FYS-STK4155 H22: Project 1

Lasse Totland, Domantas Sakalys, Andreas Lie Massey, Synne Mo Sandnes
(Dated: October 1, 2022)

abstracctttttttttttttttttttttt

I. INTRODUCTION

This report demonstrates and compares various methods for regression and resampling analysis by applying them to three-dimensional data-sets; one set semi-randomly generated and the other a real set of terrain data. We will also use different methods, like evaluating mean square error, to quality check the resulting regression models. (...)

II. THEORY

Some of the formulae presented in this report have been borrowed from the task descriptions' original LaTeX file. [1]

Borrowed figures are marked as such, and figures made by us are unmarked.

A. Franke Function

The Franke function, which is a weighted sum of four exponentials reads as follows:

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

For a two-dimensional xy-plane input, the function gives us a 'terrain' plot.

B. Mean Square Error (MSE)

The Mean Square Error (MSE) of an approximated model $\tilde{y}$, given the inital data y can be expressed:

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

where n is the number of values in $\tilde{y}$.

C. Score Function, R^2

The score function can be used to determine the quality of our model. If $\tilde{y}_i$ is the predicted value of the $i'th$ sample and y_i is the corresponding true value, then the score R^2 is defined as

$$R^2(y,\tilde{y}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2},$$

where we have defined the mean value of y as

$$\bar{y} = \frac{1}{n}\sum_{i=0}^{n-1} y_i.$$

The score function can have values in the range $R^2 \in [0,1]$ where $R^2 = 1$ would mean that $\tilde{y}$ was a perfect fit of y.

D. Ordinary Least Squares (OLS) method

Given a dataset with n values in the form $y = f(x)+\epsilon$, where $\epsilon \sim N(0,\ \sigma^2)$ is normally distributed noise, $f(x)$ can be approximated via the development of a model of the form:

$$\tilde{y} = X\beta.$$

$X \in \mathbb{R}^{n\times p}$ is a design matrix with columns representing the different features p of the model. For a simple second degree polynomial, the resulting model may look like this:

$$\tilde{y} = \beta_0 + x\beta_1 + x^2\beta_2.$$

β is thus a column vector with an amount of scalar values β_i ($i \in [0,p]$) representing specific features. The OLS method defines it as:

$$\beta^{OLS} = (X^TX)^{-1}X^Ty.$$

E. Ridge method

Ridge regression uses a modified definition of β^{OLS}:

$$\beta^{Ridge} = (X^TX + I\lambda)^{-1}X^Ty.$$

Here $I^{p\times p}$ is the identity matrix, and $\lambda \geq 0$ is a regularization parameter (...)

F. Lasso method

Lasso regression ('Least Absolute Shrinkage and Selection Operator') uses a modified definition of β:

$$\beta_i^{Lasso} = \begin{cases} (y_i - \frac{\lambda}{2}), & \text{if } y_i > \frac{\lambda}{2} \\ (y_i + \frac{\lambda}{2}), & \text{if } y_i < -\frac{\lambda}{2} \\ 0, & \text{if } |y_i| \leq \frac{\lambda}{2}. \end{cases},$$

where λ is a regularization parameter as in the Ridge method.

G. Variance of β

H. The Cost function

Given a model $\tilde{y}$, approximating data $y = f(x) + \epsilon$, its corresponding parameters β are in turn found by optimizing the mean squared error via the so-called cost function, which we have rewritten as:

$$C(X, \beta) = \mathbb{E}\left[(y - \tilde{y})^2\right] = (\text{Bias}[\tilde{y}])^2 + \text{var}[\tilde{y}] + \sigma^2,$$

where $var[\tilde{y}]$ and σ^2 are the variances of the model and the noise respectively, representing the spread of these data. The square of the bias of the model represents error from wrong assumptions in the model; one can think of a model with high bias to have an inherent systematic error. Figure 1 presents a useful visualization of how variance and bias of a model relate. The derivation of the equation above is presented in Appendix A.

The terms represent different potential sources of error in our model, so the cost function can be used to detect this and minimize the mean square error.

