FYS-STK 3155/4155 - Project 1: Regression analysis

Federico Nardi and Francesco Slongo
University of Oslo, Department of Physics
(Dated: October 15, 2018)

In this paper we are going to present the most common and used methods for creating and evaluating a model: the linear regression, the Ridge regression and the Lasso regression together the bootstrap resampling method. We applied this methods to an exponential function with some random noise and then to some real data, corresponding to the terrain of Møsvatn Austfjell. We suggest that the best model for the exponential function is a high order polynomial, while the polynomial model up to fifth order could not fit properly the terrain.

- GitHub folder with code: `https://github.com/FedericoNardi/MachineLearning`

I. INTRODUCTION

It is often necessary in different fields of science and not only to analyze the connection between different factors and the role they play in the context they belong to. In this way it is possible to create a model of the studied problem. One of the most used methods in statistics to test and analyze models is the linear regression, due to its generality and effectiveness. In this paper we show the theory of linear regression together with its most used variations, the Ridge and the Lasso method. In addition, we implement in the analysis bootstrap resampling, a technique that allows us to compare different models and improve the accuracy without requiring more data. To analyze the models we use statistical estimators: mean squared error MSE, the R^2 score, the variance and the bias. In section II we present the theory underlying linear regression, resampling techniques and analysis of models. In the following section (section III), we analyze a randomly generated data set by adding a noise component to the Franke function to test the effectiveness of the methods and the correctness of the developed code. Subsequently we apply the developed algorithms to some real data: we try to find a model to fit the elevation of a patch of terrain (We chose a the terrain of Møsvatn Austfjell). We use the data downloaded from [4]. All the plots are in section V. All the results, the material and the code can be found in the GitHub folder at `https://github.com/FedericoNardi/MachineLearning`. For the theoretical explanation we referred mainly to the course's lecture notes [1].

II. METHODS AND ALGORITHMS

A. Linear regression

Our aim is to find a model for the trend of n data $\hat{y}$ versus n a set of response data $\hat{x}$. We are looking for a linear model with the form $\hat{\tilde{y}} = \hat{\beta}\hat{x}$. Note specify that the data $\hat{x}$ do not necessarily have to be the simple data, but also function of them (for example some polynomials or some sinusoidal functions of x). In our analysis we use as $\hat{x}$ some polynomials of different degree p, therefore our $\hat{x}$ is an $n \times p$ matrix.

$$\hat{y} = \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_{n-1} \end{bmatrix} \qquad \hat{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \dots \\ \beta_{n-1} \end{bmatrix} \tag{1}$$

$$\hat{x} = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^p \\ 1 & x_1 & x_1^2 & \dots & x_1^p \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^p \end{bmatrix} \tag{2}$$

To study our regression we define the cost function $Q(\hat{\beta})$. This function should give us a measurement for the deviation of our prediction from the actual data and can be defined in different ways. The most common cost function is the least squares:

$$Q(\hat{\beta}) = \sum_{i=0}^{n-1} (y_i - \tilde{y}_i)^2 \tag{3}$$

The best model will be the one that minimizing Q. Therefore to obtain β, we derive Q for the parameters β and we obtain an equation for $\hat{\beta}$.

$$\hat{\beta} = \hat{H}^{-1}\, \hat{x}^T\, \hat{y} \qquad \hat{H} = \hat{x}^T \hat{x} \tag{4}$$

It is possible to show that the covariance matrix of the parameters beta is:

$$\sigma^2[\hat{\beta}] = \hat{H}^{-1}\, \sigma^2 \tag{5}$$

where:

$$\sigma^2 = \frac{1}{n-p-1}\sum_{i=0}^{n-1}(y_i - \tilde{y_i})^2 \tag{6}$$

One problem with the standard linear regression is overfitting: having huge data sets (which may results in linearly dependent columns in $\hat{x}$) and a strong noise in the data may complicate the fitting. This can lead to problems with the inversion of the matrix H, that may be singular. A simple way to overcome this problem is performing a Singular Value Decomposition (SVD) on the data matrix: any matrix A can in fact be expressed as

$$A = U\,D\,V^t$$

where $U \in \mathbb{R}^{m\times n}$ and $V \in \mathbb{R}^{n\times n}$ are two orthogonal matrices ($U^t = U^{-1}$) and $D \in \mathbb{R}^{m\times n}$ has values only on the diagonal. We can therefore express $\hat{x}$ in terms of its SVD and eliminate therefore the singularity. This procedure has been implemented in our code. To avoid this problem we define a different cost function $Q^{\mathrm{R}}(\hat{\beta})$ using an arbitrary parameter λ:

$$Q^{\mathrm{R}}(\hat{\beta}) = \sum_{i=0}^{n-1}(y_i - \tilde{y_i})^2 + \lambda\left\|\hat{\beta}\right\|_2^2 \tag{7}$$

