Project 1 in FYS-STK4155

Adrian Martinsen Kleven
Simon Elias Schrader

Autumn 2020

Contents

List of Figures

List of Tables

1 Abstract

In this project, we use some good methods and some shitty methods to test how well a polynomial can interpolate a superfancy exponential function (protip: It can't) . Then we use the same approaches to see how well random geographic data can be approximated. We expect this to look horrible. We're sorry for abusing math in that way. We shall contemplate Seppuku.

Figure 1: Some random picture.

2 Introduction

3 Methods

3.1 Linear Regression

The aim of Linear Regression is to find the conditional distribution $p(y|\boldsymbol{x})$ for y given $\boldsymbol{x}$. In Linear Regression, a linear functional dependence is assumed, and the aim is hence to create a linear function that fits a set of p predictor variables $\boldsymbol{x}$ to a target variable $\boldsymbol{y}$, where there are n discrete observations for each of the p predictor variables and the target variable. That is, the aim is to find a functional relationship of the form $f(\boldsymbol{x}) = y$.
We assume hence that each target variable y_i has a functional dependence of the predictor variables:

$$y_i = \tilde{y}_i + \epsilon_i = \beta_0 + \beta_1 x_{i0} + \beta_2 x_{i1} + ... + \beta_p x_{i(p-1)} + \epsilon_i$$

As this is true for all y, this can be written in matrix form:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where $\boldsymbol{y}$, and $\boldsymbol{\epsilon}$ are vectors of length n, $\boldsymbol{\beta}$ is of length p and $\boldsymbol{X}$ is a matrix of size $n * p$. X is called the design matrix. We now want to find fitting parameters $\boldsymbol{\beta}$ that represent the best linear fit of y to the input data $\boldsymbol{x}$. Hence, we want the fit

$$= \boldsymbol{X}\boldsymbol{\beta}$$

to be as close to the target data $\boldsymbol{y}$ as possible. There are several approaches to define the best linear fit, and in this article, we looked at three different methods that define the best fit in a different way each. All methods have in common that they define a loss function, which describes the deviation from the fit to the real data, that needs to be minimized.

3.1.1 Ordinary Least Squares Regression

In Ordinary Least Square regression, the loss function is defined as the following:

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

The solution minimizing this problem is found by deriving $C(\boldsymbol{\beta})$ with regards to each β_i and setting each derivative equal to zero. Doing this, we end up with the following equation (in matrix notation)

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

which gives us

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

which can be solved when $(\boldsymbol{X}^T\boldsymbol{X})$ is invertible.

Under the assumption that true data y is given as

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

where $\boldsymbol{\epsilon}$ follows a normal distribution $N(0, \sigma^2)$, information about the variance and the expectation value can be deduced. Assuming that $f(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{\beta}$, it can be shown that

$$\mathbb{E}(y_i) = \boldsymbol{X_{i,*}}\boldsymbol{\beta}$$

$$Var(y_i) = \sigma^2$$

where $\boldsymbol{X_{i,*}}$ is the i-th row of the design matrix.

For β, it can be shown that

$$\mathbb{E}(\boldsymbol{\beta}) = \boldsymbol{\beta}$$

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

3.1.2 Ridge Regression

3.1.3 LASSO Regression

3.2 Resampling Methods and Data Splitting

In order to get an approximate estimate of the quality of a fit, it is neccessary that the fit is tested on a set that it has not been trained on. As the training set determines the parameters $\boldsymbol{\beta}$, we need a set from the same distribution as the training set, which is different from the training set. For that reason, the full data set of predictors and targets is split into a test set and a training set. The test usually conatins $20 - 25\%$ of the full set, while the training set contains $75 - 80\%$. It is also possible to use a further validation set to do a final estimate using data that has never been "seen" by the programmer or the code, however, this was not implemented in this article.

3.3 Bias-variance relationship

In order to evaluate the quality of a fit, it is customary to evaluate the Mean Square Error (MSE) of the test data to the prediction. The Mean Square Error is defined as

$$MSE(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = n^{-1}\sum_{i}^{N}(y_i - \tilde{y}_i)^2 = \mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right]$$

where the left formula is the within-sample mean square error. The Mean Square Error can be decomposed into three parts: The unavoidable Error σ^2, the squared Bias (which is the deviation of the mean value of an estimator and the true value), and the Variance (which is a measure of the spread of the estimator). Hence

$$MSE(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = Bias[\hat{\boldsymbol{y}}]^2 + Var[\tilde{\boldsymbol{y}}] + \sigma^2 = \left(f - \mathbb{E}(\tilde{f})\right)^2 + \mathbb{E}\left(\left[\mathbb{E}\left[\tilde{f}\right] - f\right)^2\right] + \sigma^2$$

A proof of this decomposition can be found in the appendix. In this article, we will however calculate the following sizes:

$$Bias[\tilde{\boldsymbol{y}}]^2 = n^{-1} \sum_i^N (y_i - avg(\tilde{\boldsymbol{y}}))^2$$

$$Var[\tilde{\boldsymbol{y}}]^2 = n^{-1} \sum_i^N (\tilde{y}_i - avg(\tilde{\boldsymbol{y}}))^2$$

This is because the true values f_i are not known and impossible to obtain.
An important observation when fitting the data is that the Bias decreases as the model gets more complex, while the variance increases. When the Bias is large, important relations of the underlying data are not considered. When the Variance is high, small, unimportant features in the training set have too high of an impact. This is called overfitting. The aim is hence to find the model complexity that gives the lowest MSE. This is usually the case around the point where the Bias has the same magnitude as the Variance.

