Linear Regression and Resampling Methods

Project 1 FYS-STK4155

Grzegorz Dariusz Kajda

October 13, 2022

Abstract

The following paper presents research conducted for the methods of Linear Regression, resampling techniques used for assesment of the models, and their application to real terrain data. The study of the regression methods starts with the application of Ordinary Lest Squares regression to a dataset consisting of polynomials of x and y, and fitting of said polynomials to Franke function. We measure models performance through the use of MSE and R2-score, and then evaluate it further using resampling techniques known as bootstrapping and kfold cross-validation. After finishing the evaulation of OLS, we repeat this process for Ridge and Lasso regressions, and compare the results of each model against each other. The evaluation process shows that Ridge regression clearly fits the data best, with OLS falling only slightly behind, while Lasso regression turns out to be computationally expensise. By the end of this project we also test the models using topograhic data. Unsuprisingly, Ridge regresison delivers the best performance out of all the models, with OLS coming in a close second position. The Lasso regression pn the other hand struggles to achieve comparable results, and takes considerably longer time predicting than the two other regression models.

Contents

1 Introduction

In a world experiencing an abundance of data never seen before, the wish of predicting the unkown has never been stronger. While the ability to predict unseen data is a nontrivial task, it most certainly is possible, and with the use of adequate tools, one may take advantage of the underlaying patterns that much of the data surrounding us displays. Today, one of the most commonly used methods for uncovering relationships between variables is machine learning, a filed within artificial intelligence which has made astounishing progress in the field of learning from data since 2011. And although there still exist problems that machine algorithms may struggle with, we shall prove that for many tasks, simple methods such as regression analysis will suffice.

In this research paper we will hence study three various methods of Linear regression known as the Ordinary Least Squares regression, Ridge regression and Lasso regression, and how this methods can be used to model relationships between variables. Our resarch will begin with the generetion of a small, noisy dataset, which will be fed to our models to fit a weighted sum of polynomials upto n-th order to the Franke Function. We will then measure the performance of our algorithms by computing the mean squared error (MSE) and R2-score, before we proceed with the application of resampling techniques called bootsrap and kFold cross valtidation to our models. The former enables the decomposition of the models error into bias, variance and noice, while the latter can be used for the estimation of the test error associated with a given method of statistical learning. Lastly, when our models have been evaluated using synthetic data, we will repeat this procedure to model topograhic data.

Following the introduction, the report introduces the fundamental mathematical theory and methods used, results obtained from running the algorithms, and a discussion about the perforamnce of the models presented in this research.

2 Theory

2.1 Linear models

As mentioned in the introduction, the aim of this project is to study methods of Linear regression, which are one of the best tools for building predictive models when we have measured data. Now, Linear regression can be described as a statistical approach to the explanation of a dependant variable $\mathbf{z}$ in terms of at least one independent, predictor variable $\mathbf{x}$. This allows us to model a measured respone $\mathbf{z}$ as a function of a set of k variables $\mathbf{x} = (x_0,x_1,x_2,...,x_{k-1})^T$:

$$\mathbf{z}=f(\mathbf{x})+\epsilon$$

Here ϵ is the error of our approximation that we wish to minimize for all data points. Now if no prior knowlegde in the form of a functional relationship is available to us, we assume the existance of a linear relationship between variables $\mathbf{z}$ and $\mathbf{x}$, which gives rise to the analytical equations of linear regression allowing us to write the expression above as

$$\mathbf{z}=\tilde{\mathbf{z}}+\epsilon$$

where $\tilde{z}$ describes the product of the k features $\mathbf{x}$ and k regression parameters $\beta=(\beta_0,\beta_1,\beta_2,...,\beta_{k-1})$, $\tilde{z}=\mathbf{x}\beta$, and is known as our prediction. The β parameters are the unknown variables that we wish find through solving the equation of linear regression. Now expanding will often find ourselves in situations where we want to approximate a set of n such response variables, $\mathbf{z}=(z_0,z_1,z_2,...,z_{n-1})$. One of the most common solutions to this problem is to parametrize our linear equation in terms of a polynomial function of n-1 degree:

