge}} = X\hat{\boldsymbol{w}}_{\text{Ridge}} \leq \hat{\boldsymbol{f}}_{\text{LS}}. \tag{22}$$

See [6] for an excellent derivation for the above relation.

H. LASSO regression

Least Absolute Shrinkage and selection operator (LASSO) regression, in similarity with Ridge regression, adds a *regularization* parameter λ. This parameter acts as a penalty on the L_1-norm $\|\boldsymbol{w}\|_1$ of the solution vector instead. Formally, it is defined as

$$\hat{\boldsymbol{w}}_{\text{LASSO}} = \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \left(\|\boldsymbol{z} - X\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_1 \right). \tag{23}$$

We restrict ourselves to using a package for performing LASSO regression provided by SciKit-Learn[4].

I. Resampling techniques

1. *Bootstrapping*

Bootstrapping is a resampling technique used to obtain accuracy measures for sample estimates. Suppose $\mathcal{L}$ is the dataset containing training data with n points. To perform bootstrap analysis of some sample estimate, we sample B so-called Bootstrap samples $\{\mathcal{L}_1^*, \mathcal{L}_2^*, \cdots, \mathcal{L}_B^*\}$ with n points each, sampled from $\mathcal{L}$ *with* replacement. With a large number of samples we generate an empirical sampling distribution for the statistic of interest. Say we want the sample estimate of some quantity θ. With each Bootstrap sample, we compute sample estimates $\{\theta_1^*, \theta_2^*, \cdots, \theta_B^*\}$. Next, we compute the mean value

$$\theta^* = \frac{1}{B}\sum_{i=1}^{B} \theta_i^*, \tag{24}$$

and corresponding standard error

$$\text{Var}(\theta^*) = \frac{1}{B-1}\sum_{i=1}^{B} (\theta_i^* - \theta^*)^2. \tag{25}$$

2. *k-fold Cross-Validation*

Assuming you have a dataset $\mathcal{D}$ and you shuffle it, one can perform k-fold cross-validation by splitting the shuffled data into k-number of groups. First you pick one group and choose this group to be your test dataset, while using the remaining groups as your training dataset.

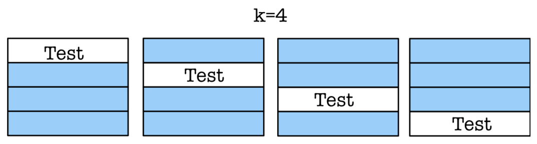

FIG. 1. Illustration of how a dataset is split into k = 4 groups in preparation of a 4-fold Cross-Validation. One uses one group as a test set to train the model, and the remaining groups as training data.

One can now use the training data to fit a model which can later be evaluated on the group you picked as test data. The test score for statistical values of interest, e.g MSE, are stored before you reject the model and pick out one of the other groups as the new test dataset. This is repeated for all of the groups until every group has been used as the test dataset one time, and as a part of the training dataset $k-1$ times. One will in the end have k-values of statistical data and by taking the mean over these be left with a less biased prediction of the performance of the model than one would get by simply splitting the dataset $\mathcal{D}$ into training and test data once.

J. Datasets

In this report we will perform regression analysis on two types of datasets. One being generated from a known two-dimensional function, Franke's function [5]. The second dataset consists of real digital terrain data [7].

1. *Franke's function*

We generate data with Franke's function in accordance to eq. (1), i. e. the ground-truth is now given by Franke's Function with added stochastic noise $\epsilon \sim \mathcal{N}(0, \sigma)$. The Franke function is a weighted sum of four exponentials, and is given by

$$\begin{aligned} f(x,y) = &\frac{3}{4}\exp\left\{\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right)\right\} \\ &+ \frac{3}{4}\exp\left\{\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right)\right\} \\ &+ \frac{1}{2}\exp\left\{\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right)\right\} \\ &- \frac{1}{5}\exp\left\{\left(-(9x-4)^2 - (9y-7)^2\right)\right\}. \end{aligned} \tag{26}$$

We sample n random tuples of $x, y \in [0,1)$, which gives the dataset

$$\mathcal{D} = \{\{(x_1,y_1),...,(x_n,y_n)\}, \{z(x_1,y_1),...,z(x_n,y_n)\}\},$$

where $z(x_i,y_i) = f(x_i,y_i) + \epsilon_i$ for $i = 1,...,n$.

