Regression analysis and re-sampling methods

Johan Nereng

Department of Physics, University of Oslo, Norway

Sep 30, 2019

Abstract

1 Introduction

The main aim of this project is to study in more detail various regression methods, including the Ordinary Least Squares (OLS) method, Ridge regression and finally Lasso regression, including use of resampling. Start by fitting polynomials to Franke's function (see Methods).

Having established the model and method, the models are further evaluated using resamling techniques such as cross-validation.

choice of modelling based on insight. Terrain possbily polynomial ? use franke function to test algorithms.

Linear regression models the relationship between the response or target and the predictors linearly. In this project, three different methods are applied to fit the model to the data. Ordinary Least Squares (OLS), Ridge, and Lasso. Since this project assumes that the

Trying to find the relationship between an independent and dependent variables chose approximation ,a..a.s

it is normal to use y for predictors in litteratur,and x0 x1 etc. but for shorthand purposes, x1x2 are called xy, thus z=y in this project.

In general, re-sampling techniques use a limited training set to perform multiple model fittings, treating a subset of the available data as training data and another subset as test data. The purpose of this is to obtain additional information not available when doing only one model fit [5][p.175]. There are many such techniques, such as *cross-validation* and *bootstrap*, of which cross-validation is used in this project.

Project flow: This project starts by introducing

Main findings

2 Methods

2.1 Multiple Linear Regression

Usually, insights into the underlying mechanisms of the origins of a data set, will guide the the choices one makes when trying to model the relationship between the dependent and independent variables. If such insights suggest a linear relationship between response and target, the model describing that relationship should reflect this. When modeling for one response, this means using multiple linear regression, while for more than one response, this would mean using multivariate linear regression. A response y, with approximation $\tilde{y}$, on predictor x with a $m-1$ degree linear approximation may be expressed as

$$z = \sum_{j=0}^{m-1} \beta_j f_j(x) + \epsilon = \tilde{z} + \epsilon \tag{1}$$

, where ϵ is the residual error between the model and the true response [7][p.19]. When the target-predictor relationship can be modeled by power functions, (1) leads to a polynomial fit, ie $f_j(x) = x^{j-1}$. For a data set $\{z_i, x_i\}_{i=0}^{n-1}$, where y_i is the response on predictor x_i, (1) results in n equations. In order to effectively solve for the regression coefficients (β_j), the set of equations may be represented by a the matrix-vector multiplication. For a polynomial of order $m-1$, $\boldsymbol{\beta} = [\beta_0, \beta_1, ..\beta_{m-1}]^T$, and for $n-1$ data points, $\boldsymbol{z} = [z_0, z_1, ..z_{n-1}]^T$, while $\epsilon = [\epsilon, \epsilon, ..\epsilon n-1]^T$. Lastly, the $n \times m$ matrix

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{m-1} \\ 1 & x_1 & x_1^2 & \dots & x_1^{m-1} \\ & & \dots\dots\dots\dots & & \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^{m-1} \end{bmatrix} \tag{2}$$

, known as a Vandermonde Matrix [1][p. 147-148], yield the vector matrix product:

$$\boldsymbol{z} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{3}$$

An outtake of the methods which may be applied to (3) in order to solve for the regression coefficients are discussed later. However, the Vandermonde Matrix - which is a special case of what is generally known as a design matrix - is limited to a univariate polynomials. If instead the response z_i is on two predictors (x_i and y_i), such as will later be used in this project, a bivariate polynomial approximation of z will result in a design matrix $\boldsymbol{X}$ where row i is on the form $[1, x_i, y_i, x_i^2, y_i^2, x_i y_i ...]$, with a total of $d = \binom{m+2}{m}$ coefficients.

