Analyzing the Performance of Linear Regression on the Franke Function and Real-World Terrain Data

Carl Fredrik Nordbø Knutsen, Halvor Tyseng, Benedict Leander Skålevik Parton

September 2022

Linear regression is a powerful tool for approximating functions. In this report, we test the performance of linear regression when approximating the Franke function on the interval $[0, 1] \times [0, 1]$. We approximate the function using Ordinary Least Squares (OLS) regression, and compare with results from ridge and lasso regression to quantify the effect of L^2 and L^1 regularization. We employ bootstrapping to estimate bias and variance, and use cross-validation to compute estimates for the mean squared error (MSE). We find that ridge regression performs slightly better than OLS regression, and significantly better than lasso regression on the Franke funtion data, yielding an optimal model with polynomial degree 16 and $\lambda = 1.456 \cdot 10^{-6}$. We then perform the same analysis on real world terrain data from Saarland, Germany. Here we find the same pattern, namely that Ridge regression performs better than OLS and lasso regression, yielding an optimal model with polynomial degree 5 and $\lambda = 8.685 \cdot 10^{-8}$. A key observation we make, is that ridge regression tends to perform better than OLS for higher polynomial degrees.

1 Introduction

When measuring real-life data, we are seeking to discover some underlying relation between the different variables, so that we can later make predictions based on new data. However, with measurement might come random noise, and making more measurements may be costly or time-consuming, so uncovering these internal relations could prove difficult - this is where linear regression comes in. Linear regression is a collection of methods to approximate the underlying relations between variables with a given set of data.

In this report, we aim to quantify the performance of linear regression. To do this, we will consider polynomial functions for approximating terrain. First we will approximate the Franke function, a smooth function that qualitatively looks like smoothened terrain. After examining this idealized case, we will analyze terrain data from Saarland, Germany. Our goal with this report is to compare the performance of linear regression when applied to an idealized case with its performance on real-world data.

We will test the performance of ordinary least squares (OLS), ridge, and lasso regression when approximating the two terrain functions. We will also evaluate the performance of these regression techniques using resampling methods. In particular, we will use bootstrapping and cross-validation to generate error estimates.

We will first present the methods used in our analysis in section 2. We briefly introduce the regression estimators and resampling methods used, discuss the data sets we use, and data scaling. Then, we will present our findings from the Franke function data in section 3.1, and the findings from the Saarland terrain data in section 3.2. We then discuss our results in section 4, with our main finding being that ridge regression performs the best. Finally, we summarize our findings in section 5.

2 Methods

Here, we will derive some important results about linear regression, and in particular the least squares approximation. Firstly, we make the assumption that there exists some continuous function $f(\boldsymbol{x})$ such that our data $\boldsymbol{y}$ is given by

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

where $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$. We then then assume that the function f can be approximated by a linear function $\tilde{y}$, where

$$\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}. \tag{2}$$

2.1 Ordinary Least Squares Regression

When using OLS regression, we minimize the sum of squared errors $\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}\|_2^2$, which is proportional to the MSE. Computing derivatives, it follows that the $\hat{\boldsymbol{\beta}}$ that minimizes this cost is given by

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

We call $\hat{\boldsymbol{\beta}}$ the OLS estimator for $\boldsymbol{\beta}$. In practice, we calculate the Moore-Penrose inverse of $\boldsymbol{X}^T\boldsymbol{X}$ instead of the ordinary inverse. Then we can compute estimates for singular matrices $\boldsymbol{X}^T\boldsymbol{X}$, and we achieve improved numerical stability. We have that

$$\mathbb{E}(\hat{\boldsymbol{\beta}}) = \boldsymbol{\beta}, \tag{4}$$

meaning that $\hat{\boldsymbol{\beta}}$ is an unbiased estimator. We also have that the variance of $\hat{\boldsymbol{\beta}}$ is given by

$$\text{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}, \tag{5}$$

the proof for these two statements can be found in appendix B. The unbiasedness of $\hat{\boldsymbol{\beta}}$ turns out to be a useful property.

2.2 Ridge Regression

Ridge regression is largely similar to OLS regresssion, however we add an L^2 regularization term $\lambda\|\boldsymbol{\beta}\|_2^2$ to the cost function. That is, we minimize

$$\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 \tag{6}$$

instead. The regularization term forces the solution $\boldsymbol{\beta}$ to be simpler in some sense, working to prevent overfitting. However, this comes at the cost of the estimator being biased. The ridge estimator for $\boldsymbol{\beta}$ is given by

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})\boldsymbol{X}^T\boldsymbol{y}, \tag{7}$$

where $\boldsymbol{I}$ is an identity matrix.

2.3 Lasso Regression

Similarly to ridge regression, lasso regression also adds a regularization term to the cost function. However, lasso regression instead uses L^1 regularization. Then the cost that we minimize becomes

$$\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1. \tag{8}$$

The lasso estimator $\hat{\boldsymbol{\beta}}$ does not have a nice analytic solution, unlike the OLS and ridge estimators. Therefore it is standard practice to estimate the lasso $\boldsymbol{\beta}$ estimate using numerical methods.

2.4 The Bias-Variance Trade-off

Performing any statistical analysis, it is essential to keep the Bias-Variance trade-off in mind. We can always decompose the MSE into

$$\mathbb{E}((\boldsymbol{y} - \tilde{\boldsymbol{y}})^2) = \text{Bias}(\tilde{\boldsymbol{y}})^2 + \text{Var}(\tilde{f}) + \sigma^2, \tag{9}$$

where

$$\text{Bias}(\tilde{\boldsymbol{y}})^2 = (\boldsymbol{y} - \mathbb{E}(\tilde{\boldsymbol{y}}))^2 \tag{10}$$

and

$$\text{Var}(\tilde{f}) = \frac{1}{n}\sum_i(\tilde{y}_i - \mathbb{E}(\tilde{\boldsymbol{y}}))^2. \tag{11}$$

The derivation of equation 9 can be found in appendix C. In essence, this decomposition shows us that one needs to find a model that minimizes the error caused by both bias and variance. Often, minimizing bias entails finding a model with sufficiently high complexity so that it can accurately approximate the original function. On the other hand, one can often reduce variance by restricting the model complexity so that the parameter estimates are not excessively influenced by the error terms ϵ_i. In turn, the bias-variance trade-off is useful to keep in mind when choosing one's model.