$$\mathbf{z}=f(\mathbf{x})+\epsilon=\tilde{\mathbf{z}}+\epsilon=\sum_{j=0}^{n-1}\beta_j x_i^j+\epsilon_i$$

which as you shall see, is the approach used throughout this project. With a little linear algebra, we can stack all the feature vectors $\mathbf{x}$ on top of each other to form a *design matrix*

$$\boldsymbol{X}=\begin{bmatrix}1 & x_0^1 & x_0^2 & ... & ... & x_0^{k-1}\\ 1 & x_1^1 & x_1^2 & ... & ... & x_1^{k-1}\\ 1 & x_2^1 & x_2^2 & ... & ... & x_2^{k-1}\\ ... & ... & ... & ... & ... & ...\\ 1 & x_{n-1}^1 & x_{n-1}^2 & ... & ... & x_{n-1}^{k-1}\end{bmatrix}$$

also known as the *Vandermonde matrix*. Using the design matrix, and the set of k regression parameters β, our set of linear regression equations can be rewritten as

$$\tilde{y}=\mathbf{X}\beta+\epsilon$$

With a model of general linear regression defined, and our goal of minimizing the error of the models approximation, we can now move on to the Least Squares regression.

2.1.1 Ordinary Least Squares

In Ordinary Least Squared regression, we approach the task of finding the optimal parameters β for our model defined in the section above, by defining a cost function describing the average squared difference between our predicted values $\tilde{z}$ and the actual values $\mathbf{z}$, namely:

$$C(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = \frac{1}{n}\Big\{(\boldsymbol{y}-\tilde{\boldsymbol{y}})^T(\boldsymbol{y}-\tilde{\boldsymbol{y}})\Big\},$$

which we rewrite to a more compact form with the use of the design matrix $\mathbf{X}$

$$C(\boldsymbol{\beta}) = \frac{1}{n}\Big\{(\boldsymbol{y}-\boldsymbol{X\beta})^T(\boldsymbol{y}-\boldsymbol{X\beta})\Big\}.$$

Now, in order to compute the optimal parameters β, we are going to minimize the cost function by differentiating it with respect to the parameters β, and setting the resulting equation equal to zero. In other words, we will minimize the distance between the predicted data points, and the target values by solving the following problem

$$\frac{\partial C(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0$$

Which results in

$$\frac{\partial C(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0 = \boldsymbol{X}^T(\boldsymbol{y}-\boldsymbol{X\beta}).$$

By applying the rules of matrix mutliplication, we can rewrite the resulting expression as follows

$$\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{X}^T\boldsymbol{X\beta},$$

In the simple case where the matrix X^TX is invertible, we can simply solve this equation by multiplying both sides from the left with the inverse of this matrix, $(X^TX)^{-1}$, giving us

$$\boldsymbol{\beta_{optimal}} = \Big(\boldsymbol{X}^T\boldsymbol{X}\Big)^{-1}\boldsymbol{X}^T\boldsymbol{y}.$$

This means that we can now compute the prediction of our model as simply as

$$\tilde{\boldsymbol{z}} = \boldsymbol{X\beta_{optimal}}$$

The size of our design matrix defined as $\boldsymbol{X} \in \mathbb{R}^{k \times n}$ may possibly be quite large in situations where the number of predictors per response variable is a lot smaller than the number of response variables ($k \ll n$). Although it may seem like a heavy computation to perform, the fact that $\boldsymbol{X^T X}$ is invertible assures a low-dimensional product matrix of dimension k x k, hence allowing for a efficient calculation process.