where:

$$\left\|\hat{\beta}\right\|_2 = \sqrt{\sum_{i=0}^{n-1}\beta_i^2} \tag{8}$$

Using the previous operations, we obtain a new equation for $\hat{\beta}$:

$$\hat{\beta} = \left(\hat{H} + \lambda\hat{1}\right)^{-1}\hat{x}^T\,\hat{y} \tag{9}$$

and the covariance matrix becomes:

$$\sigma^2[\hat{\beta}] = \left(\hat{H} + \lambda\hat{1}\right)^{-1}\hat{H}\left(\left(\hat{H} + \lambda\hat{1}\right)^{-1}\right)^T\sigma^2 \tag{10}$$

This regression is known as Ridge regression. The value λ is an arbitrary parameter that punishes models with high parameters β and for this reason it is called penalty parameter. It is possible to show that the Ridge regression shrinks more the columns of $\hat{x}$ with the lower variance (a proof of this can be found in [2]). The advantage of the Ridge regression resides in the fact that the model obtained has a lower variance, in spite of a high bias on the parameter λ.

Obviously we can define even different cost functions. The so-called Lasso regression is based on the following $Q^{\mathrm{L}}(\hat{\beta})$:

$$Q^{\mathrm{L}}(\hat{\beta}) = \sum_{i=0}^{n-1}(y_i - \tilde{y_i})^2 + \lambda\left\|\hat{\beta}\right\|_1^2 \tag{11}$$

where:

$$\left\|\hat{\beta}\right\|_1 = \sum_{i=0}^{n-1}|\beta_i| \tag{12}$$

The problem in this case is that $Q^{\mathrm{L}}(\hat{\beta})$ is not a linear function in the parameters $\hat{\beta}$, so the minimum of this function cannot be found analytically. In order to obtain the parameters $\hat{\beta}$, it is necessary to minimize $Q^{\mathrm{L}}(\hat{\beta})$ numerically.

It can be shown that the Lasso regression is most likely to set to 0 some of the parameters β_i, based on the value of λ. Moreover in this way one can find out what parameters β (and therefore what part of the data $\hat{x}$) are not related to $\hat{y}$). If all the parameters β are not zero and they are not sparse, it is possible to calculate their variance (for a proof of the equation, see [3]):

$$\sigma^2[\hat{\beta}] = \left[\hat{H} + \Psi(\hat{\beta})\right]^{-1}\hat{H}\left[\hat{H} + \Psi(\hat{\beta})\right]^{-1}\sigma^2 \tag{13}$$

where (if $|\beta_i| \neq 0$):

$$\Psi(\hat{\beta})_{ii} = \frac{1}{|\beta_i|} \qquad \Psi(\hat{\beta})_{ij} = 0 \tag{14}$$

B. Evaluation of the model

To compare different models and their goodness, we have to introduce some new cost functions (we cannot use the previous ones because the model we obtained are already the ones that minimized them). Therefore we define the Mean Squared Error MSE and the Coefficient of Determination R^2, that indicate how much our model $\hat{\tilde{y}}$ describes the data $\hat{y}$:

$$MSE = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y_i})^2 \tag{15}$$

$$R^2 = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \tilde{y_i})^2}{\sum_{i=0}^{n-1}(y_i - E[y])^2} \tag{16}$$

where:

$$E[y] = \frac{1}{n}\sum_{i=0}^{n-1} y_i \tag{17}$$

A good model is one that has a low value of MSE and a value of R^2 close to 1.

We want to evaluate two more values about the model: the variance and the bias. The variance gives an idea of how much our model is specific to our set of data and not general to the case we are studying. The bias, instead, is the error caused by the

simplifying assumptions built into the method. The variance is:

$$Var[\tilde{y}] = E[\tilde{y}^2] - E[\tilde{y}]^2 \tag{18}$$

while the bias is:

$$Bias[\tilde{y}] = E[(y - E[\tilde{y}]) \tag{19}$$

The variance and the bias of the model are related by the bias-variance trade off. If we define the total error of the model as:

$$Err(y, \tilde{y}) = E[(y - \tilde{y})^2] \tag{20}$$

it is possible to derive the following equation:

$$Err(y, \tilde{y}) = Var[\tilde{y}] + Bias^2[\tilde{y}] + \sigma^2 \tag{21}$$

where σ^2 in this case is the variance of the noise.