2.1.1 Ordinary Least Squares (OLS)

OLS fits a function by minimizing the sum of the squares of the errors between a model and a data set. This results in the cost function

$$C(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 = \frac{1}{n}(\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{\beta}) \tag{4}$$

which, if A has full column rank, has a unique solution [1][p.144] which satisfies the normal equations:

$$(\boldsymbol{X}^T\boldsymbol{X})\boldsymbol{b} = \boldsymbol{X}^T\boldsymbol{y} \tag{5}$$

Solving this corresponds to finding the minima of the derivative of the cost function with respect to $\boldsymbol{\beta}$:

$$\boldsymbol{\beta}^{OLS} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{A}^{\dagger}\boldsymbol{y}. \tag{6}$$

This matrix inversion may then be carried out using for example SVD or LU decomposion. Alternatively, minimizing (4) may be accomplished using orthogonal transformation with QR decomposition, the details of which may be found in chapter one of Uri M. Ascher and Chen Grief's book [1]. The choice between the two approaches hinges on computational stability versus computational speed.

Computational Speed: The two approaches are as mentioned dissimilar in computational cost, measured in floating point operations (flops) - especially in overdetermined cases, where the matrix $\boldsymbol{X} \in \mathbb{R}^{n,m}$, has $m < n$. The normal equation solution uses $mn^2 + (1/3)n^3$ flops [1][p.146], while the QR decomposition requires $2mn^2 - (2/3)n^3$ flops [1][p.155]. Which means that the normal equations method is significantly faster than the QR approach.

Computational Stability: The conditioning of the two approaches, or stability with respect to input perturbation, are measured in conditioning number of the matrix representing the problem, $K(\boldsymbol{X})$. Higher $K(\boldsymbol{X})$ means less stable.

$$K(\boldsymbol{X}) = ||\boldsymbol{X}^{-1}|| \times ||\boldsymbol{X}|| \tag{7}$$

In order to acquire the normal equations, the product of $\boldsymbol{X}$ and it's transpose is used. Hence, it's conditioning number goes as $K(\boldsymbol{X}^T\boldsymbol{X}) = K(\boldsymbol{X})^2$. This squaring of the conditioning number is not required for the QR approach, which means that the QR method has a lower conditioning number, and is thus the more stable method.

2.2 Shrinkage methods

Alterations to the OLS cost function (4) may be done in order to assert some control of the bias variance trade off. Specifically, in order to reduce the variability of prediction errors associated with with fitting complex models. Shrinkage methods introduce a penalty parameter λ, which effectively shrinks the regression coefficients, thus decreasing bias towards towards data used in training the model. In this project, two such shrinkage methods are used, Ridge Regression and Lasso Regression.

2.2.1 Ridge Regression

The Ridge Regression cost function [6][p.22] minimizes the square of the residual sum with a quadratic penalty term:

$$C(\boldsymbol{\beta}) = \frac{1}{n}(\boldsymbol{z} - \boldsymbol{X}^T\boldsymbol{\beta})^T(\boldsymbol{z} - \boldsymbol{X}^T\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta} \tag{8}$$

, where $\lambda \geq 0$ is the penalty constant. As is evident from the equation, $\lambda = 0$ results in a solution similar to OLS. Higher values of λ results in increased shrinkage, and in fact each value λ will produce a different set of coefficient estimations [5][p.215]. This parameter may decrease variance when compared to OLS, especially when the dependency of the response is close to linear on the predictors and the number of coefficients is high. In such scenarios, minor perturbations in training data may cause large changes in the OLS estimated coefficients [5][p.219]. Originally, the shrinkage parameter was introduced to address rank deficiency [3][p.64] - which it does, as the method adds a non-zero constant to the diagonal elements.
The cost function (8) results in the following expression for the regression coefficients:

$$\boldsymbol{\beta}^{Ridge} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{z} \tag{9}$$

RIDGECOST?