2.1.2 Ridge Regression

In the solution for the optimal parameters for the least squares we propesed the cost function called mean sqaured error, which we then used to transform the regression problem to a optimization problem by minimizing the value of the cost function

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \frac{1}{n} ||\boldsymbol{y} - \boldsymbol{X\beta}||_2^2,$$

where we used the definition of the *Euclidean L^2-norm*

$$||\boldsymbol{x}||_2 = \sqrt{\sum_i x_i^2}.$$

While this approach proves to be quite efficient when applied to many problems, it may be prone to underfitting and overfitting as a result of the unconstrained nature of the OLS. In order to avoid this problem, we can add a regularization factor λ to the mean sqaured error function, shrinking the regression coefficeients β and thus defining a new cost function

$$C(\boldsymbol{X}, \boldsymbol{\beta}) = \frac{1}{n} ||\boldsymbol{y} - \boldsymbol{X\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_2^2$$

Rewritting the function terms of matrix-vector notation and removing $1/n$ yields a more familiar expression

$$C(\boldsymbol{X}, \boldsymbol{\beta}) = \big\{ (\boldsymbol{y} - \boldsymbol{X\beta})^T (\boldsymbol{y} - \boldsymbol{X\beta}) \big\} + \lambda \boldsymbol{\beta}^T \boldsymbol{\beta},$$

This newly defined cost function gives rise to a new optimization problem called Ridge regression, where $\lambda > 0$ represents a tunable penalty quantifying the amount of shrinkage we want to impose on the regression parameters (setting λ equal to zero gives us the standard OLS) . Solving the problem by again differentiating the cost funtion with respect to parameters $\boldsymbol{\beta}$ produces a slightly altered expression, which for finite values of the parameter λ ensures that our matrix will be non-singular. Now analytically solving the inversion problem, we aquire the optimal parameters through the equation given below

$$\hat{\boldsymbol{\beta}}_{\text{Ridge}} = \Big(\boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I} \Big)^{-1} \boldsymbol{X}^T \boldsymbol{y},$$

2.1.3 Lasso Regression

Another method of dealing with overfitting tendency of the least sqaures regression, is the application of the *Least Absolute Shrinkage and Selection Operator*, simply known as *Lasso Regression*. This method too introduces a tunable penalty factor λ, which when added to the mean sqaured error, produces a cost function similiar to that of Ridge regression

$$C(\boldsymbol{X},\boldsymbol{\beta})=\{(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})\}+\lambda||\boldsymbol{\beta}||_1,$$

where the $||\boldsymbol{\beta}||_1$ is a norm-1 parameter defined as

$$||\boldsymbol{x}||_1=\sum_i|x_i|.$$

The key difference between Ridge and Lasso stems from the different L-norms used with the λ.parameter, which for Lasso regression lead to the cost function not being differentiable everywhere. There are two consequence of that, one being the *absolute shrinkage* of many components contained within the vector of $\boldsymbol{\beta}$-parameters, which simply put means that many of the regression ceofficients are set to zero (Ridge regression shrinks, but does not set parameters equal to zero). Hence, Lasso is said to encourage simple sparse models with fewer parameters. Now the other consequence can be visualized through the derivation of the cost function with respect to β

$$\frac{\partial C(\boldsymbol{X},\boldsymbol{\beta})}{\partial\boldsymbol{\beta}}=-2\boldsymbol{X}^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})+\lambda sgn(\boldsymbol{\beta})=0,$$

where $sgn(\boldsymbol{\beta})$ is the derivative of the L_1-norm, with value equal to -1 for $\beta<0$ and 1 for $\beta>0$. Reordering of the solution for the derivative of the cost function yields the following expression

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}+\lambda sgn(\boldsymbol{\beta})=2\boldsymbol{X}^T\boldsymbol{y}.$$

which does not have any closed form solution. There are however other methods such as convex optimization, which employs the subgradient method, an iterative algorithm used for minimizing convex, nondifferentiable functions such as Lasso. However, due to the difficulty associated with the implementation of these methods, we are going to use a version of Lasso regression built into the Scikit-Learn library.

