FYS-STK4155 - Project 1

Kari Lovise Lodsby

October 11, 2021

1 Overview

In this project, I will implement and examine three methods for linear regression: Ordinary Least Squares (OLS), Ridge regression and Lasso regression. I will test the methods on both data generated by the Franke function with noise and real terrain data.

2 Exercise 1 - Ordinary Least Square (OLS) on the Franke function

In this exercise, I will use the ordinary least squares method. It is a regression algorithm that aims to minimize the cost function $C(\beta) = \|\mathbf{y} - \tilde{\mathbf{y}}\|_2^2 = \|\mathbf{y} - \mathbf{X}\beta\|_2^2 = \sum_{i=1}^{n} \left| y_i - \beta_0 - \sum_{j=1}^{p} X_{ip}\beta_p \right|^2$,

To find the minimum, we differentiate $C(\beta)$ with respect to β and find the β that yields $\frac{\partial C(\beta)}{\partial \beta} = 0$. Writing $\|\mathbf{y} - \mathbf{X}\beta\|_2^2$ as $(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)$, we find that

$$
\begin{aligned}
\frac{\partial C(\beta)}{\partial \beta} &= \frac{\partial}{\partial \beta}(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) \\
&= -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) \stackrel{!}{=} 0 \\
\Rightarrow \mathbf{X}^T\mathbf{y} &= \mathbf{X}^T\mathbf{X}\beta \\
\beta &= \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}, \qquad (1)
\end{aligned}
$$

Now we can calculate the model prediction as $\tilde{\mathbf{y}} = \mathbf{X}\beta_{\text{optimal}}$.

The design matrix, $\mathbf{X}$, contains a set of linearly independent functions of the predictor, with the columns corresponding to a mapping of the predictors. In this project, I will use the design matrix corresponding to the polynomial basis that consists of every monomial of the predictors x and y up to and including n.

In the code, I scaled the data to remove the mean and scale to unit variance. Some algorithms may behave badly if the data is not standardized, but this is not the case for linear regression, which yields equivalent results if the data is not linearly scaled beforehand. I chose to scale the data anyway because it makes interpretation easier. Additionally, I think it is useful to be in the habit of scaling data.

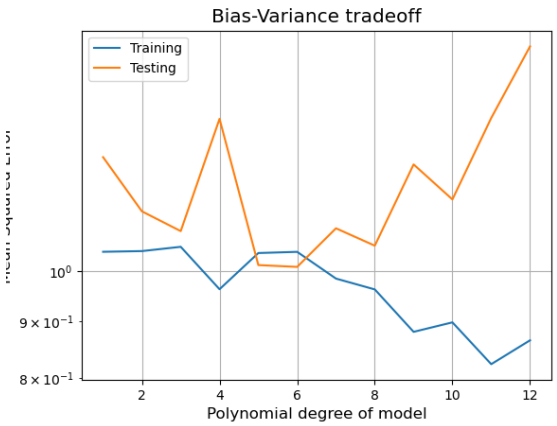
Here is the result:

```
Training data, R2 score: 7.349123e-01

Test data, R2 score: 3.593507e-01

Training data, MSE score: 3.207427e-02

Test data, MSE score 6.822319e-02

Confidence interval score
```

	Lower CI	beta	Upper CI
0	0.427961	0.459093	0.490225
1	-1.578683	0.017861	1.614405
2	-0.143330	2.303160	4.749651
3	-7.651334	0.880638	9.412610
4	-6.736477	-1.750787	3.234902
5	-22.968530	-11.074308	0.819914
6	-24.096473	-5.145763	13.804946
7	-8.025070	0.242605	8.510280
8	-3.168076	7.994602	19.157281
9	-8.746460	15.180625	39.107710
10	-11.794760	7.312945	26.420651
11	-11.037200	-2.547800	5.941599
12	-7.916466	0.188434	8.293335
13	-19.241504	-8.717592	1.806319
14	-30.897428	-8.396978	14.103473
15	-10.510470	-3.298754	3.912962
16	-0.904099	2.799940	6.503978
17	-5.776034	-2.594325	0.587383
18	-1.162817	2.549399	6.261616
19	-2.020244	1.957344	5.934931
20	-6.299929	1.733086	9.766101

3 Exercise 2: Bias-variance trade-off and resampling techniques

In this exercise I perform a resampling of the data where I split it in training data and test data using the function provided by Scikit-Learn. While testing OLS for polynomials of degree 1 up to and including 12, I saw the effect of the bias-variance trade-off.