2.5 Resampling and Evaluation Methods

When analyzing data, we are not only interested in making predictions. We would also like to assess the accuracy of our estimates. To make the most of our limited data, we employ resampling methods.

2.5.1 Mean Squared Error - MSE

The Mean Squared Error (MSE) is a commonly used measure of error in a model; it calculates the average of the squares of the difference between the estimated and real values at all points; namely, it is given by:

$$MSE(\boldsymbol{z}, \tilde{\boldsymbol{z}}) = \frac{1}{n} \sum_{i=1}^{n} (z_i - \tilde{z}_i)^2 \tag{12}$$

for n datapoints, where $\boldsymbol{z}$ are the real values and $\tilde{\boldsymbol{z}}$ are the estimated values. MSE is minimal at 0, which represents a perfect fit. The Mean Squared Error is the error function which determines the OLS-regression.

2.5.2 R-Squared - R2

R2 is a function that measures how well the model is likely to estimate new data. Its has a maximum value of 1, which occurs when the model perfectly fits the data, and a lower R2-score means a worse fit. The R2-function is given by:

$$R^2(\boldsymbol{z}, \tilde{\boldsymbol{z}}) = 1 - \frac{\sum_{i=0}^{n-1} (z_i - \tilde{z}_i)^2}{\sum_{i=0}^{n-1} (z - \overline{z})} \tag{13}$$

for n datapoints, where $\boldsymbol{z}$ are the real values, $\tilde{\boldsymbol{z}}$ are the estimated values, and $\overline{z}$ is the mean of $\boldsymbol{z}$.

2.5.3 The Bootstrap

Bootstrapping is a method for approximating the distribution of an estimator. Both parametric and nonparametric bootstrapping are useful tools, though we will restrict ourselves to only discussing nonparametric bootstrapping in this report. The idea is to sample data points with replacement from the empirical distribution of pairs (x_i, y_i) and reestimate $\boldsymbol{\beta}$, as shown in algorithm 1. The approximated distribution for $\boldsymbol{\beta}$ can be used to calculate useful quantities such as the variance of the estimator.

Algorithm 1 The Bootstrap

```
procedure BOOTSTRAP(x, y, B)
    n ← len(x)
    initialize array distribution[n, B]
    for b = 1, 2, ... B do
        initialize arrays x_b[n], y_b[n]
        for i = 1, 2, ..., n do
            j ← random number in {1, ... n}
            x_b[i] ← x[j]
            y_b[i] ← y[j]
        end for
        distribution[:, b] ← regression parameters estimated from x_b and y_b
    end for
    return distribution
end procedure
```

2.5.4 Cross-validation

Cross-validation is another resampling method that we will employ. In particular, we have used k-fold cross-validation. This means that we partition our data set into k groups, train our model on data from $k-1$ of the groups, and use the remaining group as a test set. By repeating this process k times, each time using the i'th group as a test set, we can again approximate the distribution of $\boldsymbol{\beta}$. With the cross-validation method we prohibit the MSE evaluation and model creation to be exposed to the randomness accruing in the test-train split. The test-train-split function could possible by chance, split the data such that the they do not look alike at all.

Algorithm 2 K-fold Cross-validation

```
procedure CROSS_VALIDATION(X, z, k_fold, regression_model)
    MSE_train = MSE_test = empty array of len(k_fold)
    X, z = array_split(X, k_fold), array_split(z, k_fold)
    for i = 1, 2, ..., K_fold do
        X_test, X_train = X[i], z[i]
        X_train = X without X[i]
        z_train = z without z[i]
        beta = regression_model(X_train, z_train)
        ztilde_train, ztilde_test = X_train@beta, X_test@beta
        MSE[i] = MSE(ztilde, z)
    end for
    return mean(MSE_train), mean(MSE_test)
end procedure
```

Imagine for example that 25% of the highest points out off our terrain data-points was only included in the test dataset. Our model would then consistently guess too low. Cross-validation prevents this by considering all the data as test-data in different groups.

This is especially effective when the data-sets are small. We can then obtain a MSE from cross-validation, and later use all the data-points in construction of the model.

We will utilize cross-validation to perform a parameter grid search. This means that we estimate the MSE for different parameter values, to find the parameter choice that leads to the lowest error.

2.6 Data analysis

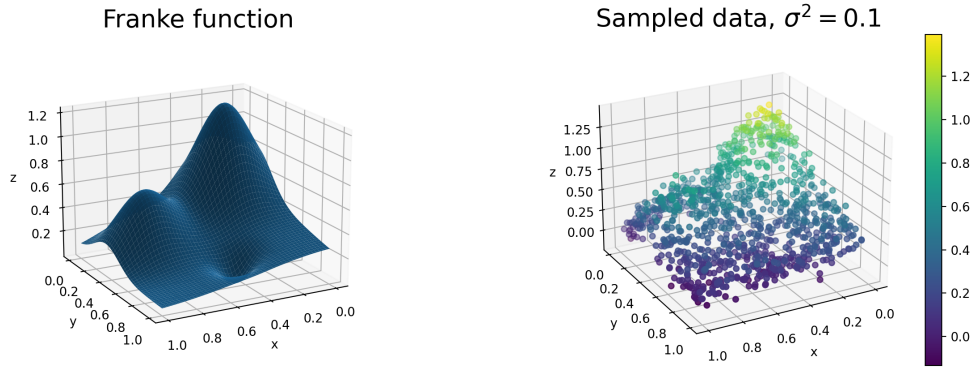
We have analyzed two data sets in this report. First, we have generated synthetic data using the Franke function, to study the performance of our regression methods in an idealized setting. Afterwards, we have applied the same methods to a terrain data from Saarland, Germany.