2. *Terrain data*

The digital terrain data is stored as a raster image, which again is said to be a bitmap. A bitmap is a grid of individual pixels that collectively compose an image [2]. Each pixel is coded with a value corresponding to different shades of white.

To fit the data to our model we interpret the degree of shading for each pixel as a function value of some function $z(x,y) = f(x,y) + \epsilon$, where both $f(x,y)$ and ϵ are unknowns. We assume here that $z(x,y)$ describes the height above an unknown reference point. Each pixel corresponds to a coordinate on the grid, so by parametrizing the grid we obtain the (x,y)-tuples needed to construct the design matrix. Due to limited RAM-capacities of the computers used to do the analysis presented later, we extract a smaller image from the full image. The image we perform regression analysis on is shown in figure 2. To extract the same image and corresponding dataset, the code found at [1] produces the correct dataset.

III. ALGORITHMS

The implementation of the various regression methods are rather trivial, in the sense that it is only based on a single matrix inversion, or utilization of a Python module to perform the LASSO regression. Bootstrap analysis is also fairly straight forward to implement based on the preceding discussion and no further treatment of this idea is judged as a necessity. We therefore restrict this section to simply sketch the implementation of the k-fold cross-validation resampling technique.

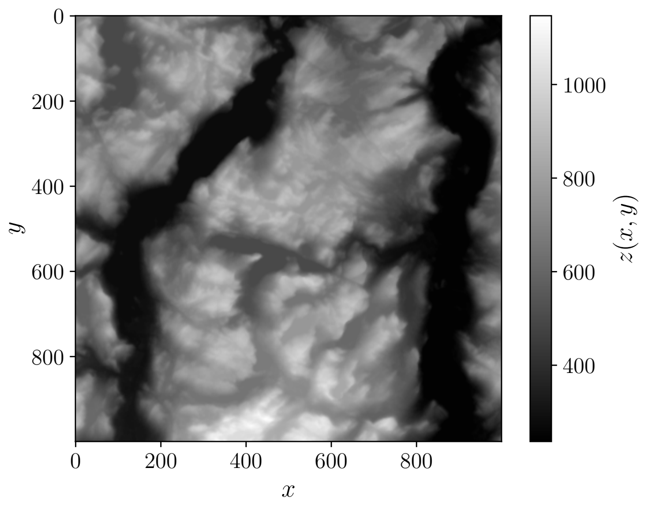

FIG. 2. The image shows a subimage of the full terrain data. The code that produces this particular subimage can be found at [1].

The devised algorithm seeks to avoid unnecessary storage of the data by, say, splitting up the data in several folds and storing them all *simultaneously* in addition to the design matrix since this is intractable for very large datasets. Instead, we devise an algorithm based much on the same idea as compressed row-storage is - although we're not really dealing with a sparse matrix and the only element we borrow from this idea is the *row pointer*. We create this so-called row pointer that encodes information on how to distribute the rows in the design matrix for a given test- and training fold. From this we can extract which rows of the design matrix (recall that each row contains a single datapoint (x_i, y_i) with several features) that each training- and test fold pertain to. We denote this row pointer as r. The following algorithm summarizes the high-level algorithmic idea.

IV. RESULTS

A. Franke's function

All of the results presented in this section are obtained by applying the bootstrapping resampling technique with one hundred bootstrap samples, $B = 100$.

1. *Ordinary least squares*

Figure 3 displays the in- and out-of-sample errors as a function of model complexity for $n = 1000$ and $\sigma = 1.0$.