I. Bias-variance trade-off

The details of the relation between the bias and variance of a model is interesting as well, and can tell us something about the optimal complexity of the model. Complexity in regression analysis refers to the polynomial degree of the model attempt. The 'trade-off' refers to a compromise; we mostly cannot reduce both the error from the bias and the variance to its minimum, but we can determine a middle-point that sufficiently avoids *overfitting* and *underfitting*.

In Figure 2 we can see how the optimal model complexity is determined to be at the minimum total error (about where the bias and variance cross). Decreasing the complexity leads to underfitting; the model does not have the sufficient parameters to accomplish a good fit. Increasing the complexity from the 'sweet spot' also increases error as the model gains more features but also most likely gains more spread in the model data.

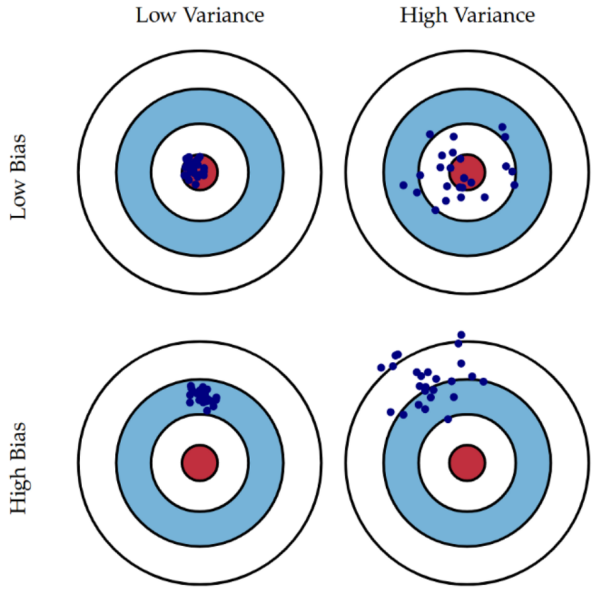

Figure 1. Visualization of the dynamic between bias and variance in a model. The intent of this model would be to hit the red bullseye in all instances. [a]

[a] 'Understanding the Bias-Variance trade-off', Scott Fortmann-Roe (2012) http://scott.fortmann-roe.com/docs/BiasVariance.html

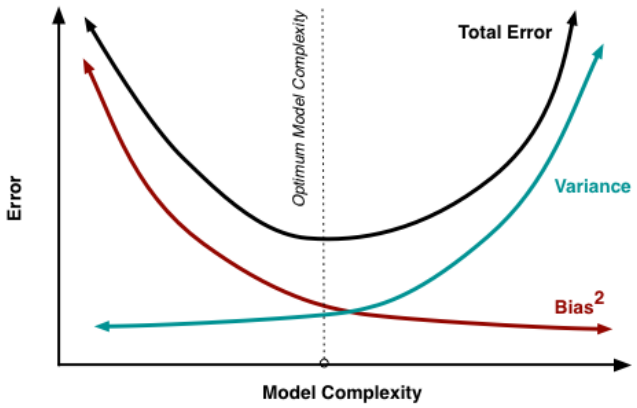

Figure 2. Visualization of the dynamic between bias, variance and total error in a model as a function of model complexity. [a]

[a] 'Understanding the Bias-Variance trade-off', Scott Fortmann-Roe (2012)

A bias variance plot most likely won't be as smooth as this example figure and the optimal complexity won't be as obvious, so we may have to compare with other methods to make a good evaluation.

J. Resampling methods

While analyzing a limited dataset it may be useful to 'resample' it; create subsets of the original data with new

variations (for example by replacing values) and then analyze those as well and evaluate how this extends the model. The intent of resampling is essentially to have more data to work with during analysis. We're going to use two such methods for this project.

The *bootstrap method* involves generating additional datasets y_i from an original y, where a larger range of i's would mean a more thorough evaluation. The additional sets resample randomly chosen values from the original, and analyzing all of these sets will give a distribution of predicted models where the mean is determined to be the optimal one. In regression analysis for example, we'll get a distribution of β_k's that represent approximation models.