2.6.1 The Franke Function

The function we use to generate data is the Franke function

$$f(x,y) = \frac{3}{4}\exp(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}) + \frac{3}{4}\exp(-\frac{(9x+1)^2}{49} - \frac{(9y+1)^2}{10}) + \frac{1}{2}\exp(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}) - \frac{1}{5}\exp(-(9x-4)^2 - (9y-7)^2). \tag{14}$$

We generate points (x_i, y_i) from a uniform distribution on $[0,1] \times [0,1]$. We then compute the dependent variable $z_i = f(x_i, y_i) + \epsilon_i$, where $\epsilon_i \overset{\text{iid}}{\sim} \text{N}(0, \sigma^2)$.



Visualisation of the Franke function for $x, y \in [0,1] \times [0,1]$

Visualisation of the noisy data sampled from Franke function (without scaling)

Figure 1: Visualisation of the Franke function, and the sampled data

To simulate data collection we choose a constraint number of points, $i = 1, ..., 1000$ (x_i, y_i), and calculate associated z_i from the Franke function with noise and set $\sigma^2 = 0.1$ eq(14). Further we will use the three regression models motioned above to create linear models with goal of finding the one which best predicts future z-values.

2.6.2 Saarland Terrain Data

Why we do what do with the Franke function will be clearer as we approach a similar case with terrain data. The analysis above will serve as a stepping stone to the terrain data analysis.

We obtained data of terrain from Saarland in Germany through the EarthExplorer feature by the United States Geological Survey[1]. This data was downloaded as a TIF-file, and could be represented as a 3601×3601 matrix, where each element position i, j relative to each other

corresponded to their real life relative position, and the elements denoting their height, ranging between a maximum of 683 and a minimum of 59. For our analyses, we picked a random subset of these points; due to time constraints, the amount of points we have chosen for the analyses is not constant, but varies based on what was computationally agreeable.

2.7 Scaling

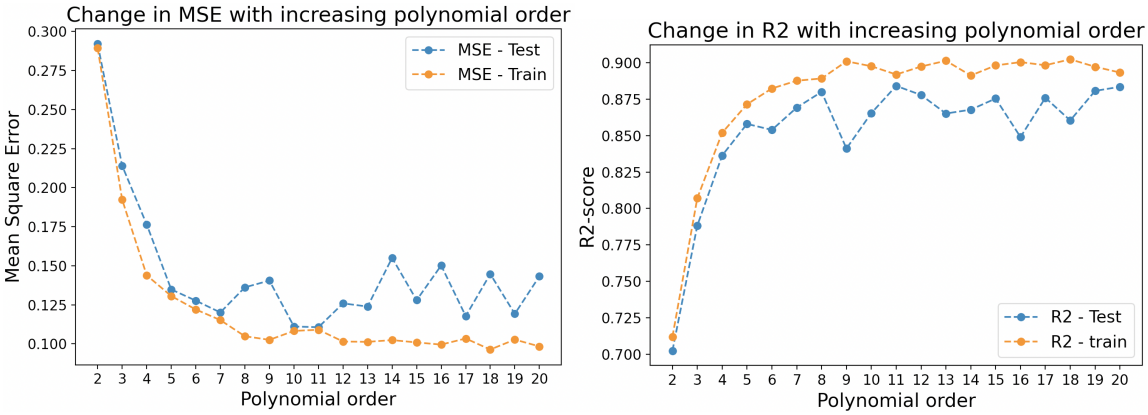
Since the goal of this report is to compare the performance of linear regression on two different data sets, we scale both data sets to the same scale to more easily compare our error estimates. It is also worth noting that we wish to penalize each feature equally, when applying regularization. To satisfy both of these requirements, we center our features, and we divide each feature by its standard deviation. We also center the dependent variable to ensure that the regularization does not affect the constant term.

3 Results

The source code is linked in appendix A.

3.1 Approximating the Franke Function

Figure 2 shows the MSE and R2 of the training data and the test data as a function of polynomial degree when performing OLS regression. We find that the train MSE tends downwards, and that the train R2 tends upwards with increasing polynomial degree. We note that the test MSE, however, appears to stabilize in this degree range.



MSE for polynomial fits of sampled data from Franke function for increasing orders

R2-score for polynomial fits of sampled data from Franke function for increasing orders

Figure 2: MSE and R2-score for polynomial fits on data from Franke function

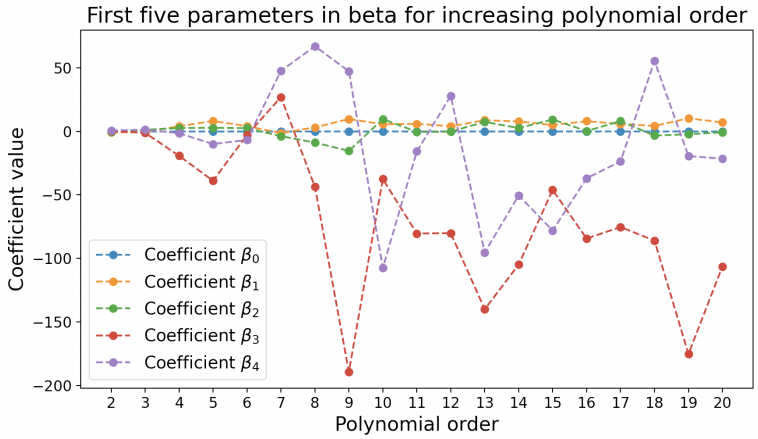


Figure 3: First five parameters of $\boldsymbol{\beta}$ as a function of polynomial order

Figure 3 shows the first five parameters of $\boldsymbol{\beta}$ as a function of polynomial order. We observe that the parameter values fluctuate erratically with increasing polynomial degrees. We note that the parameter values even change signs.

We use bootstrapping to estimate the bias and variance of models for polynomials of degrees 1 through 10. We find that the bias tends to decrease with polynomial degree, and that the variance tends to increase, as seen in figure 4. We note that the estimated error is minimal for a polynomial of degree around 6, using these parameters. However, the difference between the errors are relatively small, meaning that the particular degree with minimal error that we found might be due to randomness.