This graph suggests that as the complexity of the model increases, the prediction error in the training sample decreases, but the prediction error in the test sample decreases only to increase again later. This reflects the fact that if the model is too complex, it will model the training data too closely and not generalize well. A model with low complexity will have high bias and low variance, while it is the other way around for a model with high complexity. We want to choose a level of complexity that provides a good middle ground. Throughout this project, I often find that 5 is a good choice.

To prove the desired equality, we first show this:

$$\mathbb{E}[(\mathbf{y} - \tilde{\mathbf{y}})^2] = \mathbb{E}[(\mathbf{x} + \varepsilon - \tilde{\mathbf{y}})^2] =$$
$$\mathbb{E}[(\mathbf{x} - \tilde{\mathbf{y}})^2] + \mathbb{E}[\varepsilon^2] + 2\mathbb{E}[(f(x) - \tilde{y})\varepsilon] =$$

$$\mathbb{E}[(\mathbf{x}-\tilde{\mathbf{y}})^2]+\mathbb{E}[\varepsilon^2]+2\mathbb{E}[(f(x)-\tilde{y})]\mathbb{E}[\varepsilon]=\mathbb{E}[(\mathbf{x}-\tilde{\mathbf{y}})^2]+\varepsilon^2$$

where σ^2 is the variance of the error ϵ.

Further, we have

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

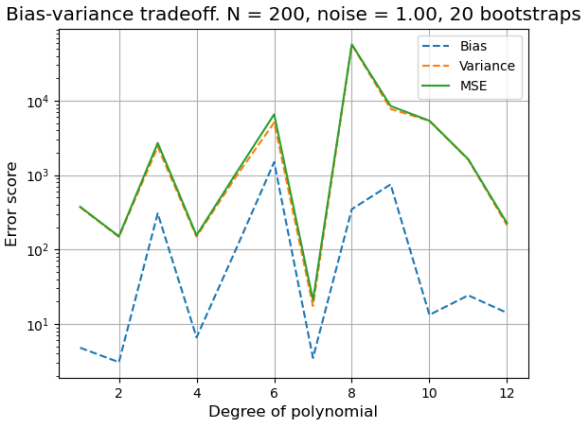
As

$$\mathbb{E}[(f(x)-\mathbb{E}[\tilde{y}])(\tilde{y}-\mathbb{E}[\tilde{y}]))]=(f(x)-\mathbb{E}[\tilde{y}])\mathbb{E}[\tilde{y}-\mathbb{E}[\tilde{y}]))]=$$
$$(f(x)-\mathbb{E}[\tilde{y}])(\mathbb{E}[\tilde{y}]-\mathbb{E}[\tilde{y}])=0$$

the last expression is equal to 0, and we have the bias-variance decomposition - the fist term is the bias squared, and the second term is the variance.

Writing the bias as $\frac{1}{n}\sum_{i=0}^{n-1}(f_i-\mathbb{E}[\tilde{\mathbf{y}}])^2$ and the variance as $\frac{1}{n}\sum_{i=0}^{n-1}(f_i-\mathbb{E}[\tilde{\mathbf{y}}])^2$, we show the desired formula.

In this exercise I also implemented bootstrapping, a method that creates "new" data sets by drawing data from the existing one. This method was used to check the bias.

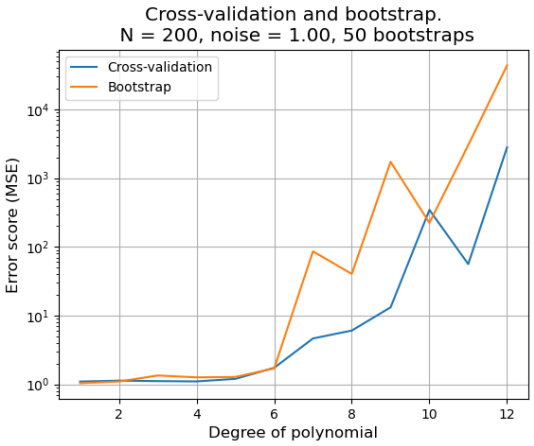


4 Exercise 3: Cross-validation as resampling techniques, adding more complexity

In this exercise I implemented k-fold cross-validation, a method that splits the existing data set to create multiple data sets. I then tried different numbers of folds too what would happen. A simple test of the method shows that the MSE seems to decrease as the number of folds increases. (However, it is possible to have so many folds that the possible number of sample combinations starts to decrease, which makes the method less accurate.)