3.3.1 Bootstrap method

The Bootstrap method is a method that can be used to calculate statistical quantities. The main idea is the following: Let n be the size of the training set $Z = (z_1, z_2, ...z_n)$, where each z_i contains p predictors and one target value, that is $z_i = (x_{i,1}, x_{i,2}, ...x_{i,p}, y_i)$. Now, n data points z_i are drawn randomly from Z with replacement. This is done B times, creating B new sets Z'_j. We can now create a fitting model for each of the sets Z'_j. For each of these data sets Z'_j, we can now calculate any variable $\hat{\theta}_j$ given the data. Now, the distribution of $\hat{\theta}_j$ can be used to estimate the distribution of θ and get values of interest, such as the mean value or the variance. In this article, Bootstrap is used to get information about the test MSE, variance and bias in order to evaluate the quality of the fit. This is done by using the B bootstrap sets Z'_j to calculate the fitting parameters β and calculate B fits to the test set. $\tilde{Y_j}$. From these B fits, we can then approximate the test MSE, variance and bias.
TODO: Talk some more about the mathematical validity of Bootstrap.

3.3.2 K-fold Cross validation

3.4 Fitting the Franke function

3.5 Korea Data

4 Computational Implementation

4.1 Finding the ideal parameter λ

In order to find the ideal parameter λ for both LASSO and Ridge Regression, we looped over a set of lambda-values in different magnitudes and estimated the MSE using 4-fold Cross Validation. We chose to use the parameter λ that gave the lowest MSE. This was repeated for each fit, that is, λ was recalculated for each new Design Matrix X. This is because the ideal λ value changes with model complexity.

4.2 Error in LASSO-Regression

As there is no way to find the parameters ***beta*** for LASSO-Regression analytically, numerical methods need to be used. Here, we used Scikit-Learn in order to solve the problem numerically. We have observed that the LASSO-method converges rather slowly. For this reason, we have

chosen the tolerance to be equal to 0.03 for fitting the Franke function, while the number of maximum iterations is chosen to be 10^5. We observed that decreasing the tolerance to 0.01 lead to no convergence even at 10^5 iterations. Similarly, we used XXX as tolerance at 10^5 maximum iterations for fitting the country data. As this is very time-consuming, we found it impossible to find the ideal parameters ***beta*** for LASSO-Regression. Each time the LASSO-fit was higher than the OLS fit, no ideal parameters have been found, as the ideal LASSO-fit cannot be no larger than the OLS-fit (because $\lambda = 0$ is the OLS fit).

4.3 Sampling Franke Function

The Franke Function is given by

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

and serves as a test function to check how well the implemented methods beform. We sample randomly points x_i and y_i between 0 and 1 and calculate the target points

$$z_i = f(x_i, y_i) + \epsilon_i$$

where ϵ_i follow a normal distribution with standard deviation σ.

4.4 The Design Matrix

The Design Matrix implemented is a 2D-version of the Vandermonde Matrix, hence, approximate z_i as, for a given degree k

$$\tilde{z}_i = \beta_0 + \beta_1 x_i + \beta_2 y_i + \beta_3 x_i^2 + + \beta_2 x_i y_i + ... + \beta_p y_i^k$$

In the practical implementation, we "drop" β_0 by removing the first column from the Design Matrix, as our data is scaled.

4.5 Scaling?

The design matrix is scaled using Scikit-Learn's StandardScaler.

5 Results

5.1 Error comparision

5.1.1 Franke Function

As excpected from the theory, increasing the model complexity lead to a decreased Mean-Square-Error and an increased R2-Score for all three methods when fitting the Franke Function. However, as can be seen in figures XXX and XXX, the mean square error for the Ordinary Least Square approximation suddenly raises sharply (and the R2 score drops sharply) after some complexity is reached: This behaviour cannot be observed with LASSO and Ridge. Another interesting observation is that there is much more fluctuation in the MSE when using bootstrap than when using 5-fold Cross validation. We are not quite sure why this happens, but we assume that one possible cause is that when using bootstrap, some very "undesirable" combinations of values might be chosen, which leads to a very high Bootstrap. Thus, the graphs might look smoother when running more Bootstrap-samples. Another reason is that the training set in the Bootstrap method is not equally large as the training set when using 5-fold cross validation (75% vs 80%). We also observed that Bootstrap is more time-demanding than k-fold Cross Validation.

5.1.2 Map data

5.2 Bias-Variance tradeoff

5.2.1 Franke Function

Figures XXX, XXX and XXX show the MSE, the Training-MSE, the Variance, and the bias for the Franke Function. In all figures, the aforementioned statement that the variance increases and the bias decreases with increasing model complexity, hold true. Especially for the OLS, we see that this resembles figure XXX on page XXX in HASTIE a lot.

5.2.2 Map data

6 Conclusion

7 Conclusion

"I always thought something was fundamentally wrong with the universe" [1]

References

[1] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.