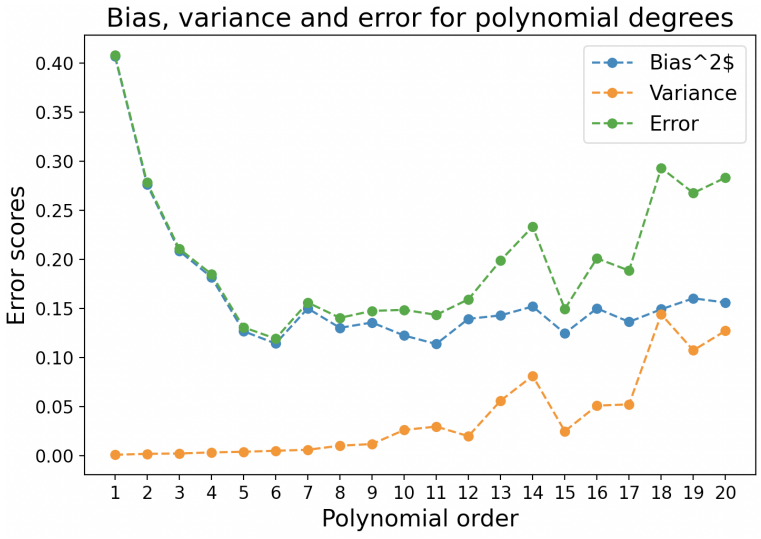
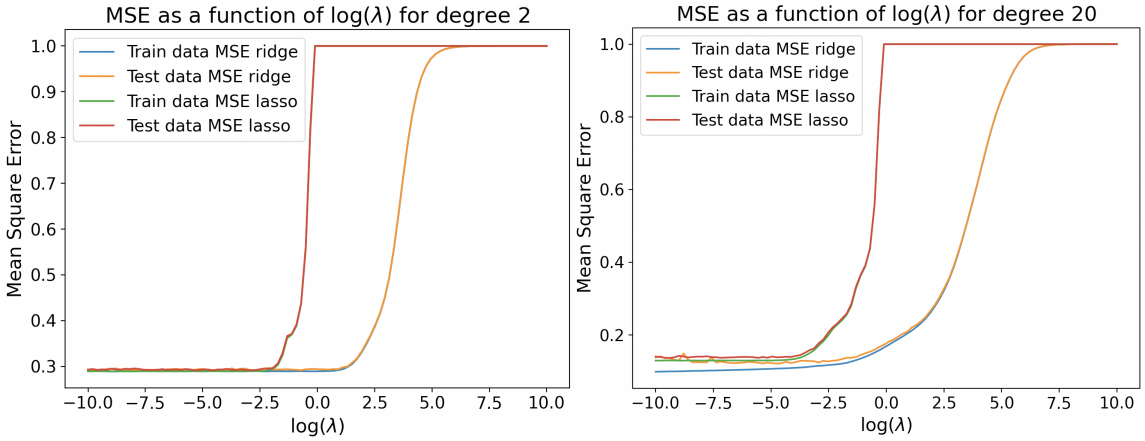


Figure 4: Bias, variance and error as a function of polynomial degree



MSE as a function of λ for polynomial degree 2, with data from Franke function

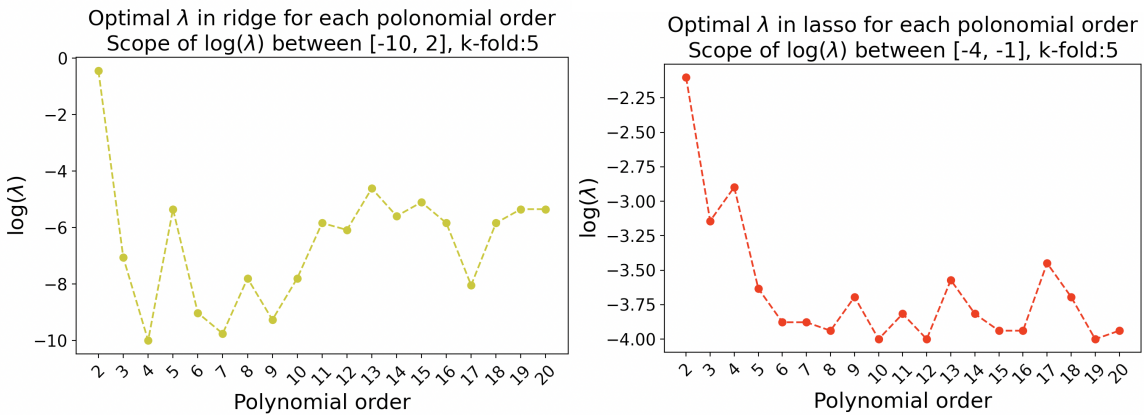
MSE as a function of λ for polynomial degree 20, with data from Franke function

Figure 5: MSE as a function of λ for different polynomial degrees, with data from Franke function

For polynomials of degree 2 and 20, we perform ridge and lasso regression for λ values between 10^{-10} and 10^{10}. We do so to see which values of λ are worth investigating. As we can see in figure 5, we observe that choosing λ values below 10^{-1} is reasonable for lasso regression. We also note

that λ values below 10^2 are worth investigating for ridge regression. As these ranges remain more or less constant between degrees 2 and 20, we assume that this is a reasonable range to explore for all degrees.

To find the optimal parameter λ when performing ridge and lasso regression, we perform a cross-validation grid search. Figure 6 shows the optimal $\log(\lambda)$ value as a function of polynomial degree for ridge regression and lasso regression. We see that we get a high value of λ for degree 2, whilst the rest seem to vary randomly at low values.



Optimal value of λ with ridge regression for each polynomial degree

Optimal value of λ with lasso regression for each polynomial degree

Figure 6: Optimal λ for each polynomial degree from 2 through 20, for ridge and lasso regression on the Franke function.