Algorithm 1 k-fold Cross-Validation

```
Determine size of each fold
$s = \lfloor n/k \rfloor$                          ▷ $k$ number of folds
$\rho = n \mod k$                                  ▷ Remainder
for $i = 1, 2, .., k$ do
    $n_i = s + (\rho > 0)$                         ▷ Size of fold $i$
    $\rho = \rho - 1$
Encode the index in $X$ where a given fold starts
$r_1 = 0$.
for $i = 1, ..., k$ do
    $r_{i+1} = r_i + n_i$        ▷ size of each fold $i$, $n_i = r_{i+1} - r_i$
Perform k-fold cross-validation
for $i = 1, ..., k$ do
    for $j = r_i, ..., r_{i+1}$ do                 ▷ Extract test data
        $X^{\text{test}}_{j,:} = X_{j,:}$
        $z^{\text{test}}_j = z_j$
    for $j = 1, ..., i$ do                         ▷ Extract training data
        for $l = r_j, ..., r_{j+1}$ do
            $X^{\text{train}}_{l,:} = X_{l,:}$
            $z^{\text{train}}_l = z_l$
    for $j = i + 1, ..., k$ do
        for $l = r_j, ..., r_{j+1}$ do
            $X^{\text{train}}_{l,:} = X_{l,:}$
            $z^{\text{train}}_l = z_l$
    Train model
    Predict and compute performance metrics
Compute average of performance metrics
```

Showcasing the two together highlights some of the problems connected to the bias-variance tradeoff. To further investigate the tradeoff, figure 4 shows the out-of-sample error, along with the bias and variance, as a function of model complexity. For comparison against the two remaining regression methods, the results in figure 4 are produced with the noise $\sigma = 0.1$.

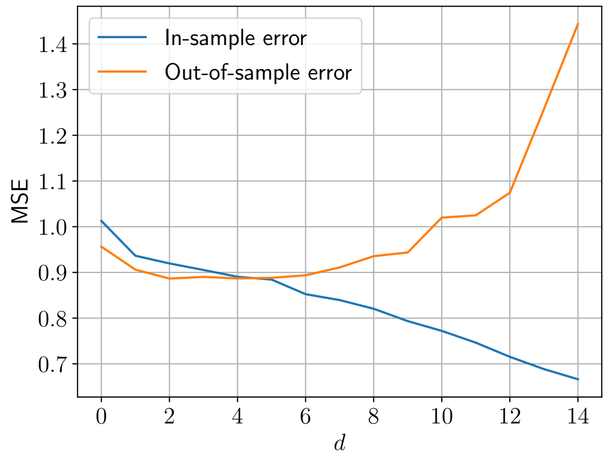

FIG. 3. The figure depicts the in- and out-of-sample errors as a function of model complexity d. Here the OLS regression method is applied with boostrapping to a dataset for Franke's function with $n = 1000$ and $\sigma = 1.0$.

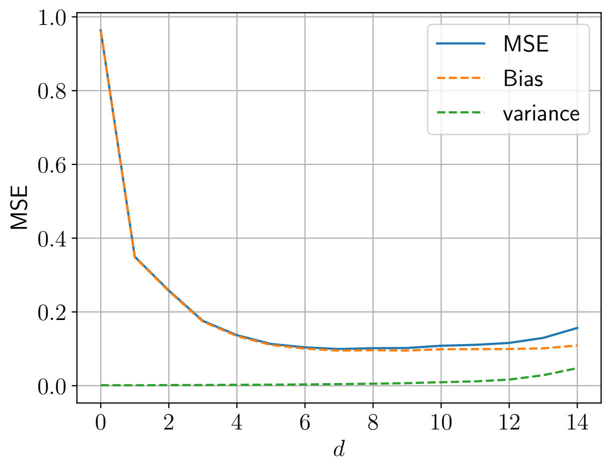

FIG. 4. Displaying the out-of-sample error, bias and variance as a function of polynomial degree. The results are obtained by applying OLS with the bootstrapping resampling technique on the dataset for Franke's function with $n = 1000$ and $\sigma = 0.1$.

The lowest MSE score in figure 4 were estimated to

$$\begin{aligned} &\min \mathrm{MSE}(d) \approx 0.0994 \\ &\arg\min_d \mathrm{MSE}(d) = 7. \end{aligned} \tag{27}$$

2. Ridge regression

Figure 5 depicts the regularization path, with the optimal choice of polynomial degree d and penalty parameter λ, on a dataset of the Franke function with $n = 1000$ and $\sigma = 0.1$.