	# of folds (K)	Mean squared error
0	2	1.344791
1	5	1.248795
2	8	1.276715
3	10	1.228466

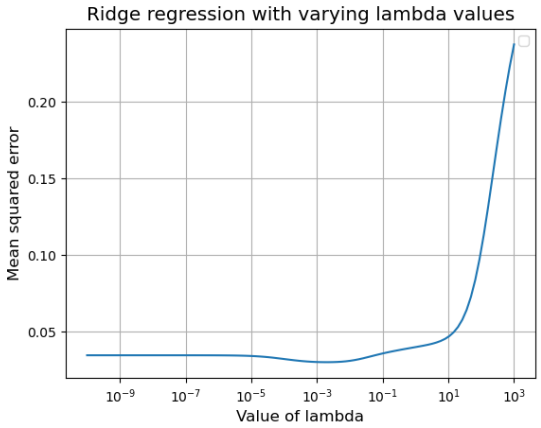
I then compared it to the bootstrap method. Cross-validation is stronger than the bootstrap for model validation. My results support this. (I used 5 folds for the comparison.)



5 Exercise 4: Ridge Regression on the Franke function with resampling

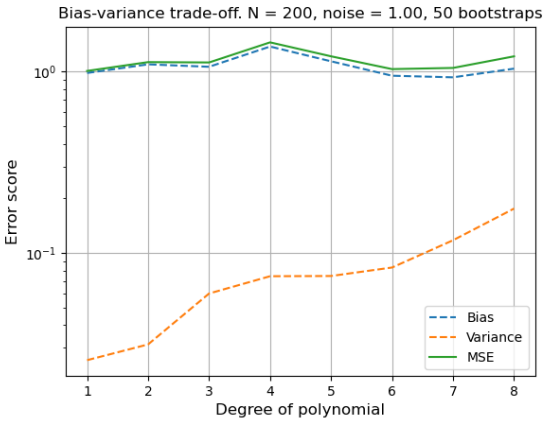
Ridge regression is a regression model that is similar to ordinary least squares, but includes an extra term - while OLS has $\beta = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$, Ridge regression has $\beta = \left(\mathbf{X}^T\mathbf{X} + \lambda I\right)^{-1}\mathbf{X}^T\mathbf{y}$ with a small $\lambda > 0$. This allows it to correct for independent variables which are highly correlated. The Ridge estimator is biased, but has lower variance than the OLS estimator.

First I performed a simple test that shows how the MSE is affected by the choice of λ. This result suggests that the ideal value is low - if λ is too large, it places more importance on the β being small, which leads to underfitting. However, if it is too low, the model becomes close to OLS, which is a worse fit than an optimal Ridge regression.

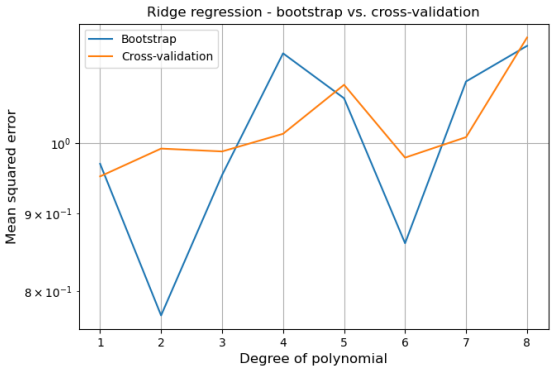


My Ridge regression function tests several values for λ and uses the best one for its output.

My results from applying the bootstrap method indicates that Ridge regression has a bias (even for high complexity), while the variance is lower. This matches what we know about Ridge regression in comparison to OLS.