C. Resampling methods

Sometimes is not possible to collect more data and therefore test the model versus new data. A way to deal with this problem is resampling. The basic idea is to split all the data in two groups, the training group and the test group. First we build the model on the train data and we obtain the parameters β, then we analyze the model against the test data. From the analysis against test data, we then obtain one value of MSE and R^2 for every model, so we can just choose the better and we know its error, because we calculate the total error, the variance and the bias. However doing this method only one time does not let us to conclude that the model we obtained is the best we could have: we are biased by how we split the initial data in test and train data. We repeat this procedure many times and we split the data every time in different ways. Anyway it is important to notice is better to have a more number of training data instead of testing data, because this will ensure that the model is better. Then the parameters of the model are calculated by taking the average of all the values obtained.

A way to split the data is to divide them in k groups with the same number of data and doing the resampling k times. Every time one of the groups is chosen as testing data while the other are used as training data. This method is called k-fold.

Another common resampling method is bootstrap. It consist in creating a group of data by picking randomly for an arbitrary number of times some data y and x (the data can be even repeated). Then the train and the test is done on that group. This procedure is repeated an arbitrary number of times and the parameters calculated averaging on the parameters of each set.

Another advantage of these methods is that they allow us to calculate the error of the model and of the parameters β without having to calculate directly the covariance, that is computationally expensive.

III. RESULTS

A. Franke function

In the first part of the project we test the methods against a function with some random noise. The used function is the Franke function:

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

We then define the variable z as $z = f(x,y) + \epsilon$, where ϵ is a random noise component. We are looking for a model that is function of x and y. We then make a matrix $\hat{x}$ with polynomials of different degree of x and y (in our code we used the `fit_transform` function from *scikit-learn* package for Python). For example for the model of the second order polynomial the matrix $\hat{x}$ is:

$$\hat{x} = \begin{bmatrix} 1 & x_0 & y_0 & x_0^2 & x_0\,y_0 & x_0^2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & x_{n-1} & y_{n-1} & x_{n-1}^2 & x_{n-1}\,y_{n-1} & x_{n-1}^2 \end{bmatrix}$$

We can hence apply the methods shown above. The results are presented in the graphs in section V (figure from 2 to 19).

If we look at the graph a in figure 2, we can see that the MSE and the R^2 decrease for higher orders, but that is obvious: a complex model can reproduce in better way the testing data with respect to a simpler one, but will not necessarily be general enough. In fact if we check the graph (a) in figure 3 we see that the trend is reversed: the models with too many degree of freedom are worse compared to the polynomial of first and second order. The models with high order polynomial fit very well the training data but then they become too specific. Looking at the graphs of the parameters, we can see that they are quite different (except for the first order polynomial): we did not expect this. This could suggest how much the analysis is biased if we just fit the model on some data but we do not think this is the only reason. If we look for the graphs of the regressions, we see

that they have the same trend, while the only one that is different is the one in figure 3. We suppose that we repeated the bootstrap resampling technique not enough times, so that the results are biased by how the data were picked: by selecting each time a particular path in the (x, y) plane we have a much wider range for our parameters, and this explains the higher MSE.
We therefore conclude that the best model is a fifth order polynomial. This would make sense since the Taylor expansion of an exponential is a sum of polynomials of any degree, so the more polynomials we sum, the more we can approach the Taylor expansion of the Franke function.

In the graphs (a) of figures 4, 5, 6 and 7 we can see how the different parameters of λ give different errors in the Ridge regression. The more complex models fit better the data and in this case the resampling confirms this. Anyway we can see that linear regression and the Ridge regression give the same results for polynomial of order less or equal than 3. This makes sense since the penalty parameter is multiplied for a sum of few parameters. For higher order polynomials we sum many more parameters, so the Ridge shrinks all of them. This is evident for the fifth order polynomial. We may even notice that for $\lambda = 0.01$ the Ridge regression is the only one that differs from the linear regression: probably the penalty parameter is high enough to shrink the parameters, but its effect kicks in only when we use high order polynomials. With polynomial of order higher than five, we may be able to see that the parameters are shrinked even for smaller values of λ.
Moreover we can compare the results for different noise factors. For the values 0.001 and 0.01 (figures 4 and 5) of noise amplitude, the regression does not change. For the values 0.1 and 0.5 (figures 6 and 7) the error increases and therefore we have higher values of MSE and lower values of R^2. This is reflected in the bootstrap: the regression with and without resampling are equal for amplitude of the noise of 0.001 and 0.01 (figures 4, 8 and 5,9), while it differs a bit for the value 0.1 (figures 6 and 10) and way more for the value 0.5 (figures 6 and 10).