2.2.2 Lasso Regression

Lassso Regression is similar, but not identical, to Ridge Regression. Instead of a quadratic penalty term, Lasso Regression uses the absolute value of the coefficients [4]

$$C(\boldsymbol{\beta}) = \frac{1}{n}(\boldsymbol{z} - \boldsymbol{X}^T\boldsymbol{\beta})^T(\boldsymbol{z} - \boldsymbol{X}^T\boldsymbol{\beta}) + \lambda||\boldsymbol{\beta}|| \tag{10}$$

The minimization problem based on this cost function has solutions which are non-linear to $\boldsymbol{y}$, and there is, in contrast to Ridge Regression, no closed form solution [3][p.68]. Thus, to determine model coefficients by Lasso Regression, a computing algorithm must be used. In addition, where Ridge Regression does not set any coefficients to exactly zero [3][p.68], Lasso Regression may do exactly that, which means that Lasso Regression may eliminate certain input parameters. This means that Lasso Regression produce *sparse* models - models which may only include a subset of the variables in the data set [5][p.219]. A consequence of this is that Lasso Regression generally produce models that are easier to interpret than Ridge Regression and OLS.

A general comparison between Ridge and Lasso may be studied further in literature, such a [5][p.223-224], on which the following is based. Simply put, Ridge regression leads to better prediction when performed on data where there are few predictors of roughly the same importance. However, for a large set of predictors, where a smaller subset of predictors are significantly stronger linked to the response than the rest, Lasso performs better. Especially so when the response has low dependency

on the remaining subset.In this regard, it is however important to note that when analyzing a real data set, the number of significant predictors is unknown a priori. Therefor, in order to determine the method of regression, the use of re-sampling techniques such as cross validation or bootstrap is advisable.

2.3 Re-sampling procedure: k-Fold Cross-Validation

As other re-sampling procedures, the k-Fold Cross-Validation (KfCV)uses a limited training set to perform multiple model fittings in order to obtain additional information about model parameters [5][p.175]. In the case of this specific procedure, this is achieved by splitting the data into k, approximately equaled sized groups, normally after shuffling the data sets. Then, the first of those k sets is treated as a test set, or validation set, while the remaining $k-1$ sets are used to fit the model. This is then repeated k times, using a new subset for validation in each iteration [5][p.181]. In each iteration$i$, the mean square error, MSE_i (12), is calculated using the validation group as $\boldsymbol{z}$. After all iterations are complete, the Cross-Validation estimate is computed by:

$$CV_(k) = \frac{1}{k}\sum MSE_i \tag{11}$$

One of the applications of this procedure is finding a suitable value for the shrinkage parameter λ. Rather than For each value of a variety of λ-values, a KfCV is carried out, and the $CV_(k)(\lambda)$ computed. Thus, the λ which produced the lowest $CV_(k)(\lambda)$ may be applied in the finished regression based on the original data set.

2.4 Model evaluation

There are, as outlined above, several regression models to chose from. These models all have parameters that impacts their predictions, such as the number of polynomial coefficients and the regularization parameter. As there are many choices, a measurement of how well a regression model predicts the response corresponding to a real data point is needed. One such measurement is the mean square error:

$$MSE = \frac{1}{n}\sum_{i=0}^{n-1} \tag{12}$$

This is a natural measurement to use, both because it's simple and because the cost function is based on this measurement when solving the normal equations. However, the MSE is data specific. An alternative is using the R^2, which is a normalized measurement:

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

, where $\bar{y}$ is the mean value of $\boldsymbol{y}$. As it is independent of scale, R^2 always returns a value between zero and one, making it easy to interpret.
Another way to measure how well the model works is by finding the confidence intervals for the regression coefficients. Following the train of thought from [3][p.47-49]: When using the normal equations for OLS, these may be found by using an analytic expression for the variances of the coefficients

$$Var(\boldsymbol{\beta}) = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\hat{\sigma}^2 \tag{14}$$

Where if σ is unknown, an unbiased estimate $\hat{\sigma}$ should be used. Having obtained the variance of the regression coefficients, the 95% confidence interval (CI) of β_i may be found by

$$CI_{0.95}(\beta_i) = \beta_i - 19.6 v_j^{1/2}\sigma, \beta_i + 19.6 v_j^{1/2}\sigma \tag{15}$$

Where v_j is the diagonal element of row j of the matrix $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$.

