Regression and Resampling Techniques Applied to Franke's Function and Terrain Data

René Ask, Kaspara Skovli Gåsvær & Maria Linea Horgen
(Dated: October 5, 2020)

We implement two linear regression methods: ordinary least squares (OLS) and Ridge regression. In addition LASSO regression is studied through scikit-learn's LASSO module. We take a quick detour to study the bias-variance trade-off problem as well. Resampling techniques such as k-fold cross-validation and bootstrapping is presented. The methods are compared and contrasted on two datasets: one generated by Franke's function and one extracted from a terrain image. We find that the models generalize well on data generated by the Franke's function which we reason is because the datasets used for training and test both approximately encode the same data distribution. Ridge and LASSO are applied to the terrain data and we find the generalize poorly due to a small sample size of the training data which resulted in a largely different data distribution between the training and test data.

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 can 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 error 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

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

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 training data can again be partitioned into a validation set with the remaining data being used as training data. The validation set gives an unbiased evaluation of the results from the training set, and hence can be used for determining hyperparameters.

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

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

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.

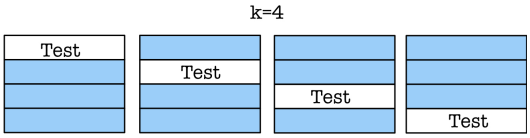

FIG. 1. Illustration of how a dataset is split into k = 4 groups in preparation of a 4-fold Cross-Validation. One uses one group as a test set to train the model, and the remaining groups as training data.

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 two smaller images from the full image, both containing 1024 pixels. To extract the images and corresponding datasets, 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.

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. Algorithm 1 summarizes the high-level algorithmic idea.

Algorithm 1 k-fold Cross-Validation

```
Determine size of each fold
s = ⌊n/k⌋                                   ▷ k number of folds
ρ = n  mod k                                ▷ Remainder
for i = 1, 2, .., k do
    n_i = s + (ρ > 0)                       ▷ Size of fold i
    ρ = ρ − 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^test_{j,:} = X_{j,:}
        z^test_j = z_j
    for j = 1, ..., i do                    ▷ Extract training data
        for l = r_j, ..., r_{j+1} do
            X^train_{l,:} = X_{l,:}
            z^train_l = z_l
    for j = i + 1, ..., k do
        for l = r_j, ..., r_{j+1} do
            X^train_{l,:} = X_{l,:}
            z^train_l = z_l
    Train model
    Predict and compute performance metrics
Compute average of performance metrics
```

IV. RESULTS

A. Franke's function

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

1. *Ordinary least squares*

Figure 2 displays the in- and out-of-sample errors as a function of model complexity for $n = 1000$ and $\sigma = 1.0$. Showcasing the two together highlights some of the problems connected to the bias-variance tradeoff. To further investigate the tradeoff, figure 3 shows the out-of-sample error, along with the bias and variance, as a function of model complexity on a validation set. For comparison against the two remaining regression methods, the results in figure 3 are produced with the noise $\sigma = 0.1$.

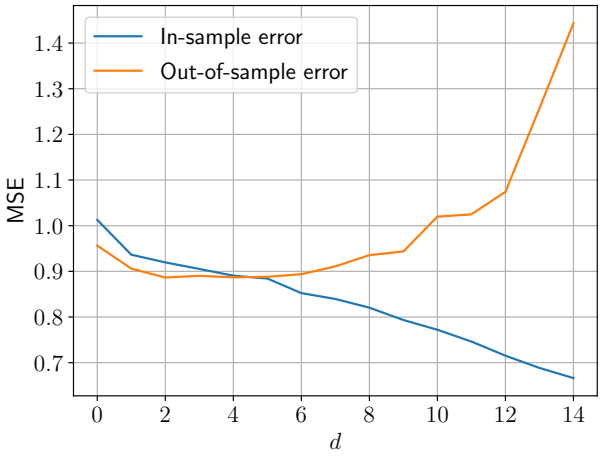

FIG. 2. The figure depicts the in- and out-of-sample errors as a function of model complexity d. Here the OLS regression method is applied with 100 boostrapping samples to a dataset for Franke's function with $n = 1000$ and $\sigma = 1.0$.