In the plots (a) of figures 12, 13, 14 and 15 we can see how the different parameters of λ give different accuracy in the Lasso regression. As in the linear regression and the Ridge regression, a higher complexity leads to a lower error (except for $lambda = 0.01$). We note that the Lasso regression gives a different result for values of λ lower then 0.0001. This may suggest that Lasso regression is more sensitive to the penalty parameter than the Ridge regression. For $\lambda = 0.01$ the error on the model is rather high: this is due the fact that the Lasso can set to zero some of the parameters and this is more likely to happen the higher the value of λ and the complexity becomes. In fact if we look at higher order polynomials all the parameters are zero or close to this value: this tells us that for high values of λ the regression is not reliable. Anyway we may use the regression with $\lambda = 0.0001$ to find what parameters are not necessary for the fit and therefore reduce the number of degrees of freedom.
We can also look at the influence of the noise. For low values of noise amplitude the fit does not change (check figures 12 and 12), but for higher values it does: the shrinkage on parameters β is higher and in general the MSE increases, while the R^2 decreases. However we did not expect that the MSE would be higher with a smaller noise for low order polynomials (figure 14 and 15) and we are not able to explain this.
If we compare the Lasso regression with and without the bootstrap technique, we see that the bootstrap method not always gives the same results as the normal Lasso regression: for high order polynomials and high noise the parameters β are even more shrinked: the bootstrap algorithm could create some samples and the parameters of that samples could be put to zero by the Lasso algorithm. Averaging on the different results may then lead to a further shrinkage.

We now analyze the bias and the variance of our models (table I).

p	Err	$Bias^2$	Var
Linear regression			
1	0.0989	0.09962	2.125×10^6
2	0.0980	0.09962	9.319×10^6
3	0.1000	0.09962	1.692×10^5
4	0.0997	0.09962	2.775×10^5
5	0.1003	0.09962	4.272×10^5
Ridge regression			
1	0.0230	0.0886	0.0656
2	0.0163	0.0886	0.0721
3	0.0077	0.0886	0.0786
4	0.0071	0.0886	0.0796
5	0.0059	0.0886	0.0799
Lasso regression			
1	0.0288	0.0887	0.0334
2	0.0275	0.0887	0.0364
3	0.0277	0.0887	0.0362
4	0.0275	0.0887	0.0366
5	0.0276	0.0887	0.0369

Table I: Values of total error $Eerr$, Var and $Bias^2$ of the polynomial models of degree p, for linear regression , Ridge regression and Lasso regression (with $\lambda = 0.01$) with bootstrap. The data for the Ridge and Lasso regressions have a noise factor of 0.1 and $\lambda = 0.01$

We can see from table I that the bias is predominant in our errors. As we expected, the variance increases with the degree of the polynomial p. Anyway we see that bias is constant: we did not expect this, because the bias would decrease with the increase of the complexity of the model. We suspect that this is due an error in the code, but we could not find that.

B. Real data

We choose to analyze one of the terrains in the Github folder of the course FYS4155 corresponding to Møsvatn Ausfjell. The terrain is shown in figure 1 below:

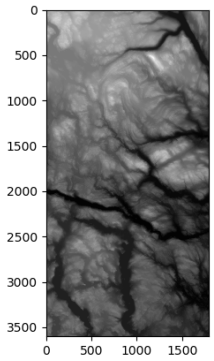


Figure 1: Picture of the terrain analyzed. The height is represented by the gray color: a color more similar to black means a lower height, while a color more similar to white means a higher height.

After having loaded all the data in our Python script, we perform the same analysis as on the Franke function with a fifth order polynomial. We thought that if a high order polynomial would give us low values of MSE, it would be better to analyze even the lower orders; otherwise it is not use full because less complex model will likely fit worse the data. To have a better computational efficiency we have divided all the region in different patches according to the code suggested in the online discussions of the course, for a total of 5 patches. The results of the fits of a polynomial model are in table II, while the plots are in section V (figures from 20 to 24).
The confidence intervals for the parameters β can be found in the GitHub folder (parameters.txt) (the error calculated corresponds to one STD, i.e. 68% of confidence).

Patches	MSE	R^2
Linear regression		
1	276	0.990
2	1819	0.924
3	164	0.991
4	259	0.926
5	210	0.953
Ridge $\lambda = 0.01$		
1	529	0.981
2	2736	0.886
3	316	0.983
4	293	0.916
5	241	0.946
Ridge $\lambda = 0.0001$		
1	282	0.990
2	1822	0.924
3	168	0.991
4	260	0.926
5	210	0.953
Lasso $\lambda = 0.01$		
1	795	0.972
2	3874	0.839
3	537	0.971
4	357	0.898
5	301	0.932
Lasso $\lambda = 0.0001$		
1	771	0.972409
2	3794	0.842329
3	534	0.971216
4	352	0.899709
5	279	0.937526

Table II: Values of MSE and R^2 of the models for the different patches.

