FYS-STK4155, Autumn 2019, Project 1

Aksel Kristofer Gravir (akselkg)

October 10, 2019

Introduction

The project code can be found at my github: `https://github.com/Akselkg/FYS-STK4155-2019` The code is split into two python files. pro1.py is the "main" file, which sets everything up, and is what you run to execute the project. functions.py contains both helper functions and meatier functions that compute the more complicated methods. Plots and a sample run can be found in the results folder, and the data set I used is in the DataFiles folder.

The main goal for this project was to implement different regression methods and attempt to analyse them through examining statistics from the resulting regression lines. The methods were tested on the Franke function f, and attempted to be applied on some real geographical data. The Franke function is an exponential function that takes two parameters (x, y) as input and looks somewhat like a hilly landscape, making it a good choice for testing. The base regression method is OLS, using polynomials in x and y.

a)

The OLS model is $\tilde{z} = X\beta$, where I used z as my response variable since y is used as a parameter. X is the design matrix, with columns $[1, x, y, x^2, xy, y^2, ...]$, up to order 5. I implemented the model using a matrix inversion by

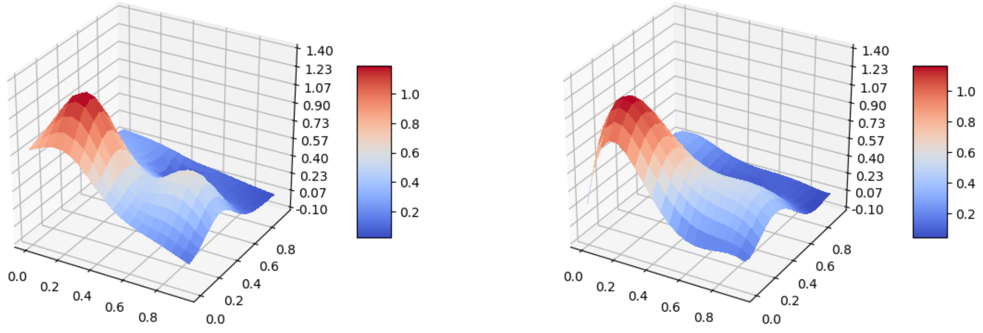
$$\beta = (X^T X)^{-1} X^T z$$

. The base data for the regression are $n = 20$ uniformly distributed x and y values, making 400 uniformly distributed points in the square $[0, 1] \times [0, 1]$ and corresponding true values $z = f(x, y)$. These values will be used throughout unless stated otherwise. This gave me scores $R2 = 0.983$, $MSE = 0.00124$, certainly a good fit. With an added error term of $0.1N(0, 1)$ I got slightly worse values $R2 = 0.873$, $MSE = 0.0109$ as you would expect.

I plotted the randomly distributed points by triangulating them and plotting the resulting surface.

b)

My implementation of k-cross validation lets the order of the polynomial vary for comparison. The dataset is split into k parts, and a regression is ran using each of the k parts as a validation set in turn. This is done for all polynomial maximal orders. The result is values for the training and test error for each maximum order of the polynomial fit. In Fig. **??** you can see the test error deviating, increasing, from the training error as the complexity of the model increases as expected.



(a) Function

(b) Regression

Figure 1: Franke function in comparison to OLS regression surface. $n = 400$, fifth order polynomial, no error.

c)

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

$$= E[(f-E[\tilde{y}])^2] + E[\epsilon^2] + E[(E[\tilde{y}]-\tilde{y}))^2] + \text{cross terms}$$

$$\text{Bias}^2 + \sigma^2 + \text{Variance}$$

I skipped a bit of the calculation here, namely the cross terms. They all individually cancel out, mostly since the expected value of $\epsilon = 0$. In the end we are left with the wanted expression. The bias in the model is represented by the average distance between the model and the real value, and the variance is the expected difference within the model itself. My program does calculate variance and bias in every regression implementation. I was however not able to read as much from them as i wanted. They did not seem to add up to the total MSE in general. My value for the variance was often higher than the MSE!

d

In Fig. **??** there is a clear dip in the test error indicating a benefit in having a λ parameter with a positive value $\approx 10^{-4}$ in this specific case. For a very small lambda ridge is approximately OLS, so we get a distance between the errors like the distance for order$= 5$ in Fig. **??**.

e

Lasso scored a little worse on all metrics. This is probably because i divided the data set in half here. Possibly not the best comparison. But as we'll see lasso did the worst on the terrain data as well. In total pure OLS scored the best with the alternatives in ridge and lasso scoring sligthly worse on MSE, all compared under k-cross validation.