The minimal MSE in figure 5 is

$$\begin{aligned} &\min \mathrm{MSE}(d, \lambda) \approx 0.0991 \\ &\arg\min_{(d,\lambda)} \mathrm{MSE}(d, \lambda) = (7, 10^{-7}). \end{aligned} \tag{28}$$

3. LASSO regression

The regularization path for LASSO regression in figure 6 are produced from the same dataset as OLS and Ridge.

The optimal values of (d, λ) yields the following MSE

$$\begin{aligned} &\min \mathrm{MSE}(d, \lambda) \approx 0.0984 \\ &\arg\min_{(d,\lambda)} \mathrm{MSE}(d, \lambda) = (10, 10^{-11}). \end{aligned} \tag{29}$$

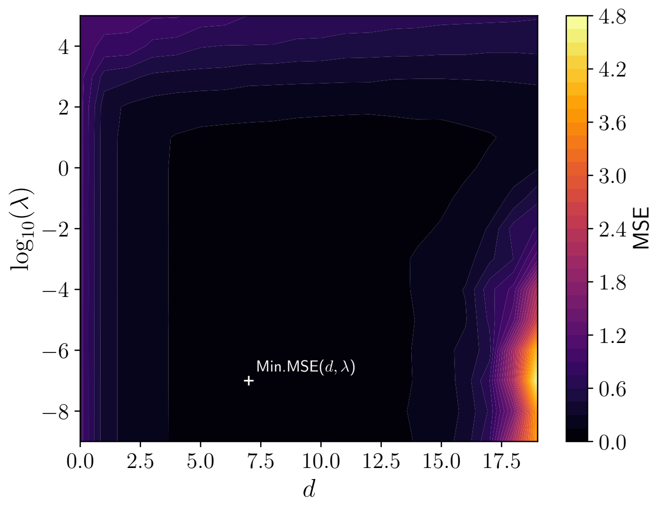

FIG. 5. Regularization path for Ridge regression, as a function of polynomial degree d and penalty parameter λ. Min. MSE$(7, 10^{-7}) \approx 0.0991$, for a dataset with $n = 1000$ and $\sigma = 0.1$.

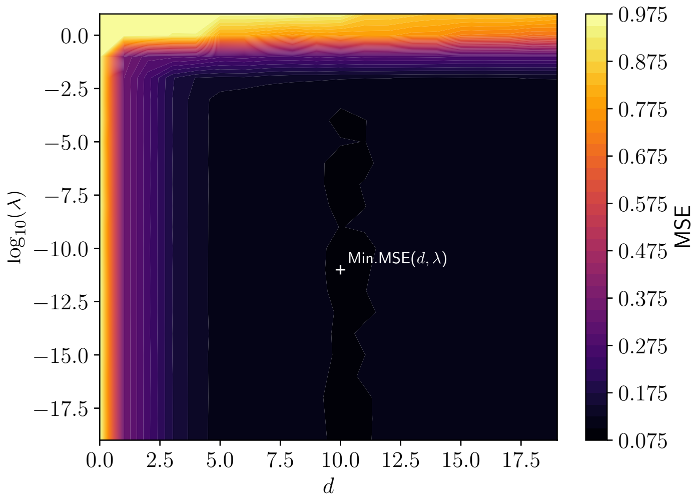

FIG. 6. Regularization path for LASSO regression, as a function of polynomial degree d and penalty parameter λ. Min. MSE$(10, 10^{-11}) \approx 0.0984$, for a dataset with $n = 1000$ and $\sigma = 0.1$.

B. Terrain data

Based on the upper-bound on the Ridge predictor in eq. (22), we deem it unnecessary to perform any further analysis with the OLS method on the terrain data. This is connected to the fact that for sufficently small λ, the estimators are approximately the same, $\boldsymbol{w}_{\text{Ridge}} \approx \boldsymbol{w}_{\text{LS}}$, since $(X^TX + \lambda I_{p\times p})^{-1} \approx (X^TX)^{-1}$.