From table II we see that in general the polynomial model does not fit correctly the data: in general the MSE is higher than 100, but for some patches we have good values of R^2. Anyway we think that such a relatively low degree polynomial fit is not suited to the data, due to the irregularities on the features of the terrain; in fact the patches has very different values of MSE. Moreover for all the regressions the patch number 2 was the one with the worst values of MSE. We see that the regression does no fit equally well every patch: the patch that is better fitted by the linear regression is the number 3, while for the Ridge regression with $\lambda = 0.01$ and the Lasso regressions is the patch number 5. Probably the patch number 3 requires a model with rather high values for the weighs β, which are heavily punished by the high values of λ. Anyway we think that dividing in smaller patches the terrain may be a good idea, since fitting all the region would have lead to an even higher MSE.

We suppose that low degree polynomial model is not suitable to reproduce all the features the terrain, because there are different creeks: the polynomial model is a smooth function and in order to represent such great changes in the data the model it should be a rather high order polynomial. Moreover, a polynomial blows up at ∞ and at the same time we need a sum of different polynomials to try to fit the changes in the terrain. This leads to a model that cannot be as dishomogeneous as our terrain. We suggest that probably a fit to sinusoidal would be even more effective.

Another solution could be to analyze even smaller regions, since mathematics teaches us that every smooth function can be seen locally as a flat function. In this way we could be able to fit the terrain at least to a first or second order polynomial. In the aftermath we think that it would have been better better to choose a more flat terrain to be able to apply a polynomial model.

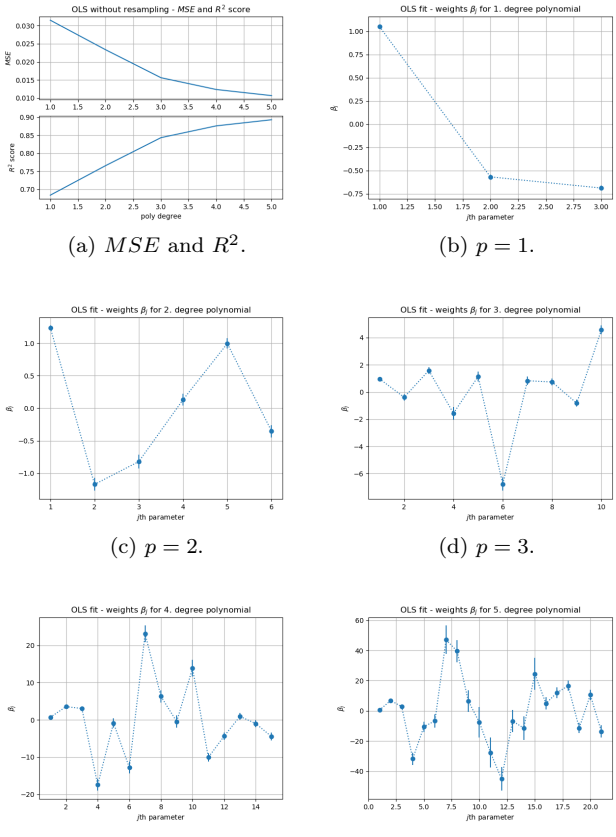
IV. CONCLUSIONS

The test on the Franke function has given us the expected results: an exponential is better fitted with an high order polynomial model, because its Taylor expansion is a sum of polynomials. Anyway we encountered two unexpected results: the bootstrap resampling applied to the linear regression gives a completely different result from the rest of the analysis (figure number 3, (a) and in the Lasso regression a lower noise leads to a bigger MSE for low order polynomials (figures 14 and 15, (a). While we are not able to provide an explanation for the second problem, we suppose that the first one may be due to a low number of repetition of bootstrap algorithm and a statistical error, and with a bigger data set the situation slightly improves: if we look at the error bars of the parameters we can see that they are bigger compared to the ones of the regressions without bootstrap. Anyway we conclude that using bootstrap shrinks the parameters for all the regressions, but increases the error on the parameters. The Ridge regression and the Lasso regression for big enough values of λ and high order polynomials shrink the parameters: the Lasso regression is more sensitive to the value of λ and the higher the noise, the higher the shrinkage.

We were not able to fit a fifth order polynomial to our terrain. The main cause are the creeks which make the terrain quite irregular. We think that a solution to this problem would be to fit smaller regions or to do a sinusoidal fit instead of a polynomial one.

We can say that we applied the methods correctly but we did not use the correct model for the real data, so our analysis can rely only on the study of the exponential function. In this case, we can prove the effectiveness of this methods: they are a strong tool to create models, because they are general and offer a large amount of tools for a deep analysis.

V. APPENDIX

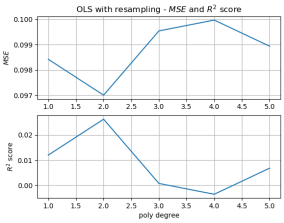


(a) MSE and R^2. (b) $p = 1$.

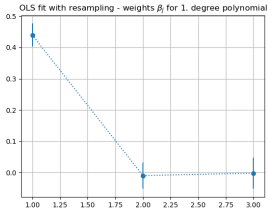
(c) $p = 2$. (d) $p = 3$.