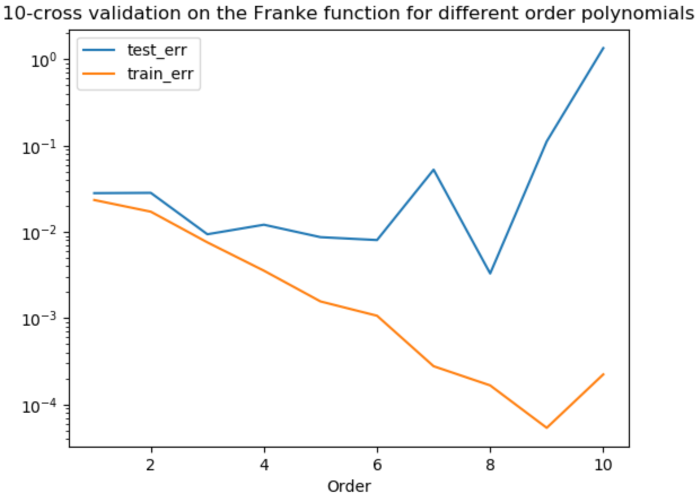


Figure 2: The test error seems quite stable in the order of the polynomial, before $n = 6$.

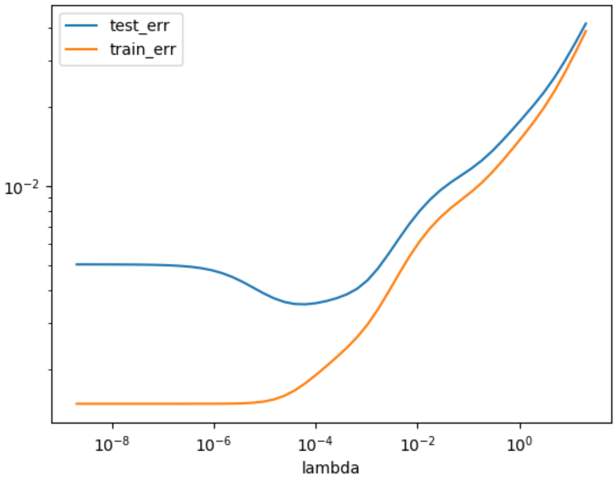
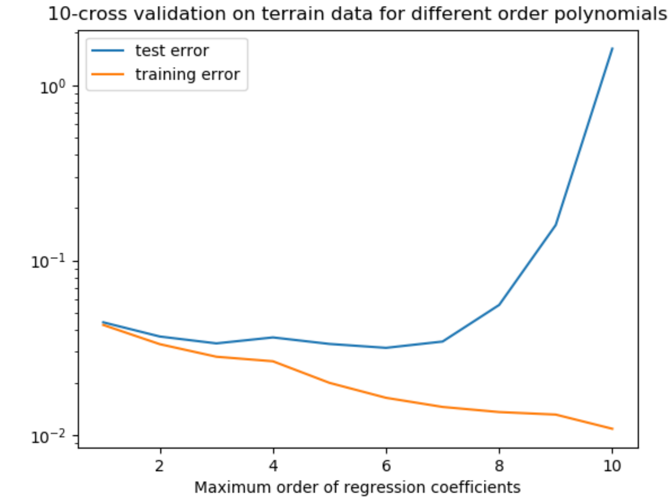
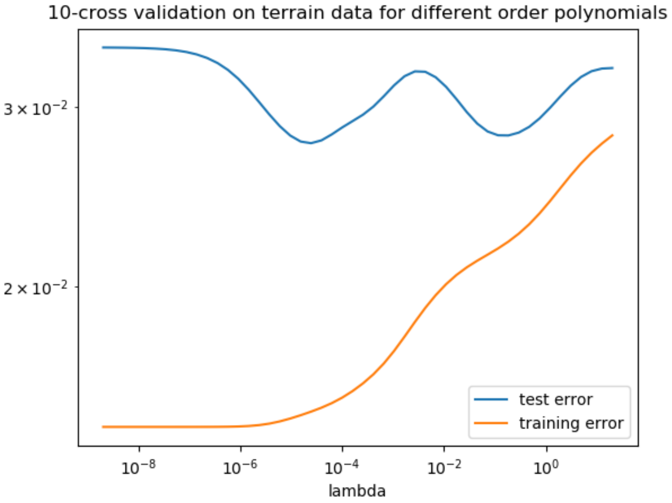