2.2 Singular Value Decomposition

Before we continue, we give a brief introduction to the well-known *Singular Value Decomposition* algorithm also known as the SVD. When working with both OLS and Ridge regression, we may sometimes run into the problem where the matrix X^TX is non-invertible. In such situations, we can use the SVD to decompose the matrix $\boldsymbol{X}$ with dimensions m x n in terms of a diagonal matrix $\boldsymbol{\Sigma}$ of dimensionality m x x (holding the singular-values of $\boldsymbol{X}$) and two orthogonal matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ of with dimensions m x x and n x n respectively:

$$\boldsymbol{X}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

This allows us to find the solution for both OLS and Ridge regression when our design matrix is near-singular or singular (a problem often arising when $\boldsymbol{X}$ is high dimensional). We can even use the *Economy-SVD* to speed up our calculations by contructing a pseudoinverse matrix A. This is done by removing all the singular-values equal to zero along the leading diagonal of matrix $\boldsymbol{\Sigma}$, and columns and rows from matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ which these singular-values correspond to. Going back to the case when X^TX is singular, we can rewriting it like this using the SVD

$$\boldsymbol{X}^T\boldsymbol{X}=\boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T,$$

and using the fact that $\boldsymbol{U}$ is orthonormal, shorten this expression to

$$\boldsymbol{X}^T\boldsymbol{X}=\boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}\boldsymbol{V}^T.$$

We now proceed by inserting this decomposition of matrix X^TX into the expression giving optimal regression parameters for OLS, and using SVD on the remaining matrices:

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

$$\hat{\boldsymbol{\beta}}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\left(\boldsymbol{V}\tilde{\boldsymbol{\Sigma}}^2(\boldsymbol{V}^T\right)^{-1}\boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{U}^T\boldsymbol{y},$$

Given the orthogonality of $\boldsymbol{U}$ and $\boldsymbol{V}$, we can shorten down this expression to yield the following equation:

$$\hat{\boldsymbol{\beta}}=\boldsymbol{U}\boldsymbol{U}^T\boldsymbol{y}=\sum_{i=0}^{p-1}\boldsymbol{u}_i\boldsymbol{u}_i^T\boldsymbol{y},$$

Using the decomposition of X^TX, we can rewrite the solution for Ridge regression in a similiar manner:

$$\hat{\boldsymbol{\beta}}_{\text{Ridge}}=\boldsymbol{X}\boldsymbol{\beta}_{\text{Ridge}}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}}\left(\boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^T+\lambda\boldsymbol{I}\right)^{-1}(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}})^T\boldsymbol{y}=\sum_{j=0}^{p-1}\boldsymbol{u}_j\boldsymbol{u}_j^T\frac{\sigma_j^2}{\sigma_j^2+\lambda}\boldsymbol{y},$$

2.3 Metrics for measurement of performance

As we happaned to mention in the introduction, our main metrics of models performance will be the mean sqaured error and R2-score. We are going to use the MSE to compute the error of each of the models

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

and use the R2-score function to provide us with the measure of how well our predictions approximate to real data. The R2-score can is given by the following function

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

and returns values ranging from 0 to 1, with $R^2 = 1$ indicating that the model always predicts the response variable, and $R^2 = 0$ representing a contant model always predicting the expected value of $\mathbf{y}$

2.4 Resampling techniques

To further evaluate our models, and gain deeper insight into their behaviour, we will subject our models to resampling techniques. The idea behind the techniques is to repeatedly draw samples from a training set and refit the model for each sample to uncover additional information about the fitted model. While resampling techniques are brilliant methods for gaining insight about our models behaviour, they suffer one major flaw, namely that they can be computationally expensive. This is a consequence of fitting the same model multiple times using different subsets of the training data, especially when large numbers of samples are involved. However,due to recent advancements in technology, these methods should in general prove to be not prohibitive.

2.4.1 The Bootstrap

Bootstraping is a non-parametric apporach to statistical inference involving the resampling of our tranining data through the drawing of n samples from it, calculating the β-parameters and evaluating the model for the data consisting of said n samples. In this project we draw k=n (n being the size of the training set) samples with replacement, meaning that an instance can occur in the sampled dataset more than once. This process is then repeated m times.