(e) $p = 4$. (f) $p = 5$.

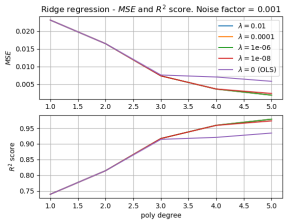
Figure 2: Parameters β of the linear regression with polynomials of different order p and error of the models on the Franke function.



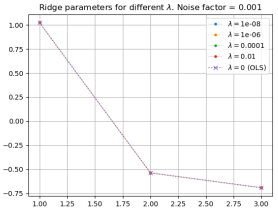
(a) MSE and R^2.



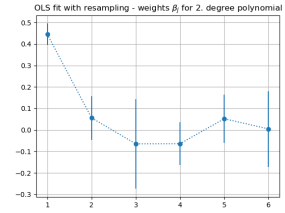
(b) $p = 1$.



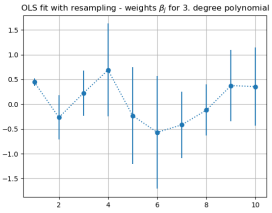
(a) MSE and R^2.



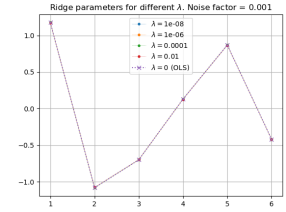
(b) $p = 1$.



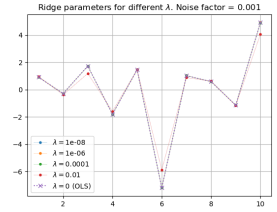
(c) $p = 2$.



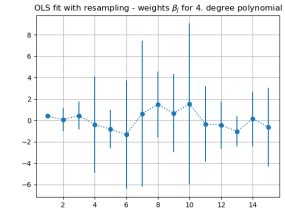
(d) $p = 3$.



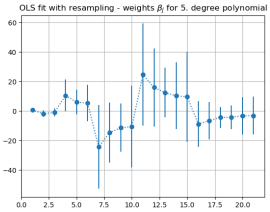
(c) $p = 2$.



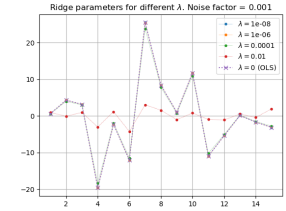
(d) $p = 3$.



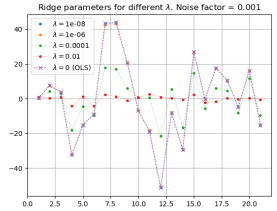
(e) $p = 4$.



(f) $p = 5$.



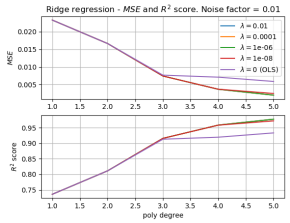
(e) $p = 4$.



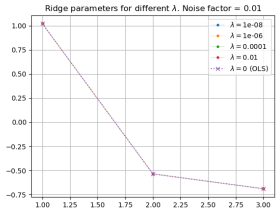
(f) $p = 5$.

Figure 3: Parameters β of the linear regression with polynomials of different order p and error of the models using the bootstrap resampling technique on the Franke function.

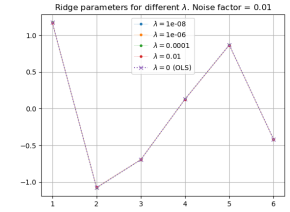
Figure 4: Parameters β for the Ridge regression with polynomials of different order p and error of the models on the Franke function. The noise has an amplitude of 0.001.



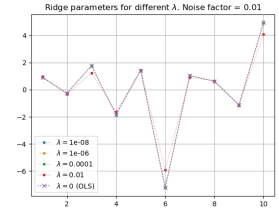
(a) MSE and R^2.



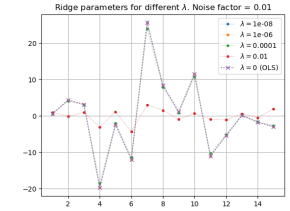
(b) $p = 1$.



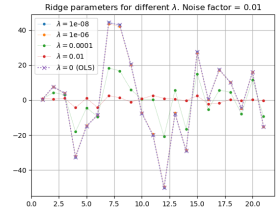
(c) $p = 2$.



(d) $p = 3$.

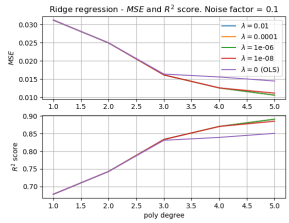


(e) $p = 4$.