2.4.1 Bias Variance trade-off

For a response $\boldsymbol{y} = \boldsymbol{f} + \boldsymbol{\epsilon}$, with $\mathbb{E}[\boldsymbol{\epsilon}] = 0$ and $Var[\boldsymbol{\epsilon}] = \sigma^2$, the expected mean square error (MSE) of a regression model fit $\tilde{\boldsymbol{y}}$ from an input may be calculated by (see appendix 1):

$$\mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right] = \frac{1}{n}\sum_i (f_i - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2 + \frac{1}{n}\sum_i (\tilde{y}_i - \mathbb{E}\left[\tilde{\boldsymbol{y}}\right])^2 + \sigma^2 \tag{16}$$

$$= Bias^2(\tilde{\boldsymbol{y}}) + Var(\tilde{\boldsymbol{y}}) + Error \tag{17}$$

As indicates above, the first term is the squared bias. This corresponds to the average of how much the predicted response, $\mathbb{E}\left[\tilde{\boldsymbol{y}}\right]$, deviates from the true mean of the real response, f_i. The second term is the variance, or the squared expected difference of the predictions about their mean [5][p.223]. When fitting a model to a particular data set, increased complexity or flexibility will tend to make highly accurate predictions on the data set. This corresponds to having a low bias. Simultaneously, such a model fit would have very high variance - or to put it differently, if small variations were made in the data set, the resulting model would drastically change. On the other hand, if the model does not predict accurately on it's own training data, the bias would be high, while the variance would be low - perturbations in input would result in only minor changes of the regression.

2.5 Franke's function

A popular function for evaluation of polynomial approximations is the The Franke's function. Named after Richard Franke's, and published in his report "*A Critical Comparison of Some Methods for Interpolation of Scattered Data*" [2], the function is a weighted sum of four exponential;

$$\begin{aligned} f(x,y) = &\frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) + \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) \\ &+ \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) - \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right) \end{aligned} \tag{18}$$

, defined on the domain $x, y \in [0, 1]$.

2.6 Code implementation

The regression methods are implemented into a program using Python 2.7, using numpy and Scikit-learn. The code is structured around a program which contains a data class, with methods for different regression procedures, as well as data generation and data reading. The results presented in the following section are produced using their own smaller programs, which draw on the classes from the main program. The class method for Lasso uses the Scikit-learn, while OLS and Ridge do not. Additionally, the k-Fold CV draws on some utility from Scikit-learn, such as dividing folds.

2.6.1 Initial evaluation of OLS implementation

In order to make an initial evaluation implementation of the OLS regression, terrain data is generated using Franke's Function (18), with 30×30 grid points. One data set with no noise, one with noise $\sim N(0, 1)$ and one with $\sim N(0, 0.3)$. Polynomials of up to fifth order are directly fitted using the entire data set, and the accuracy of the model fittings are recorded in terms of MSE and R^2, as well as 95% confidence interval for the regression coefficients.

3 Results

3.1 Initial evaluation of OLS implementation

Below, MSE and R^2 values for the initial evaluation of the OLS is presented in table 3 for polynomials up to degree five, using the 30×30 Franke's Function generated data set with noise $\sim N(0, 0.3)$. Similar tables for noise $\sim N(0, 1)$ and no noise can be foind in appendix 2. These tables clearly show that both R^2 values and MSE decrease with increased model complexity up to the fifth degree, and that both measurements increase with increased noise.

Polynomial degree	R^2	MSE
1	0.3521	0.1169
2	0.3947	0.1093
3	0.4519	0.0989
4	0.4904	0.0920
5	0.5032	0.0897

Table 1: MSE and R^2 scores for OLS on Franke's Function generated responses using a 30×30 grid with noise $\sim N(0, 0.3)$

Figure 4 shows the spread of confidence intervals for a 1. order up to 5. order fitting using OLS on the 30×30 Franke's Function generated grid with noise $\sim N(0, 0.3)$.