When developing approximation models it may be useful to divide the data y into a *training* set to build the model, and keep a spare *test* set to evaluate the quality of the model; a sufficient model should be able apply itself to new data. The *Cross-validation method* mixes up how this is structured, and generates a sequence of k equally sized groups from a shuffled subset y_i of y, and determines one of these to be the test set. Analyzing all of these training/test-set groups will give a distribution of models, where the mean is determined be the optimal one.

III. METHOD

We generate the data that will be analyzed by inserting uniformly generated points $x, y \in [0, 1]$, inserting into Franke function $f(x, y)$ and then adding stochastic, normal-distributed noise $\epsilon \sim N(0,\ \sigma^2)$. We define the dataset as:

$$z = f(x, y) + \epsilon,$$

assuming a variance of $\sigma^2 = 0.1$. See Figure 3 for an example of how the generated data may look.

A. Applying the Ordinary Least Squares (OLS) analysis

We wish to approximate the initial function $f(x, y)$ with our model $\tilde{z}$ following from the OLS-method:

$$\tilde{z} = X\beta,$$

where X is the design matrix determined by the polynomial degree of our model. The form of our design matrix will correspond to a multiple polynomial regression model, meaning each feature of our β will be a product of various degrees of x^i, y^i for $i \in [1, N]$ where N is the polynomial degree of our regression. For $N = 1$ for example, our model will have three features and may look like this (including intercept):

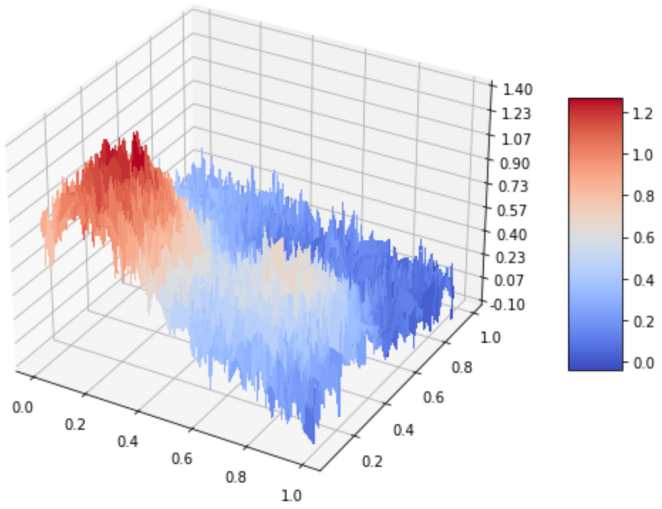

Figure 3. An example of generated data **z** from Franke function as described in the method section. This specific terrain was generated with a resolution of a 100×100 xy-grid.

$$\tilde{z} = \beta_0 + x\beta_1 + y\beta_2 + xy\beta_3.$$

The python function for constructing the design matrices ('create_X') is borrowed from the lecture notes [2], and can be found in the attached jupyter notebook with our code.

With a defined design matrix, we can can find β^{OLS} as defined in the theory section, giving us a complete model $\tilde{z} = X\beta$.

We'll use several strategies to evaluate the quality of our model. Firstly we'll split the data into 'training' and 'test' sets. More specifically, we are going to assign 80% of the data z and the same percentage of the rows in X to the training set, and use it to develop our model (calculate β), then use the remaining 20% to see if the finished model is in correspondence with the remaining data.

To quantitatively evaluate this we'll calculate the Mean Square Error (MSE) and the Score Function (R^2) for the training and test data sets and compare. We'll use the definitions presented in the theory section.

We'll also take into account the polynomial degree N of our OLS regression. A higher degree N will give the model more features, and more parameters to adjust the model to the data, but we'll have to evaluate if the added model complexity gives meaningful improvement or if it leads to overfitting (the model tries to fit to the noise and not the data 'underneath').

To do this we'll compare MSE and R^2 as functions of model complexity N. In addition, we will compare $var[\beta]$ as function of N, which represents the spread of elements in the β column-vector, and may also show signs of overfitting.