The bootstrap is particularly useful, as it allows us to decompose the error function in terms of the variance of the model itself, the mean value of the model (bias term) and the variance of the noise. This allows us to analyze the model in the light of the bias-variance tradeoff, which describes the tension between the complexity of the model and the amount of data need to train it. What do we mean by the variance and bias of the model? The variance desribes the amount the model changes by when using different samples of the training data. Bias on the other hand describes an event occuring when the the model systematically predicts the wrong variable (the model skews the result of the prediction). A golden rule for choosing a model of appropriate complexity, is to fina a balancepoint where these values equal one another, as this indicates the that the lowest possible value of error has been achieved.

As part of the discussion, we are also going to prove that the error of the model can med written as a sum of $\boldsymbol{bias}^2$, variance, and the irreducible error of the model, also defined as the variance of the noise. Now consider a dataset $\boldsymbol{L}$ consisting of data $X_L = (y_j,\, x_j,\, j = 0...n-1)$, we assume that the true data is generated from a noisy model

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

Here ϵ is the normally distributed error with mean zero and standard deviation σ^2. As in the theory section on Linear models, we define an approximation of the function f in terms of the parameters $\boldsymbol{\beta}$ and the design matrix $\boldsymbol{X}$, that is $\tilde{\boldsymbol{y}}=\boldsymbol{X}\boldsymbol{\beta}$. Given that the parameters β are optimized through the use of the mean sqaured error

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

Here the expected value $\mathbb{E}$ is the sample value. We start off by rewriting the expression for the cost function to better fit our proof

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

Expanding out the sqaures of this expression gives us the following equation:

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

As we know, the mean value of the error is zero, which means that by definition the expectation value of the error is also equal to zero. We can therefore set the last term on the RHS of hte equation above equal to zero, while the middle term can be written as $\boldsymbol{\sigma^2}$ due to it being the expaction value of the error of noise.

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

If we now subtract and add $\mathbb{E}[\tilde{\boldsymbol{y}}]$ to the equation above, we will get

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

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

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

$$=\boldsymbol{bias^2}+\boldsymbol{variance}+2(\boldsymbol{f}-\mathbb{E}[\tilde{\boldsymbol{y}}])\mathbb{E}[(\tilde{\boldsymbol{y}}-(\mathbb{E}[\tilde{\boldsymbol{y}}])]+\boldsymbol{\sigma^2}$$

$$\boldsymbol{bias^2}+\boldsymbol{variance}+2(\boldsymbol{f}-\mathbb{E}[\tilde{\boldsymbol{y}}])\mathbb{E}[(\tilde{\boldsymbol{y}}-\mathbb{E}[\tilde{\boldsymbol{y}}])]+\boldsymbol{\sigma^2}$$

$$\boldsymbol{bias^2}+\boldsymbol{variance}+2(\boldsymbol{f}-\mathbb{E}[\tilde{\boldsymbol{y}}])(\mathbb{E}[\tilde{\boldsymbol{y}}]-\mathbb{E}[\tilde{\boldsymbol{y}}])+\boldsymbol{\sigma^2}=\boldsymbol{bias^2}+\boldsymbol{variance}+\boldsymbol{\sigma^2}$$

2.4.2 kFold cross-validation

Kfold cross-validation is another resampling technique we will use to evaluate the performance of the model, however this time with the focus on how to split our data into training and test datasets. This method works on the bais of splitting the training data into k more or less equally sized mutually exclusive datasets. We are going to split the out dat for varying values of k, and begin by deciding which of the groups will play the role of a test set. We will the proceed by using the remaining groups in the training set. We will then fit our model and evaluate it on the chosen test set. This process will be repeated until each of the groups has been used as a test set, and ensures a balanced representation of each sample in both the training and the test set over the splits.