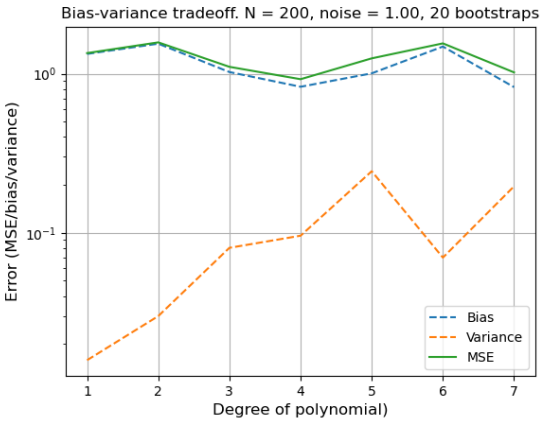
Comparison of bootstrap and cross-validation. The advantage of the latter is less clear here.



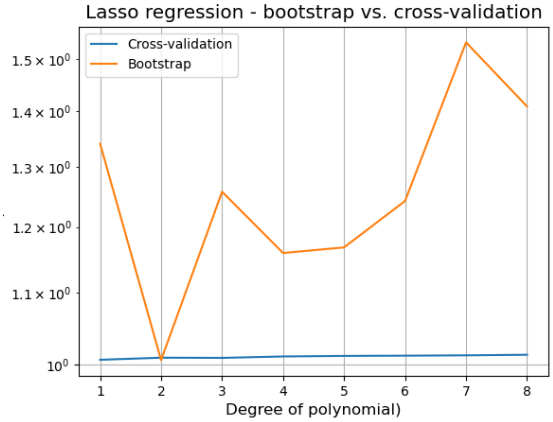
6 Exercise 5: Lasso Regression on the Franke function with resampling

Lasso regression is similar to Ridge regression, but more complicated. It converges irrelevant variable coefficients to 0 and uses regularization to make the results more accurate and easier to interpret.

For this problem I used the Scikit-Learn function instead of writing my own code for the algorithm. I use bootstrapping to look at the bias-variance trade-off:



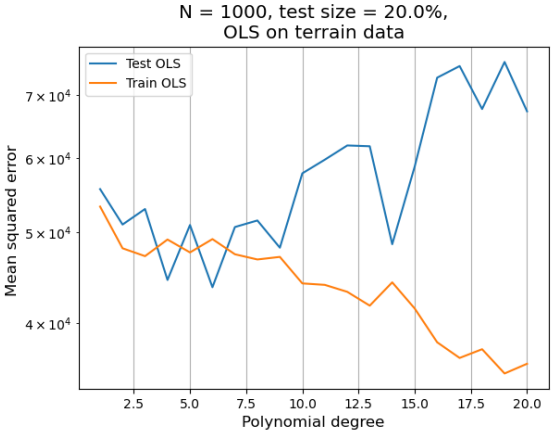
Like Ridge regression, lasso regression trades lower variance for higher bias.



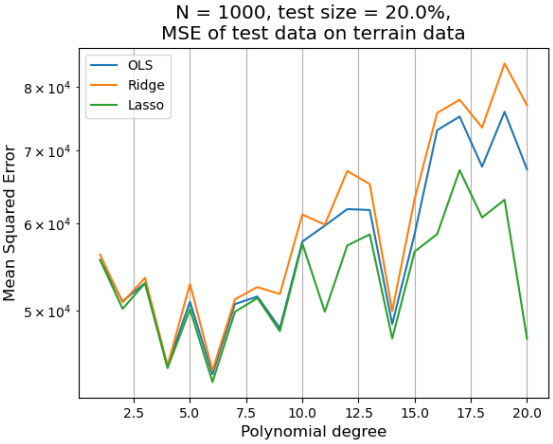
Here, cross-validation seems to perform significantly better.

7 Exercise 6: Analysis of real data

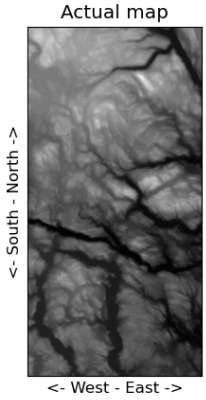
In this exercise I will analyse some real terrain data from Norway.
First I look at the bias-variance trade-off for OLS.



The optimal degree seems to be around 5.



This is the map I wish to model:



The idea is to approximate each row separately using the three methods discussed earlier in this project.

I guess that lasso regression will be the best choice, as it encourages simple models and can make noise less relevant. This is a good choice for terrain data, as this will smear out the noise while leaving the important parts of the terrain intact. (Indeed, lasso regression was originally intrduced in geophysics.)

Unfortunately, I was unable to confirm or deny this because the programs refused to give me useful output.