From figure 3, the optimal polynomial degree and lowest MSE obtained from the validation set was

$$\min \text{MSE}(d) \approx 0.099, \qquad \arg\min_d \text{MSE}(d) = 7. \tag{27}$$

Performing OLS with 100 boostrapping samples on a unseen test dataset with the polynomial degree obtained from the validation set, $d = 7$, we get the following MSE-score with standard error

$$\text{MSE}(7) = 0.112 \pm 0.009. \tag{28}$$

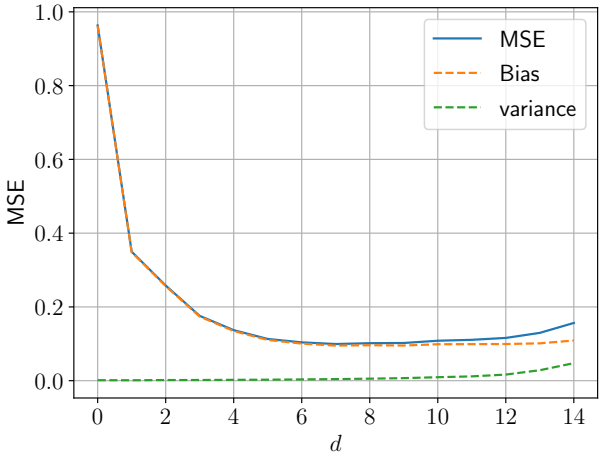

FIG. 3. Displaying the out-of-sample error, bias and variance as a function of polynomial degree. The results are obtained by applying OLS with the bootstrapping resampling technique on a validation set for Franke's function with $n = 1000$ and $\sigma = 0.1$.

2. *Ridge regression*

Figure 4 depicts the regularization path, with the optimal choice of polynomial degree d and penalty parameter λ, on a validation set of the Franke function with $n = 1000$ and $\sigma = 0.1$.

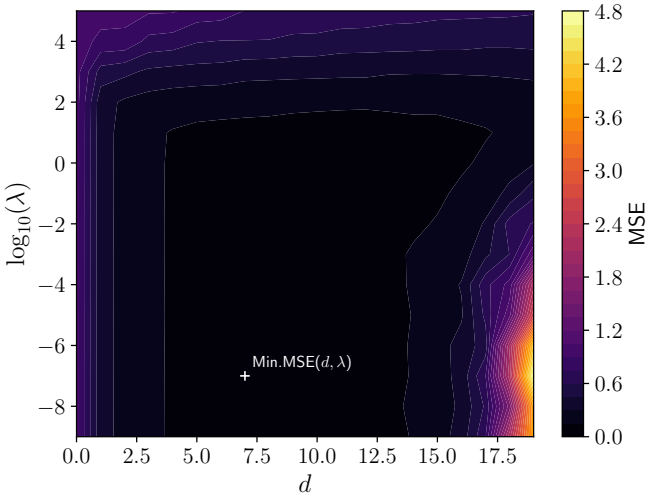

FIG. 4. Regularization path for Ridge regression, as a function of polynomial degree d and penalty parameter λ. The regularization path was computed with the bootstrap resampling technique for $B = 100$ on a validation set of the Franke Function. Min. $\text{MSE}(7, 10^{-7}) \approx 0.0991$, for a dataset with $n = 1000$ and $\sigma = 0.1$.

The minimal MSE in figure 4 is

$$\min \text{MSE}(d,\lambda) \approx 0.099$$
$$\arg\min_{(d,\lambda)} \text{MSE}(d,\lambda) = (7, 10^{-7}). \tag{29}$$

Using the optimal parameters on the same test set as the one we used with OLS, but now with Ridge regression and bootstrapping, yields the MSE-score with standard error

$$\text{MSE}(7, 10^{-7}) = 0.112 \pm 0.009. \tag{30}$$

3. *LASSO regression*

The regularization path for LASSO regression in figure 5 are produced from the same validation set as OLS and Ridge.

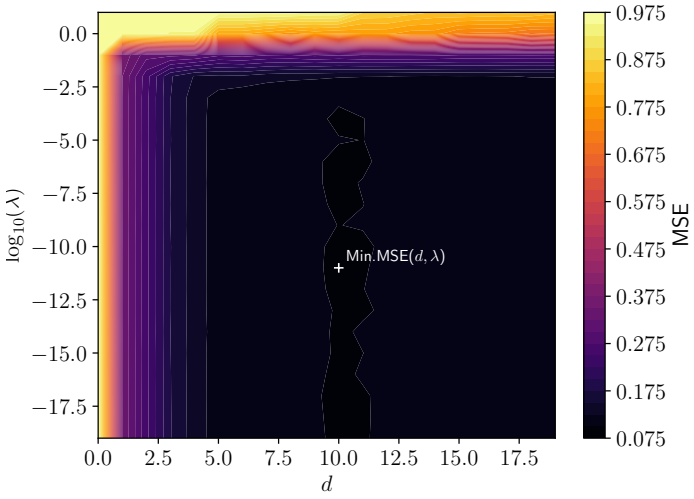

FIG. 5. Regularization path for LASSO regression, as a function of polynomial degree d and penalty parameter λ. The regularization path was computed with the bootstrap resampling technique for $B = 10$ on a validation set. Min. $\text{MSE}(10, 10^{-11}) \approx 0.0984$, for a dataset with $n = 1000$ and $\sigma = 0.1$.