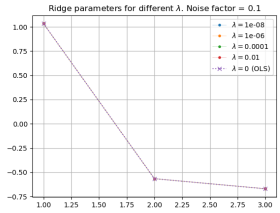


(f) $p = 5$.

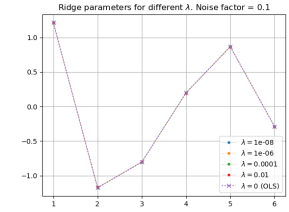
Figure 5: Parameters β for the Ridge regression with polynomials of different order p and error of the models on the Franke function. The noise has an amplitude of 0.01.



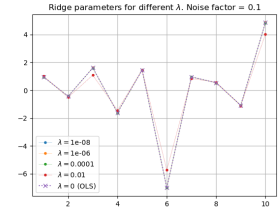
(a) MSE and R^2.



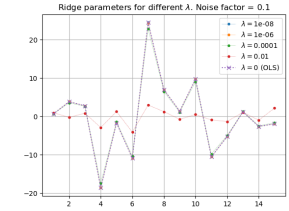
(b) $p = 1$.



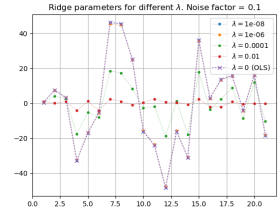
(c) $p = 2$.



(d) $p = 3$.

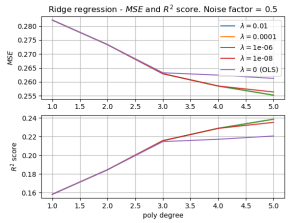


(e) $p = 4$.

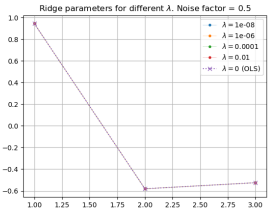


(f) $p = 5$.

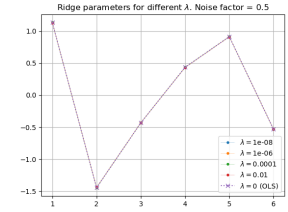
Figure 6: Parameters β for the Ridge regression with polynomials of different order p and error of the models on the Franke function. The noise has an amplitude of 0.1.



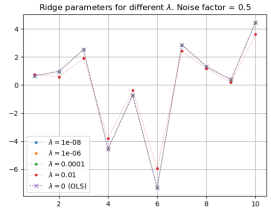
(a) MSE and R^2.



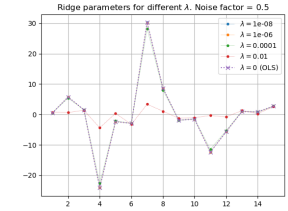
(b) $p = 1$.



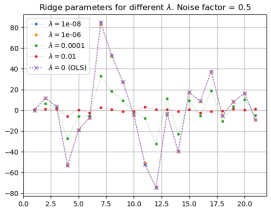
(c) $p = 2$.



(d) $p = 3$.

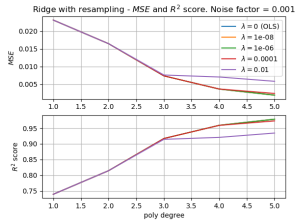


(e) $p = 4$.

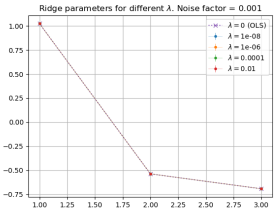


(f) $p = 5$.

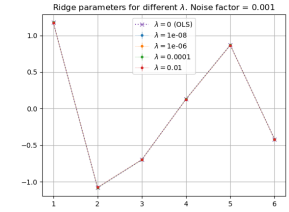
Figure 7: Parameters β for the Ridge regression with polynomials of different order p and error of the models on the Franke function. The noise has an amplitude of 0.5.



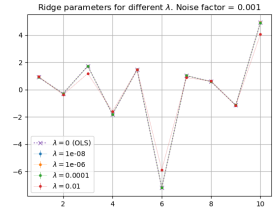
(a) MSE and R^2.



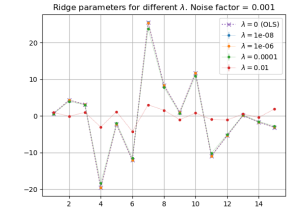
(b) $p = 1$.



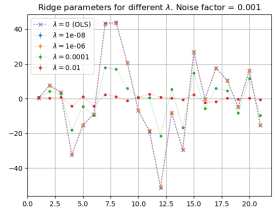
(c) $p = 2$.



(d) $p = 3$.

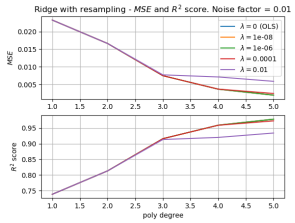


(e) $p = 4$.