1. *Ridge regression*

Figure 7 shows the computed regularization path $R^2(d, \lambda)$ used on the terrain data with 10-fold cross-validation on the full dataset. The maximal R^2-score and the corresponding parameters were estimated to

$$\max R^2(d,\lambda) \approx 0.89$$
$$\arg\max_{(d,\lambda)} R^2(d,\lambda) = (22, 10^{-4}). \tag{30}$$

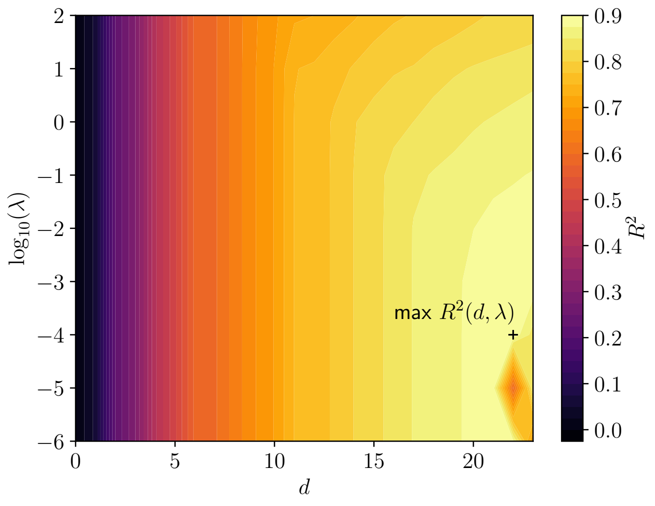

FIG. 7. The figure displays the regularization path using the R^2-score as performance metric, as a function of the polynomial degree d and regularization parameter $\log_{10}(\lambda)$. The R^2-score was averaged using 10-fold cross-validation on terrain data shown in figure 2. The best-performance was found to be $\max R^2 \approx 0.89$ for parameters $(d, \log_{10}(\lambda)) = (22, -4)$.

2. *LASSO regression*

Figure 8 shows the computed regularization path $R^2(d, \lambda)$ used on the terrain data with 10-fold cross-validation on the full dataset.

As we can see from the figure, the R^2-score did not converge to a minimum for the chosen polynomial degrees.

V. DISCUSSION

A. Bias-Variance trade-off in Franke's function with OLS

The in- and out-of-sample errors in figure 3 are produced from a dataset with a higher noise, $\sigma = 1.0$, to

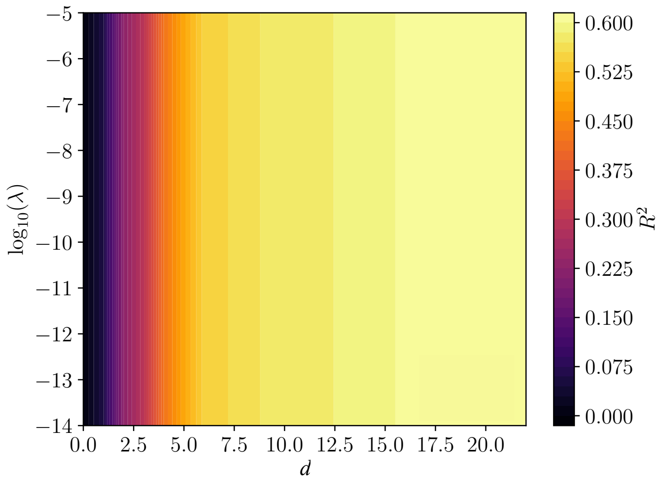

FIG. 8. The figure displays the regularization path using the R^2-score as performance metric, as a function of the polynomial degree d and regularization parameter $\log_{10}(\lambda)$. The R^2-score was averaged using 10-fold cross-validation on terrain data shown in figure 2. Due to high time consumption we were not able to obtain a general best-performance result.

demonstrate how systematic noise affects the fitting and prediction procedures. A large discrepancy between the two errors indicates that the chosen model overfits the data. A high complexity model can recognize global trends within the dataset and possible noise-generated patterns, which results in a model that is specialized on the training set. Applying the same model to the test set will then give a high out-of-sample error due to high variance in the model, which is the behaviour we observe in figure 3. A model of lower polynomial degree can thus have better predictive performance given that the dependence on the dataset is lower. A less complex model will though have a higher bias.