B. Bias-Variance trade-off analysis

To help us further determine to optimal model complexity (polynomial degree N) we'll also perform a bias-variance trade-off analysis, as described in the theory section. As a function of N, we'll plot the error, bias and variance and attempt to determine from this plot what degree N minimizes over- and underfitting.

The script used for this analysis will be an extended version of a script presented in the lecture notes which demonstrates bias-variance analysis of a two dimensional function (and implements a bootstrap method), which we will expand to function for three dimensional OLS models.

C. Cross-validation analysis

IV. RESULTS

A. Results from Ordinary Least Squares analysis

To understand the comparisons of model quality as function of complexity, it may be useful to see what the resulting OLS regression models look like for different polynomial degrees. In Figure 4 we demonstrate how increasing the degree from 2 to 5 gives a gradually more complex terrain, and that the higher degrees seem like more accurate recreations of the initial noisy terrain.

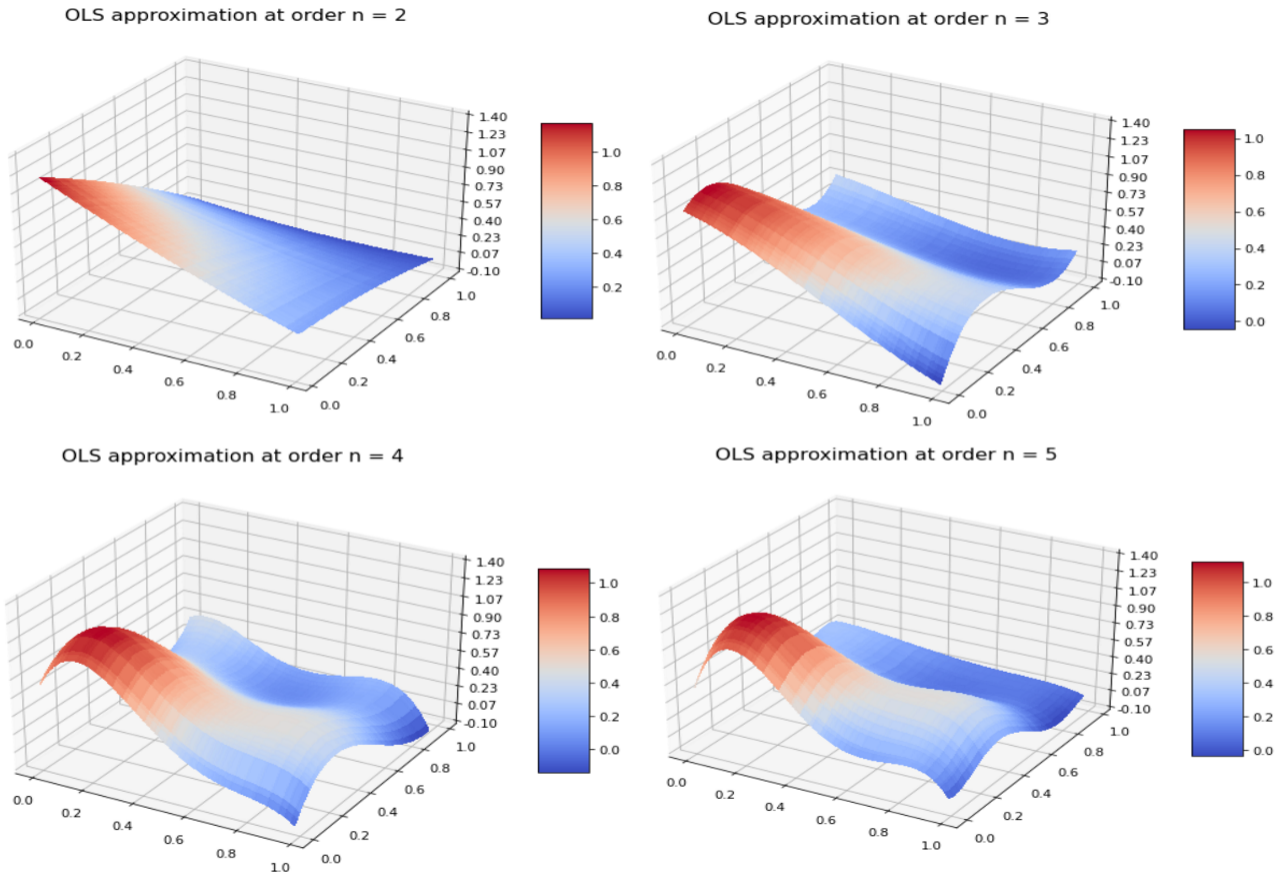