Figure 3: 10-cross validation using ridge regression, examining the effects of changing the λ parameter. There seems to be a minima of the test error at around 10^{-4}.

f, g)

I used the dataset contained in the file SRTM_data_Norway.tif, containing terrain data from the coast of Norway. As is the dataset was to big to carry out regression. If i ran a simple OLS on the set it would complete the calculation but the resulting image had some strange artifacts and looked little like the original. Cross validation would also take too long. My solution was to both look at a smaller area of the image, and also to pick out every fourth point. The area i chose was the last 1000 points in each direction, leaving me with a set of 250 × 250 points. I tried a few other resolutions, but my results did not seem to depend on it too much. In total the image lost a lot of qualites after the regression. For the ridge and lasso implementations the image is almost unrecognisable as can be seen in Fig. 4. OLS keeps the most detail. The statistics from the runs was a little hard to say anything about. K-cross seems to like an order up to 6-7, Fig. **??**. Ridge also seems to be beneficial, with two local minima strangely, Fig. **??**.





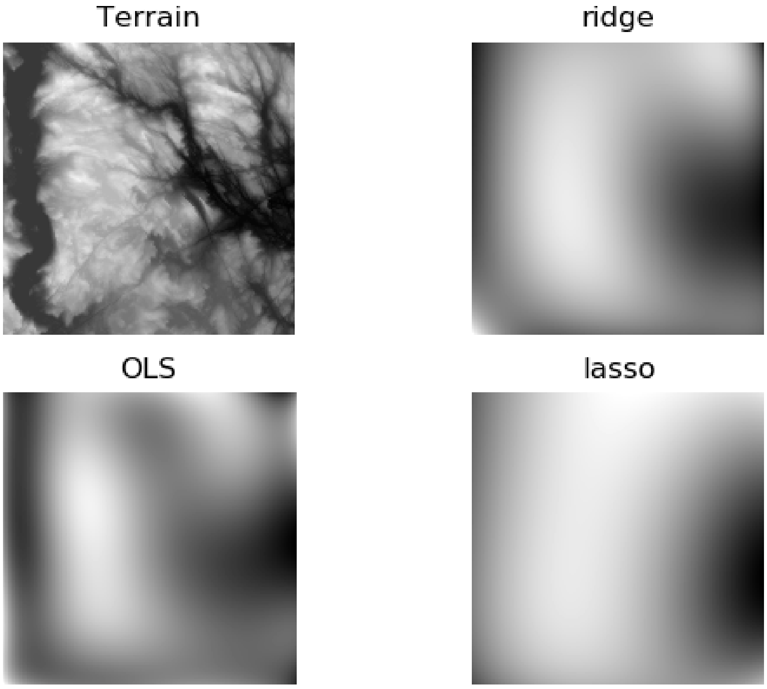


Figure 4: Raw terrain data in upper right. Other three plots are attempted regressions on the set.

Conclusion

I got a few conflicting results. My graphs showed some benefits to be had from the ridge regression, but the measured MSE was consistently lower for it and the lasso. This is also seen in the images of the terrain, where lasso and ridge look much worse in comparison to pure OLS. I'm open to being wrong here! The pure OLS did seem to catch the main details in the image, so i would say it alone could be useful in this specific case. There was of course other avenues to explore, i could have experimented more with errors and set size. Anecdotally the size of the data set did not seem to matter too much for the image data, but quite a bit for the analytic frankefunction. Adding errors seemed to just muddy the plots a little more.