We extract the optimal λ for each degree, and plot the performance of all three models as a function of polynomial degree. As can be seen in figure 7, OLS appears to perform the best up until degree 15, when ridge regression seems to outperform. Lasso, meanwhile, performs consistently worse than the other two models.

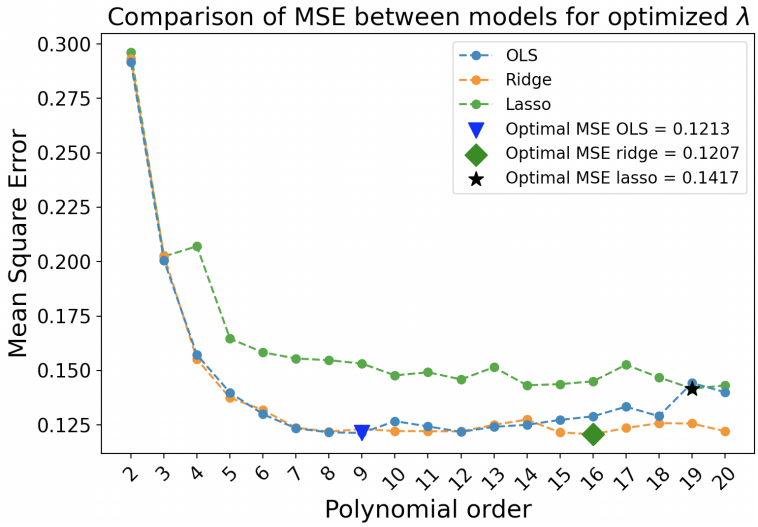


Figure 7: Comparison of MSE between OLS, ridge and lasso regression with optimal λ for each polynomial degree

Pickining the polynomial degrees that yield minimal error, we find that OLS has a minimum for degree 9. Ridge has a minimum for degree 16. Meanwhile, lasso regression has its minimum for a degree 19 polynomial. We find that ridge regression yields the lowest error, namely 0.1207. Meanwhile, OLS regression performs slightly worse with an error of 0.1213. And finally, we get that lasso regression performs even worse with an error of 0.1417. The optimal model for the Franke function is visualized in figure 9, with polynomial degree 16 and $\lambda = 1.456 \cdot 10^{-6}$.

3.2 Approximating the Saarland Terrain data

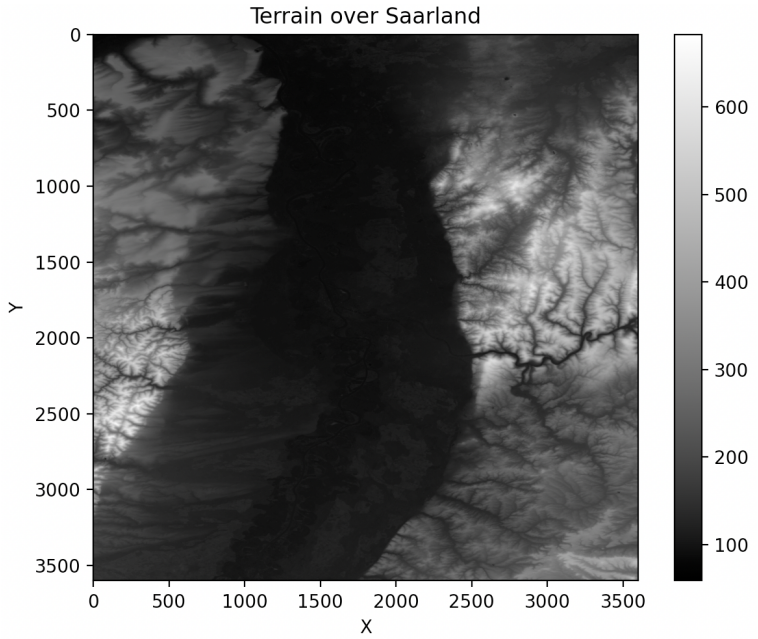


Figure 8: Height map for terrain data over Saarland

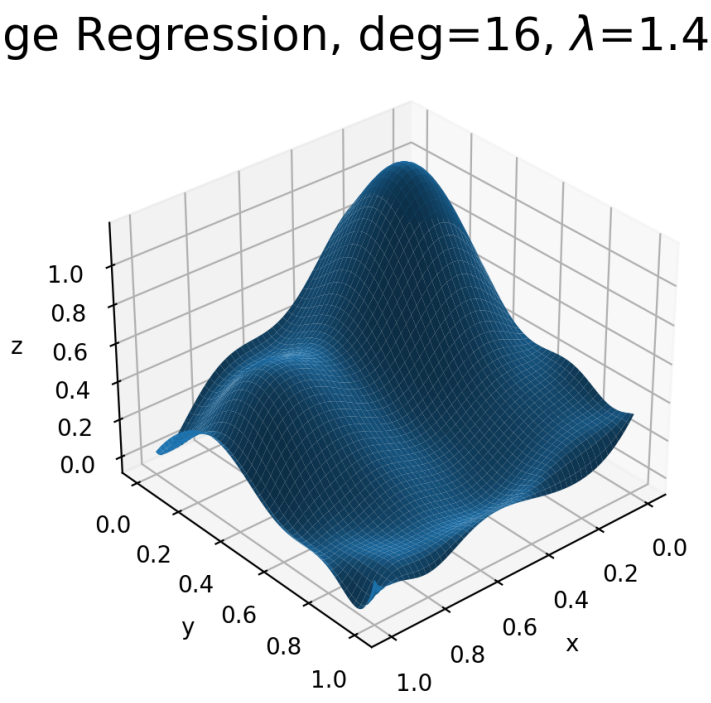
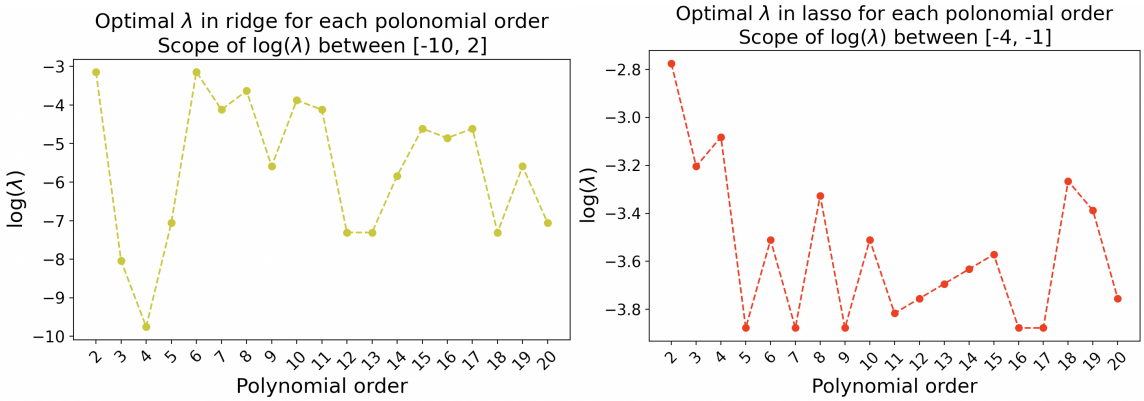