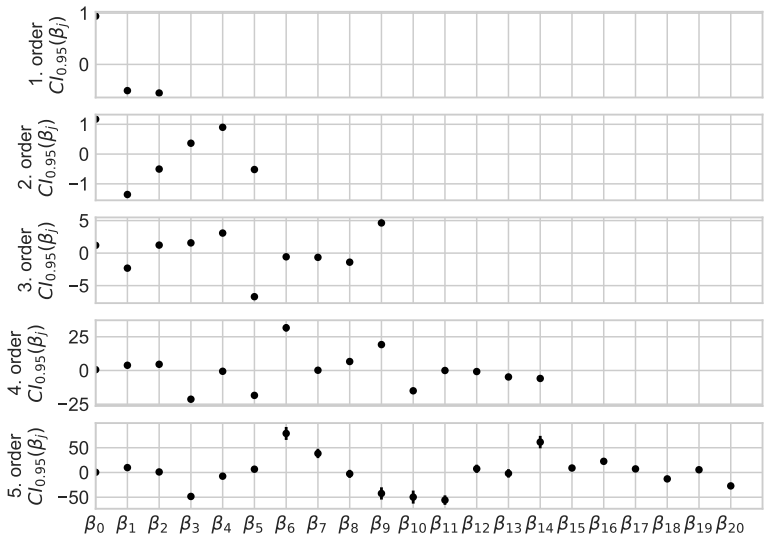

Figure 1: 95% confidence intervals for the OLS regression coefficients for polynomials of order one to five, using the Franke's Function generated response on a 30×30 grid

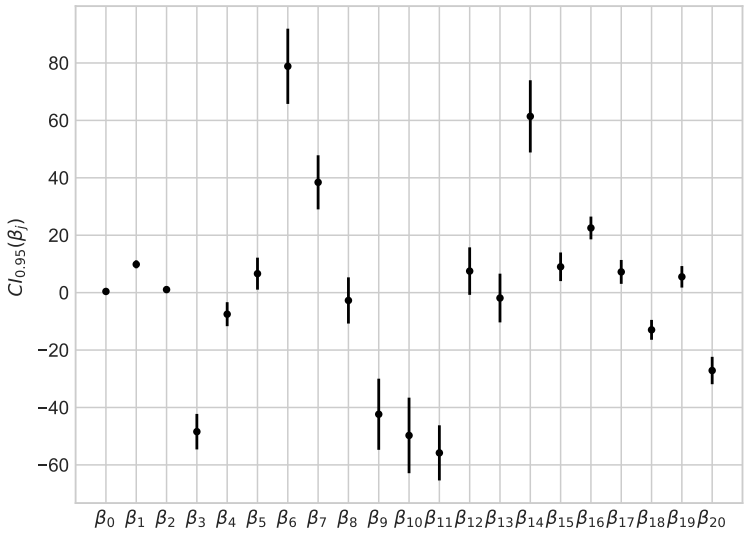

Figure 2: 95% confidence intervals for the OLS regression coefficients for a polynomial of order five, using the Franke's Function generated response on a 30×30 grid

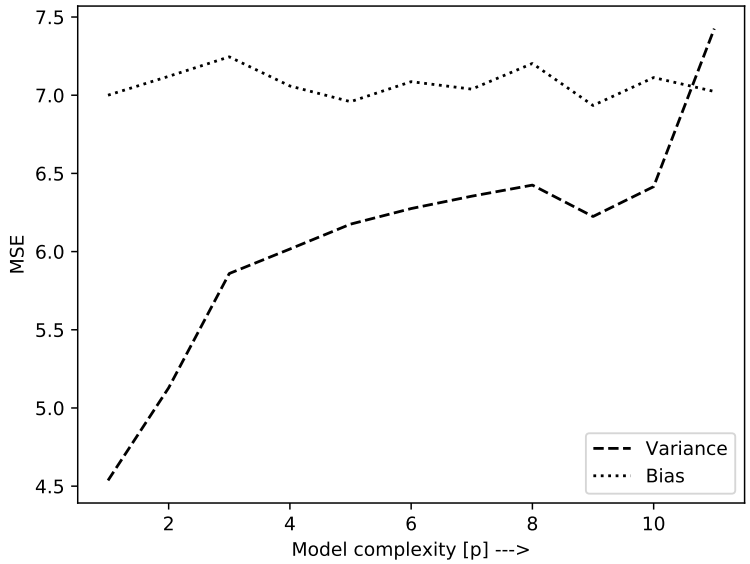

Figure 3: Comparison of Bias vs Variance, using the Franke's Function generated response on a 30×30 grid, averaged over 80 runs, as a function of model complexity in terms of polynomial degree, p