Figure 4. OLS regression models using different polynomial degrees to approximate a generated Franke function terrain data set as the one seen in Figure 3.

Figure 5 and 6 present the MSE and R^2 from a run of the script, respectively, as functions of model complexity. Figure 7 shows the variance of β as function of model complexity from the same run, as a way to compare the parameters.

B. Bias-Variance trade-off analysis

Figure 8 shows a result from bias-variance analysis of OLS approximations of Franke function as functions of model complexity N. For this run, the xy-grid had a resolution of 40×40 points, and the variance of the noise determined as $\sigma^2 = 0.1$. The script producing this plot also implemented 100 bootstrap iterations to resample the initial data for every polynomial degree.

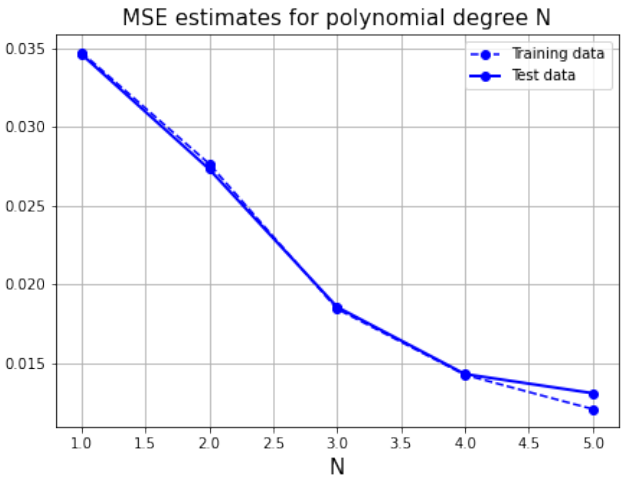

Figure 5. Mean Square Error of the OLS regression on the Franke function training and test sets as function of polynomial degree N.

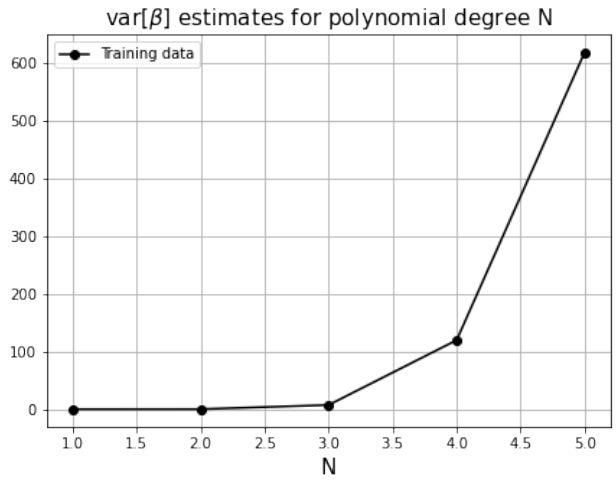

Figure 7. The variance $var[\beta]$ of the OLS regression on the Franke function training and test sets as function of polynomial degree N.