The behaviour shown in figure 4 confirms our observations from figure 3, which again is in agreement with the theory presented in the preceding section. As the polynomial degree increases, the bias decreases while the variance increases. We see that the MSE score first follows the trend of the bias but switches to mimicking the trend of the variance after reaching its minimum value. All of which is consistent with the terms in eq. (15).

B. General comparison of regression methods for Franke's function

From figure 6 it is evident that the lowest out-of-sample error for LASSO regression is ≈ 0.0984, which exceeds the value obtained from OLS. LASSO regression reduces to OLS when $\lambda = 0$, and since the optimal choice of λ for LASSO regression in this specific case is 10^{-11}, we deduce it is unnecessary to perform LASSO regression on data from the Franke function. OLS provides sufficiently good results, and even though it is not emphasized in this report we note that the LASSO regression method has a significantly higher time consumption than the two other methods. This is due to the fact that the process of determine $\hat{\boldsymbol{w}}_{\text{LASSO}}$ is not simply achieved through a single matrix inversion.

For the specific dataset with $n = 1000$ and $\sigma = 0.1$ the lowest MSE score is produced with Ridge regression. Again we observe the expected behaviour from the MSE in figure 5, where it start to increase with higher values of d.

C. Terrain data and general remarks

From the regularization path shown in figure 7, we found the maximal R^2-score to be roughly $R^2 \approx 0.89$, implying the model reproduces the test data fairly well. Although the Ridge-solver could handle the large dataset at a feasible computation time, the LASSO-solver requires far more computation to converge and gave a certain trade-off with respect to accuracy. We had to sacrifice the number of iterations the Scikit-Learn module would perform to minimize the error, and the result shown in figure 8 is mediocre, at best, with the R^2-score peaking at roughly $R^2 \approx 0.61$. We realize this mistake, and a much better application of the two methods would be to train the models on smaller patches of the image so that training could be sped up significantly and a lower complexity on each patch would be necessary.

The optimal parameters are much larger on this dataset than what was found in the case of the smaller dataset produced by Franke's function. This is consistent with bias-variance trade-off and in general the optimal parameters are highly data-dependent, especially of the size of the dataset. This means that the regression model's ability to generalize from one dataset to the next is not all that good. As a final point, the way we've performed 10-fold cross-validation has in a sense made use of the test-data to fit the parameters to find the optimal parameter set. This is in turn results in a biased model, and the results themselves gives us a weak idea of the model's ability to generalize. A better way to fit the parameters, would be to split the dataset into a training- and test set, and then perform k-fold cross-validation on the training set, using so-called validation sets to fit the hyperparameters and *then* test the model's ability to generalize on the test set.

VI. CONCLUSION

The results produced are *highly* data-dependent. It is almost impossible to draw any objective conclusions regarding the different statistical values collected from the various methods, as they heavily depend on the size and noise of the dataset the models are trained and tested on. We fitted the hyperparameters using the test-data, which resulted in a biased model and its ability to generalize remain uncertain. Although not quantified properly, it seems that training on one finite dataset does not produce a model that easily predict data from another finite dataset. This appears to be a general weakness with all three models. The maximally obtained R^2-scores and minimal MSE-values found in this article points to Ridge as the superior model, but this claim should be called into question due to the limited knowledge of their generalizability and be further tested.

VII. APPENDIX

A. Analytical expression for mean square error

The cost-function is defined as

$$\mathcal{C}(\boldsymbol{z}, \hat{f}(X)) = \sum_i (z_i - f(\mathbf{x}_i))^2. \tag{31}$$

The expected error may be decomposed as follows.