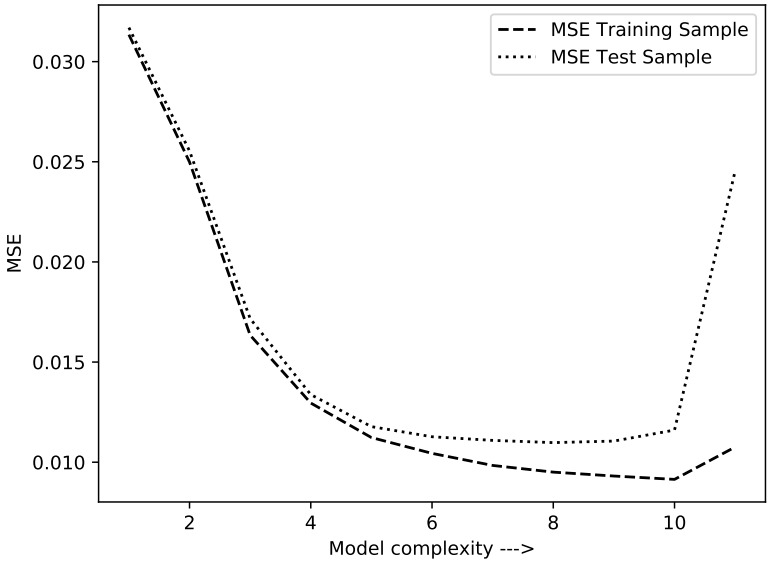

Figure 4: Comparison of MSE for training and test data, using the Franke's Function generated response on a 30×30 grid, averaged over 80 runs, as a function of model complexity in terms of polynomial degree, p

4 Discussion of results

0.3 noise used to illustrate regression with noise, without having so much noise that the regression runs the risk of not fitting very well at all.

4.1 Conclusions

References

[1] Uri M. Ascher and Chen Greif. Linear least squares problems. In *A First Course on Numerical Methods*, pages 141–166. Society for Industrial and Applied Mathematics, 2011.

[2] Richard Franke. A critical comparison of some methods for interpolation of scattered data. Technical report, Naval Postgraduate Sschool Montery CA, 1979.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer New York, New York, NY, second edition edition, 2009.

[4] Morten Hjorth-Jensen. Lectures notes in fys-stk4155. data analysis and machine learning: Linear regression and more advanced regression analysis, Sep 13 2019.

[5] Gareth James. *An Introduction to statistical learning : with applications in R*. Springer texts in statistics. Springer, New York, corrected at 8th printing 2017. edition, 2017.

[6] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G.R. Day, Clint Richardson, Charles K. Fisher, and David J. Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Physics Reports*, 810:1–124, 2019.

[7] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. Adaptive computation and machine learning. MIT Press, Cambridge, 2012.

Appendices

Appendix 1.

Following [3][p.223] følg https://stats.stackexchange.com/questions/204115/understanding-bias-variance-tradeoff-derivation

Appendix 2.

Polynomial degree	R^2	MSE
1	0.7405	0.0176
2	0.8140	0.0126
3	0.9115	0.0060
4	0.9561	0.0030
5	0.9840	0.0011

Table 2: MSE and R^2 scores for OLS on Franke's Function generated responses using a 30×30 grid without noise

Polynomial degree	R^2	MSE
1	0.0865	0.9102
2	0.0999	0.8968
3	0.1099	0.8868
4	0.1145	0.8822
5	0.1285	0.8683

Table 3: MSE and R^2 scores for OLS on Franke's Function generated responses using a 30×30 grid with noise $\sim N(0, 1)$