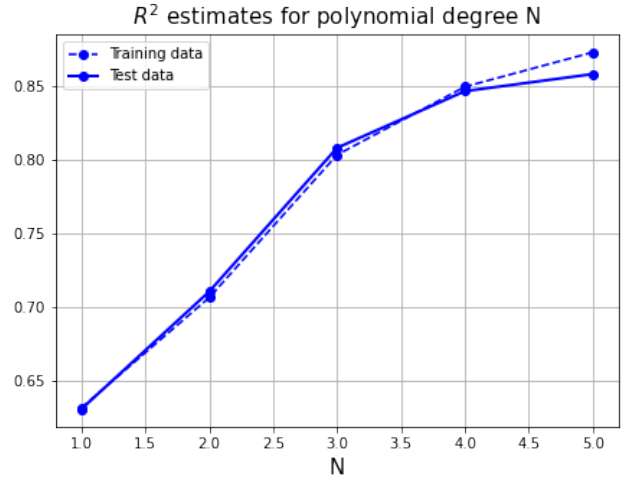

Figure 6. Score Function R^2 of the OLS regression on the Franke function training and test sets as function of polynomial degree N.

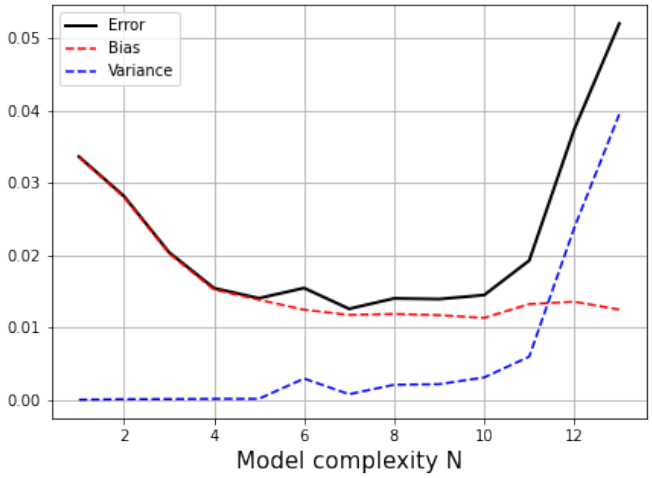

Figure 8. Error, bias and variance of OLS approximations of Franke function as functions of model complexity N.

V. DISCUSSION

When discussing the results of regression analysis of the Franke function in this report, its important to keep in mind that the datasets used were randomly generated (as described previously) giving slightly varying data, models and results during quality check. Each run of our script generates a new dataset and the figures presented in this report are the results of one such run, not a decisive and final representation of its qualities.

Despite the random effects on our analysis, there are patterns and distinctive tendencies with the results of each of the different methods however, which is what we will discuss.

A. Comparison of models

VI. CONCLUSION

VII. APPENDIX A: DERIVATIONS

A. Variance of β

B. The Cost function

The given cost function was defined as

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

Here the expected value $\mathbb{E}$ is the sample value. Given $y = f + \epsilon$ we can equate:

$$\mathbb{E}\left[(y-\tilde{y})^2\right] = \mathbb{E}\left[(f+\epsilon-\tilde{y})^2\right]$$

$$\mathbb{E}\left[(y-\tilde{y})^2\right] = \mathbb{E}\left[(f+\epsilon-\tilde{y}+\mathbb{E}[\tilde{y}]-\mathbb{E}[\tilde{y}])^2\right].$$

This multinomial expansion becomes long and tedious, but can be simplified if we keep in mind that all terms with a factor ϵ or $\mathbb{E}[\epsilon]$ become zero (the expectation value of normal distributed noise with mean value zero gives zero). The remaining terms are:

$$= \mathbb{E}[(y-\mathbb{E}[\tilde{y}])^2] + \mathbb{E}[\epsilon^2] + \mathbb{E}[(\mathbb{E}[\tilde{y}]-\tilde{y})^2]$$

$$= (\text{Bias}[\tilde{y}])^2 + \text{var}[\tilde{f}] + \sigma^2. \qquad \blacksquare$$

Here we also used that $(Bias[\tilde{y}])^2 = (y - \mathbb{E}[\tilde{y}])^2$ and that $var[\tilde{f}] = \frac{1}{n}\sum_i(\tilde{y}_i - \mathbb{E}\left[\tilde{y}\right])^2$.

[1] FYS-STK4155 Project 1 H22, Morten Hjorth-Jensen.
[2] Applied Data Analysis and Machine Learning lecture notes H22 subsection 4.8, Morten Hjorth-Jensen.