$$\mathbb{E}[\mathcal{C}(\boldsymbol{z}, \hat{f}(X))] = \sum_{i=1}^{n} \left(f_i - \mathbb{E}[\hat{f}_i]\right)^2 + \sum_{i=1}^{n} \mathbb{E}\left[\left(\hat{f}_i - \mathbb{E}[\hat{f}_i]\right)^2\right] + \sum_{i=1}^{n} \sigma^2 \tag{32}$$

where $z_i \equiv z(\boldsymbol{x}_i)$ and $\hat{f}_i \equiv \hat{f}(\boldsymbol{x}_i)$. We start from the general expression of the mean square error using vector notation. Recall that $z(\boldsymbol{x}) = f(\boldsymbol{x}) + \epsilon$, with $\epsilon \sim \mathcal{N}(0, \sigma)$

$$\begin{aligned}\mathbb{E}\left[(\boldsymbol{z} - \hat{\boldsymbol{f}})^2\right] &= \mathbb{E}\left[(\boldsymbol{f} + \epsilon - \hat{\boldsymbol{f}})^2\right], \\ &= \mathbb{E}\left[(\boldsymbol{f} + \epsilon - \hat{\boldsymbol{f}} + \mathbb{E}[\hat{\boldsymbol{f}}] - \mathbb{E}[\hat{\boldsymbol{f}}])^2\right].\end{aligned}$$

Applying a good deal of algebra to the expression inside the brackets we end up with the relation

$$\begin{aligned}\mathbb{E}\left[(\boldsymbol{z} - \hat{\boldsymbol{f}})^2\right] &= \mathbb{E}\left[(\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}])^2 + \epsilon^2 + (\mathbb{E}[\hat{\boldsymbol{f}}] - \hat{\boldsymbol{f}})^2 + 2(\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}])\epsilon + 2(\mathbb{E}[\hat{\boldsymbol{f}}] - \hat{\boldsymbol{f}})\epsilon + 2(\mathbb{E}[\hat{\boldsymbol{f}}] - \hat{\boldsymbol{f}})(\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}])\right], \\ &= \mathbb{E}\left[(\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}])^2\right] + \mathbb{E}\left[\epsilon^2\right] + \mathbb{E}\left[(\mathbb{E}[\hat{\boldsymbol{f}}] - \hat{\boldsymbol{f}})^2\right] + 2(\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}])\underbrace{\mathbb{E}[\epsilon]}_{=0} + 2\mathbb{E}[\mathbb{E}[\hat{\boldsymbol{f}}] - \hat{\boldsymbol{f}}]\underbrace{\mathbb{E}[\epsilon]}_{=0} + 2\underbrace{\mathbb{E}[\mathbb{E}[\hat{\boldsymbol{f}}] - \hat{\boldsymbol{f}}]}_{=0}(\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}]) \\ &= \mathbb{E}\left[(\boldsymbol{f} - \mathbb{E}[\hat{\boldsymbol{f}}])^2\right] + \mathbb{E}\left[(\hat{\boldsymbol{f}} - \mathbb{E}[\hat{\boldsymbol{f}}])^2\right] + \mathbb{E}[\epsilon^2] \\ &= \sum_{i=1}^{n} \left(f_i - \mathbb{E}[\hat{f}_i]\right)^2 + \sum_{i=1}^{n} \mathbb{E}\left[\left(\hat{f}_i - \mathbb{E}[\hat{f}_i]\right)^2\right] + \sum_{i=1}^{n} \sigma^2.\end{aligned}$$

where the expectation values with respect to the dataset and noise are assumed to be independent.

[1] Code that produces the correct subimage of the full dataset. https://github.com/reneaas/fys-stk4155/blob/master/project1/codes/terrain.py (02.10.20).

[2] Lesson 5. about the geotiff (.tif) raster file format: Raster data in python. https://www.earthdatascience.org/courses/use-data-open-source-python/intro-raster-data-python/fundamentals-raster-data/intro-to-the-geotiff-file-format/ (23.09.20).

[3] Our github repository containing all code produced for and used in this report. https://github.com/reneaas/fys-stk4155/tree/master/project1 (29.09.20).

[4] Scikit-learn, linear model lasso. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html (22.09.20).

[5] Richard Franke. A critical comparison of some methods for interpolation of scattered data. 1979.

[6] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre Day, Clint Richardson, Charles Fisher, and David Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Physics Reports*, 810:23, 03 2018.

[7] USGS U.S. Geological Survey. https://github.com/CompPhysics/MachineLearning/tree/master/doc/Projects/2020/Project1/DataFiles (23.09.20).