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

$$\min \text{MSE}(d,\lambda) \approx 0.098$$
$$\arg\min_{(d,\lambda)} \text{MSE}(d,\lambda) = (10, 10^{-11}). \tag{31}$$

Performing LASSO regression on a test set with the parameters above, we obtain the following MSE-score

$$\text{MSE}(10, 10^{-11}) = 0.123 \pm 0.015. \tag{32}$$

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 $(X^T X + \lambda I_{p\times p})^{-1} \approx (X^T X)^{-1}$.

1. *Ridge regression*

Figure 6 shows the computed regularization path $\text{MSE}(d,\lambda)$ used on the terrain data with 10-fold cross-validation on a small patch of the terrain image. The minimal validation error and the corresponding parameters were estimated to

$$\min \text{MSE}(d,\lambda) \approx 0.89$$
$$\arg\min_{(d,\lambda)} \text{MSE}(d,\lambda) = (18, 10^{-4}). \tag{33}$$

The optimal parameters found from the regularization path in figure 6 was used to train the model on the same patch of the image and tested on a random different patch of the terrain image. The resulting MSE and its standard error obtained through using $B = 1000$ bootstrap samples was

$$\text{MSE}(18, 10^{-4}) = 3.11 \pm 0.03 \tag{34}$$

2. *LASSO regression*

Figure 7 shows the computed regularization path $\text{MSE}(d,\lambda)$ used on the terrain data with 10-fold cross-validation on a small patch of the terrain image to find the optimal parameters. From the figure, we can read off that $d = 19$ and $\lambda = 10^{-4}$ gave the average validation error of MSE ≈ 0.018. Using these parameters to train the model on the same patch of the image with $B = 1000$ bootstrap samples and testing on a random different patch of the image gave the average out-of-sample error

$$\text{MSE}(19, 10^{-4}) = 3.01 \pm 0.02 \tag{35}$$

V. DISCUSSION

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

The in- and out-of-sample errors in figure 2 are produced from a dataset with a higher noise, $\sigma = 1.0$, to

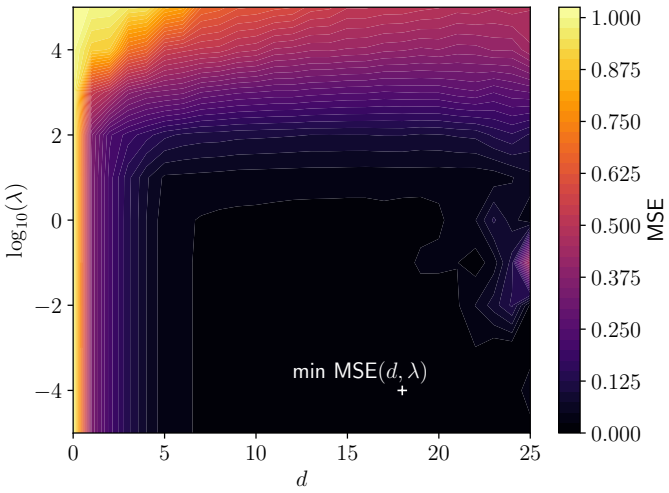

FIG. 6. The figure shows the regularization path using Ridge regression on terrain data with MSE as the performance metric. It was computed using 10-fold cross-validation and the MSE values shown here are the average validation error. The optimal parameters can be read off as $d = 18$ and $\lambda = 10^{-4}$ with the lowest validation error found to be MSE ≈ 0.0013.

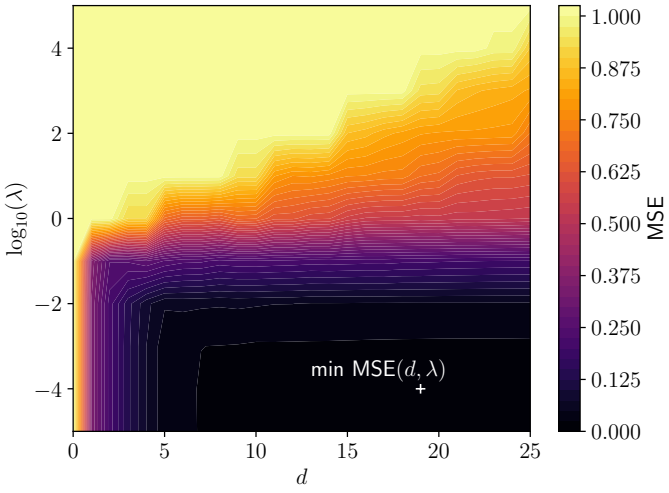

FIG. 7. The figure shows the regularization path using Ridge regression on terrain data with MSE as the performance metric. It was computed using 10-fold cross-validation and the MSE values shown here are the average validation error. The optimal parameters can be read off as $d = 19$ and $\lambda = 10^{-4}$, with the lowest error MSE ≈ 0.018.