Figure 9: Visualization of the optimal model for Franke function

Using the same analysis as we did with the Franke function, we find the optimal λ for ridge and lasso regression, by performing a cross-validation grid search for each polynomial degree. Figure 10 shows the optimal $\log(\lambda)$ value as a function of polynomial degree for ridge regression and lasso regression. We see that λ varies significantly between polynomial orders.



Optimal value of λ with ridge regression for each polynomial degree

Optimal value of λ with lasso regression for each polynomial degree

Figure 10: Optimal λ for each polynomial degree from 2 through 20, for ridge and lasso regression on Saarland terrain data

By extracting the optimal λ for each degree, we plot the performance of all three models as a function of polynomial degree. As can be seen in figure 11, OLS and ridge variably obtain the lowest MSE for each polynomial order, while lasso regression consistently performs worse than the other two at most degrees. For all models, they perform progressively worse from the minimum as the polynomial order increases.

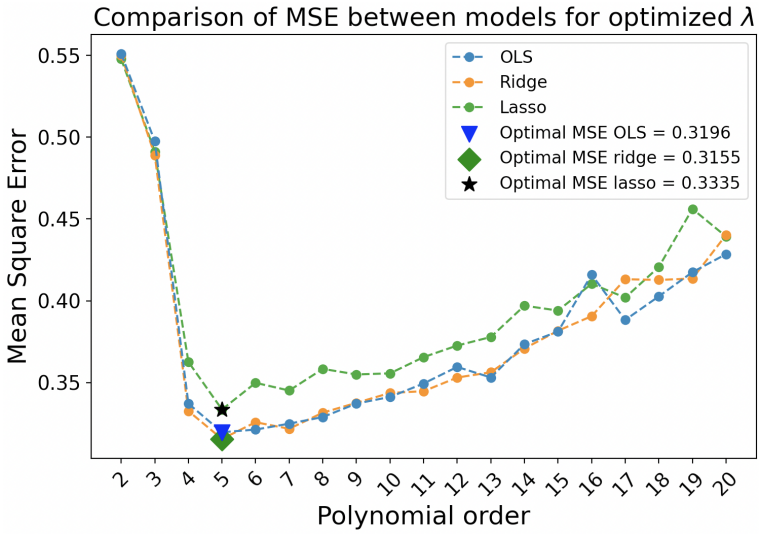


Figure 11: Comparison of MSE between OLS, ridge and lasso regression with optimal λ for each polynomial degree on Saarland terrain data

Picking the polynomial degrees that yield minimal error, we find that OLS, ridge and lasso all have a minimum for degree 5. Ridge yields a scaled error of 0.3155. Meanwhile, OLS regression performs slightly worse with a scaled error of 0.3196. And finally, we get that lasso regression performs even worse with a scaled error of 0.3335. We have visualized the optimal model - ridge regression with polynomial degree 5 and $\lambda = 8.685 \cdot 10^{-8}$ - in figure 12

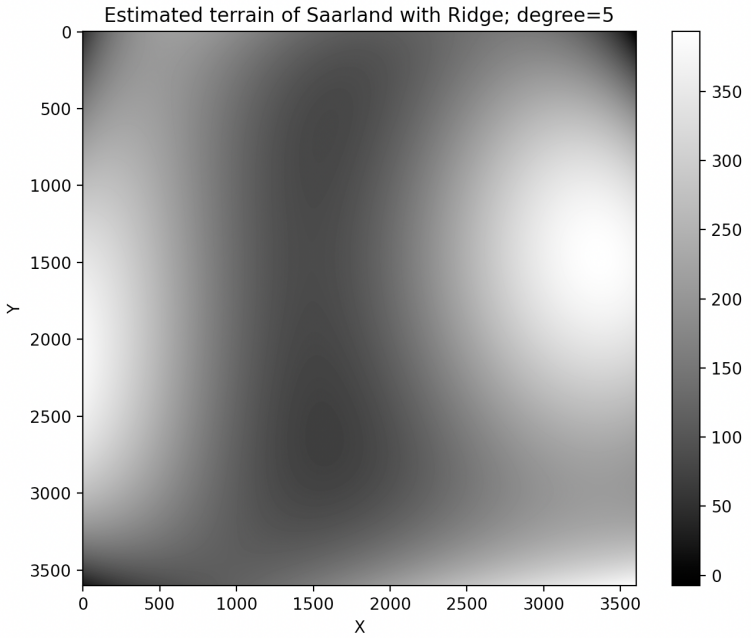


Figure 12: Visualization of the optimal model for Saarland terrain data for all data points

4 Discussion

The finding that the train MSE continually decreases as polynomial order increases is sensible, since higher polynomial degrees allow the model to better adapt to the training data set. In fact, if we have a polynomial of degree n, the polynomials of degree $k < n$ are a subset of the n-degree polynomials. This means that if a lower-degree polynomial performs at a certain level, a higher degree polynomial must perform equally or better. Then it also makes sense that the R2 improves on the training data.