2.5 The Datasets

In order to test our models and the statistical tools of resampling, we generate a polynomial function *f(x,y)* an **x** and **y** dependence on the form [x,y,x^2,y^2,xy,...] and try to fit it to the well known Franke function, which is a weighted sum of four exponential reading as follows

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

After testing our model on the Franke function, we will proceed by reading a **GeoTIF** file containing geographic information a given terrain in Norway. After succesfully extracting the data, again generate a set of polynomials which we will use to model the topograhic data and compare the models against each other

2.6 Preprocessing of Data

Thorughout this project we are going to scale training data using the method of zero centering, while simultaneously removing the intercept from the fitting process, and adding it back again when we perform our predictions. The removal of intercept from the fitting process is rooted in the assumption that the expected outputs of the models are not zero when all the predictors are zero. Thus, it may be a bad idea to implement a model which penelizes the intercept

3 Results

4 Conclussion and discussion of results

5 Appendix A - Analytical Solution of OLS

Now let us assume that there exists a continous function *(*f(x)) with a normally distributed error ϵ *N(0,*σ^2*)* which describes our data:

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

Function f(x) has been approximated through our model $\tilde{y}$, where we minimized the *Residuals sum of squares* $(y - \tilde{y})^2$, where:

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

As we know, $\mathbf{X}$ is our design matrix containing all of the independent variables $\mathbf{x}$ used to approximate $\mathbf{y}$. We are now going to show that the expectation value of $\mathbf{y}$ for any given element i can be written in the following way:

$$\mathbb{E}[y] = \Sigma_j x_i \beta_j = X_{i,*}\beta$$

Let us start the proof with the element by rewriting the expactation value of $\mathbf{y}$:

$$\mathbb{E}[y] = (1/n) * \Sigma_j y_i = (1/n) * \Sigma_{i=0}(f(x_i) + \epsilon_i)$$

Now we see that in order to prove out that $\mathbb{E}[y]$ is equal to the product $X_{i,*}\beta$, we need to prove that the value of $\epsilon_i = 0$. We can easily do it by finding the first derivative of the cost functions MSE:

$$\frac{\partial C(\beta)}{\partial \beta} = 0$$

As you can see, we set the derivative equal to zero in order to find the optimal parameters that will minimize our error.

$$X^T(y - X\beta) = X^T y - X^T X = 0$$

Now if this matrix $X^T X$ is invertible, which it is only if X is orthonormal, then with little algebra, we have the following solution for the optimal parameters:

$$\beta = (X^T X)^{-1} X^T y$$

Now in the situation where $X^T X$ is invertible, the error which we try to minimize will be equal to zero:

$$\epsilon = y - \tilde{y} = y - X(X^T X)^{-1} X^T y = y - y = 0$$

If you pay attention however, we could've from the start assumed that the value of $\epsilon = 0$, and written the proof in the following way:

$$\begin{aligned} \mathbb{E}[y_i] &= \mathbb{E}[X_{i,*} + beta] + \mathbb{E}[\epsilon_i] \\ &= X_{i,*} beta + 0 = \mathbb{E}[y] = X_{i,*} beta \end{aligned}$$

This is simply caused by $\mathbb{E}[\epsilon_i]$ being by definition equal to zero, as it can be interpreted as the mean value of the error. Since the the mean value of the distribution of ϵ is equal to zero, we can write $\epsilon_i = 0$. Now the next thing we are going to prove, is that the variance of y_i is equal to σ^2. From the lecture notes and *Pattern Recognition and Machine Learning by Christopher M. Bishop*, we know that the equation giving us variance, can be written in terms of a expectation value:

$$\begin{aligned} Var(y_i) &= \mathbb{E}[y_i - \mathbb{E}[y_i]] = \mathbb{E}[y_i^2] - (\mathbb{E}[y_i])^2 \\ &= \mathbb{E}[(X_{i,*}\beta + \epsilon_i)^2] - (X_{i,*}\beta)^2 \\ &= \mathbb{E}[(X_{i,*}\beta)^2 + 2\epsilon_i X_{i,*}\beta + \epsilon_i^2] - (X_{i,*}\beta)^2 = (X_{i,*}\beta)^2 + 2\mathbb{E} \end{aligned}$$