demonstrate how 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 2. 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 3 confirms our observations from figure 2, 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

The lowest out-of-sample error for the three regression methods is obtained from both OLS and Ridge regression, with a MSE-score of 0.112 ± 0.009. From the theory we know that Ridge is bounded by OLS, so the result is in agreement with eq. (22). Comparing the mean square errors from the validation- and test sets are not straight forward given that they are both only a estimate of the true unknown MSE value. Looking at the relative error between the two, with respect to the out-of-sample error from the test set, can still give us some insight regarding overfitting of our model. The relative error for Ridge and OLS, since they have the same MSE values for both validation- and test-set error, is 0.116. Given the low relative error we assume that our model is in fact not overfitted and thereby generalizes well to unseen data produced by Franke's function. This is natural, though, since the the datasets generated with the Franke's function necessarily all have the same underlying data distribution, and thus training on one whilst avoiding overfitting will necessarily mean the model generalize well to unseen points from the same distribution.

We observe that LASSO regression produces a slightly higher MSE value for the test dataset compared to the remaining two regression methods. LASSO regression reduces to OLS when $\lambda = 0$, with the optimal choice of λ being 10^{-11} in this specific case, the higher MSE value can be explained from the high polynomial degree. From the validation set the optimal choice of d is 10, and from the bias-variance discussion this will give rise to a higher variance. We do note that the low value of λ accompanied by a relatively high polynomial degree d is somewhat counter-intuitive, given that λ is a penalty parameter for higher complexity.

C. Terrain data and general remarks

We used the regularization paths shown in figure 6 and figure 7 to fit the parameters d and λ. Both paths yielded models of roughly the same complexity. The Ridge solver produced an average out-of-sample error $\text{MSE} = 3.11 \pm 0.03$ which indicates a low generalizability. Similarly, the Lasso solver yielded $\text{MSE} = 3.01 \pm 0.02$. This too indicates a low generalizability of the model and suggests that the model may be overfitted to the training data. The complexity of both models is pretty high and the size of the dataset is relatively small which is consistent with this interpretation. Another point to make is that the regression methods studied here are basicly polynomial fitting, and thus are useful methods to fit specific terrains, but not to predict unseen terrains. If the models are trained on data that all come from the same distribution of data, the models will generalize well as long as overfitting is avoided. It's plausible that the patch of image used as test data does not share a common distribution of data with the patch used in training, and may also be a reason for the poor generalization shown. To rule out this explanation, one could train the model on a much larger patch of the image in order to capture a larger portion of the *true distribution* the image encodes. Doing this is time consuming and we suggest this could be a fruitful pursuit to further understand the generalizabilities of the regression methods.

VI. CONCLUSION

The results produced are *highly* data-dependent. Resulting models are dependent on the size of the datasets and on the specific dataset studied. The three methods generalize well to unseen data produced with Franke's function, but from the performance on the terrain data, it's clear that the methods' generalizability are mediocre when trained on a small dataset. This is consistent with the fact that the regression methods studied here are akin to polynomial interpolation and fitting a specific terrain does not mean the model will automatically generalize well to an arbitrary terrain.

VII. APPENDIX

A. Analytical expression for mean square error

The cost-function is defined as

$$\mathcal{C}(\boldsymbol{z}, \hat{f}(X)) = \sum_i (z_i - f(\mathbf{x}_i))^2. \tag{36}$$

The expected error may be decomposed as follows.

$$\mathbb{E}[\mathcal{C}(\boldsymbol{z}, \hat{f}(X))] = \sum_{i=1}^{n} \left(f_i - \mathbb{E}[\hat{f}_i] \right)^2 + \sum_{i=1}^{n} \mathbb{E}\left[\left(\hat{f}_i - \mathbb{E}[\hat{f}_i] \right)^2 \right] + \sum_{i=1}^{n} \sigma^2 \tag{37}$$

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

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

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

---

[1] Code that produces the correct subimages 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).