Also, the fact that the performance on the test data levels out with increasing polynomial degree is sensible. As the model complexity increases, while the number of points stays constant, one would expect some degree of overfitting. In other words, more degrees of freedom allows the model to adapt to the errors in the training data as well as adapting to the underlying function. This is why the model does not appear to improve on the test data, while it does improve on the training data.

The concept of the Bias-Variance trade-off is clearly illustrated by figure 4, where we clearly see that the bias decreases as the polynomial order increases and that the variance decreases. Optimally, we are looking for a middle ground, where the bias is low enough that the model captures the general features of the data set, but where the variance is also low enough that the model generalizes to new data. While we have found that Since there appears to be a balance over a wide range of

polynomial degrees, we note that multiple degree choices can yield good models.

At the same time, one needs to look out for overfitting. For one, we need to ensure that the model performs well on test data. A train/test split, or perhaps cross-validation, are good ways of detecting this issue. On the other hand, in some situations we want our model to generalize outside of the available data domain. In such situations, one can restrict model complexity out of caution. When in doubt, less complex models generalize better. In fact, the fluctutations of the β_i's shown in figure 3 may indicate precisely that the model will yield poor predictions going outside of the $[0, 1] \times [0, 1]$ square.

The reason behind the high value of λ at polynomial degree 2 for the Franke function in figure 6 is not clear to us. We would have expected to see a low value of λ as for low polynomial degrees, as the model complexity is already low. We leave figuring this phenomenon out as an exercise to the reader.

The observation from the models for the Franke function that OLS performs better for lower polynomial degrees, and that ridge outperforms at high degrees appears reasonable to us. We should expect that a high polynomial degree would result in some extent of overfitting. Then the L^2 regularization can counteract this effect for higher polynomial degrees. For lower degree polynomials, however, this regularization constrains the models when overfitting is not an issue, causing worse performance.

We do, however, see a much clearer indication that a degree 5 polynomial performs better on the Saarland data. It is interesting to see that there is a more clear-cut optimal model for the terrain data than for the Franke function, providing a distinct minimum for all three regression models. We suspect that this may be due to the difference in variance in the noise; the noise on the data from the Franke function has low variance of $\sigma^2 = 0.1$, which likely makes the function easier to approximate even with higher degree polynomials. We expect that the Franke data would have a more clear-cut optimal polynomial degree for a higher noise-level. Meanwhile, the real world terrain data exhibits unsmooth behaviour with grooves and peaks - thus the optimal model stands out more clearly.

5 Conclusion

Using $n = 1000$ and $\sigma^2 = 0.1$, we perform a cross-validation grid search to find that ridge regression outperforms OLS and lasso regression when approximating the Franke function. In particular, ridge regression attains a minimal MSE value of 0.1207 using polynomial degree 16 and $\lambda = 1.456 \cdot 10^{-6}$. And, when analyzing the Saarland terrain data, we also find that ridge regression performs the best. There, we find that a polynomial of degree 5 and $\lambda = 8.685 \cdot 10^{-8}$ yields a scaled MSE of 0.3155, where higher degree polynomials clearly lead to overfitting. A central finding of ours is that ridge regression tends to outperform OLS for higher degree polynomials, since regularization can counteract overfitting.

6 References

1 United States Geological Survey. https://earthexplorer.usgs.gov/. Accessed: 2022-10-11.

A Code

The code used to compute estimates and generate plots can be found at:
https://github.com/carlfre/FYS-STK3155-project-1

B Deriving the Expected Value and Variance of $\hat{\boldsymbol{\beta}}$

In this appendix, we will prove that the OLS estimator $\hat{\boldsymbol{\beta}}$ is unbiased, and we will derive its variance.

In this appendix, we will derive the expected value and variance of y_i to prove equations 15 and 19. We first derive the expected value. By linearity of expectation, we have that

First, we derive the expected value and variance of y_i. We have that the expected value of the i'th component of $\boldsymbol{y}$ is given by

$$\mathbb{E}(y_i) = \sum_j x_{ij}\beta_j = \mathbf{X}_{i,*}\beta, \tag{15}$$

Proofs for these statements can be found in appendix A.

$$\mathbb{E}(y_i) = \mathbb{E}\left(\sum_j x_{ij}\beta_j + \epsilon_i\right) = \mathbb{E}\left(\sum_j x_{ij}\beta_j\right) + \mathbb{E}\left(\epsilon_i\right). \tag{16}$$

The expected value of ϵ_i is zero, thus

$$\mathbb{E}(y_i) = \mathbb{E}\left(\sum_j x_{ij}\beta_j\right). \tag{17}$$

And since $\sum_j x_{ij}\beta_j$ is not stochastic, we get that

$$\mathbb{E}(y_i) = \sum_j x_{ij}\beta_j = \mathbf{X}_{i,*}\beta, \tag{18}$$

which proves equation 15. Next, we will prove that the variance of y_i is given by

$$\mathrm{Var}(y_i) = \sigma^2. \tag{19}$$

we use that the variance is given by

$$\mathrm{Var}(y_i) = \mathbb{E}(y_i^2) - \mathbb{E}(y_i)^2 \tag{20}$$

and substitute the second term using equation 15 to get that

$$\mathrm{Var}(y_i) = \mathbb{E}((\mathbf{X}_{i,*}\beta + \epsilon_i)^2) - (\mathbf{X}_{i,*}\beta)^2. \tag{21}$$

Expanding the square and applying linearity of expectation, we get that

$$\mathrm{Var}(y_i) = (\mathbf{X}_{i,*}\beta)^2 + 2\mathbb{E}\left(\epsilon_i\right)\mathbf{X}_{i,*}\beta + \mathbb{E}\left(\epsilon_i^2\right) - \left(\mathbf{X}_{i,*}\beta\right)^2. \tag{22}$$

The expected value of ϵ_i is zero; thus

$$\mathrm{Var}(y_i) = \mathbb{E}(\epsilon_i^2) = \mathrm{Var}(\epsilon_i) + \mathbb{E}(\epsilon_i)^2 = \sigma^2. \tag{23}$$

This finishes the proof of equation 19. Next, we show that $\hat{\boldsymbol{\beta}}$ is unbiased From equation 15 and the normal equation 3, we get that

$$\mathbb{E}(\hat{\boldsymbol{\beta}}) = \mathbb{E}\left((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}\right) = \mathbb{E}\left((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}\right). \tag{24}$$

With the inverses cancelling out and using that $\mathbb{E}(\boldsymbol{\beta}) = \boldsymbol{\beta}$, we get that

$$\mathbb{E}(\hat{\boldsymbol{\beta}}) = \boldsymbol{\beta}. \tag{25}$$

And next, we show that

$$\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}. \tag{26}$$

From the definition of the variance:

$$Var(\hat{\boldsymbol{\beta}}) = \mathbb{E}\left(\left(\boldsymbol{\beta} - \mathbb{E}(\boldsymbol{\beta})\right)\left(\boldsymbol{\beta} - \mathbb{E}(\boldsymbol{\beta})\right)^T\right) \tag{27}$$

The expected value $\mathbb{E}(\boldsymbol{\beta})$ is $\boldsymbol{\beta}$ as previously shown; thus:

$$Var(\hat{\boldsymbol{\beta}}) = \mathbb{E}\left(\left((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} - \boldsymbol{\beta}\right)\left((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} - \boldsymbol{\beta}\right)^T\right) \tag{28}$$

$\mathbf{X}$ is not a stochastic variable, and so we have:

$$Var(\hat{\boldsymbol{\beta}}) = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbb{E}\left(\mathbf{Y}\mathbf{Y}^T\right)\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \tag{29}$$

We calculate $E\left(\mathbf{Y}\mathbf{Y}^T\right)$ before proceeding:

$$E\left(\mathbf{Y}\mathbf{Y}^T\right) = E\left((\mathbf{X}\boldsymbol{\beta} + \epsilon)(\mathbf{X}\boldsymbol{\beta} + \epsilon)^T\right) \tag{30}$$

$$E\left(\mathbf{Y}\mathbf{Y}^T\right) = E\left(\mathbf{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\mathbf{X}^T + 2\mathbf{X}\boldsymbol{\beta}\epsilon + \epsilon^2\right) \tag{31}$$

$$E\left(\mathbf{Y}\mathbf{Y}^T\right) = \mathbf{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\mathbf{X}^T + 2\mathbf{X}\boldsymbol{\beta}\mathbb{E}(\epsilon) + \mathbb{E}(\epsilon^2) = \mathbf{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\mathbf{X}^T + \sigma^2 \tag{32}$$

Inserting:

$$Var(\hat{\boldsymbol{\beta}}) = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\mathbf{X}^T + \sigma^2)\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \tag{33}$$

Most of the inverses cancel and we are left with

$$\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \boldsymbol{\beta}\boldsymbol{\beta}^T + \sigma^2(\mathbf{X}^T\mathbf{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}, \tag{34}$$

which finishes the proof.

C Deriving the Bias-Variance decomposition

In this appendix, we will derive the decomposition of the mean squared error, as described in equation 9. We have that

$$\begin{aligned}\mathbb{E}((\boldsymbol{y}-\tilde{\boldsymbol{y}})^2) &= \mathbb{E}((f(\boldsymbol{x})+\boldsymbol{\epsilon}-\tilde{f}(\boldsymbol{x}))^2) \\ &= \mathbb{E}((f((\boldsymbol{x}))-\tilde{f}(\boldsymbol{x}))^2) + 2\mathbb{E}((f((\boldsymbol{x}))-\tilde{f}(\boldsymbol{x}))\cdot\boldsymbol{\epsilon}) + \mathbb{E}(\boldsymbol{\epsilon}^2) \\ &= \mathbb{E}((f(\boldsymbol{x})-\tilde{f}(\boldsymbol{x}))^2) + \sigma^2,\end{aligned} \tag{35}$$

using the independence of $f(\boldsymbol{x})-\tilde{f}(\boldsymbol{x})$ and $\boldsymbol{\epsilon}$, that $\mathbb{E}(\boldsymbol{\epsilon})=0$, and that

$$\mathbb{E}(\boldsymbol{\epsilon}^2) = \mathrm{Var}(\boldsymbol{\epsilon}) + \mathbb{E}(\boldsymbol{\epsilon})^2 = \mathrm{Var}(\boldsymbol{\epsilon}) = \sigma^2. \tag{36}$$

Then we get that

$$\begin{aligned}\mathbb{E}((f(\boldsymbol{x})-\tilde{f}(\boldsymbol{x}))^2) &= \mathbb{E}(((f(\boldsymbol{x})-\mathbb{E}(\tilde{f}(\boldsymbol{x}))-(\tilde{f}(\boldsymbol{x})-\mathbb{E}(\tilde{f}(\boldsymbol{x})))^2) \\ &= \mathbb{E}((f(\boldsymbol{x})-\mathbb{E}(\tilde{f}(\boldsymbol{x})))^2) + \mathbb{E}((\tilde{f}(\boldsymbol{x})-\mathbb{E}(\tilde{f}(\boldsymbol{x})))^2) \\ &\quad - 2\cdot\mathbb{E}((f(\boldsymbol{x})-\mathbb{E}(\tilde{f}(\boldsymbol{x})))(\tilde{f}(\boldsymbol{x})-\mathbb{E}(\tilde{f}(\boldsymbol{x})))) \\ &= \mathrm{Bias}(\tilde{\boldsymbol{y}})^2 + \mathrm{Var}(\tilde{f}(\boldsymbol{x})) + 0 \\ &= \mathrm{Bias}(\tilde{\boldsymbol{y}})^2 + \mathrm{Var}(\tilde{f}(\boldsymbol{x})),\end{aligned} \tag{37}$$

which finishes the proof.