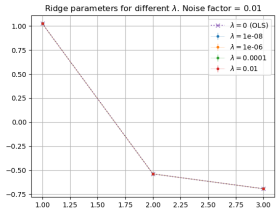


(f) $p = 5$.

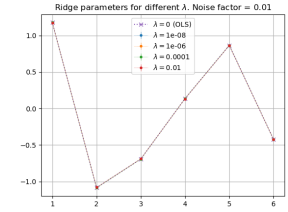
Figure 8: Parameters β for the Ridge regression with polynomials of different order p and error of the models using the bootstrap resampling technique on the Franke function. The noise has an amplitude of 0.001.



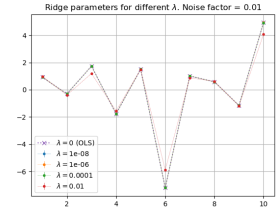
(a) MSE and R^2.



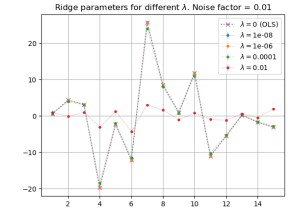
(b) $p = 1$.



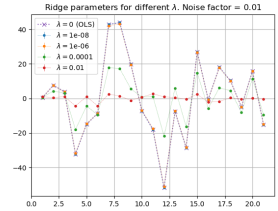
(c) $p = 2$.



(d) $p = 3$.

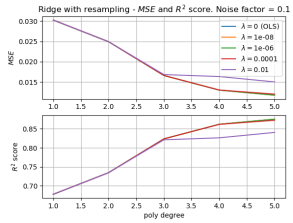


(e) $p = 4$.

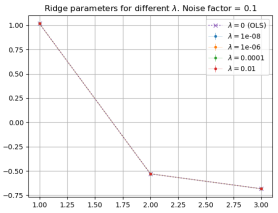


(f) $p = 5$.

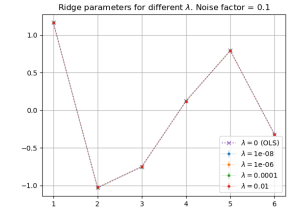
Figure 9: Parameters β for the Ridge regression with polynomials of different order p and error of the models using the bootstrap resampling technique on the Franke function. The noise has an amplitude of 0.01.



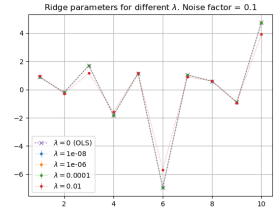
(a) MSE and R^2.



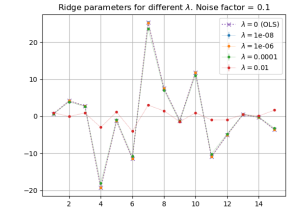
(b) $p = 1$.



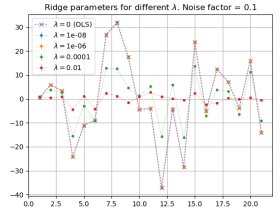
(c) $p = 2$.



(d) $p = 3$.

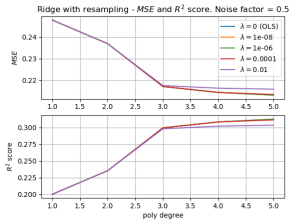


(e) $p = 4$.

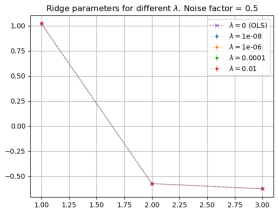


(f) $p = 5$.

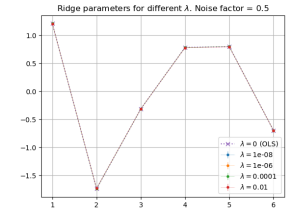
Figure 10: Parameters β for the Ridge regression with polynomials of different order p and error of the models using the bootstrap resampling technique on the Franke function. The noise has an amplitude of 0.1.



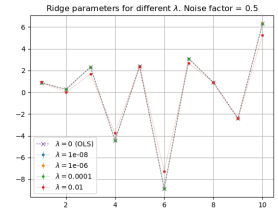
(a) MSE and R^2.



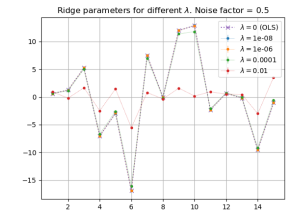
(b) $p = 1$.



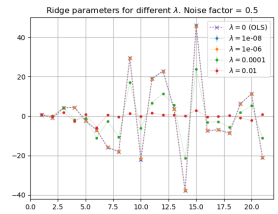
(c) $p = 2$.



(d) $p = 3$.

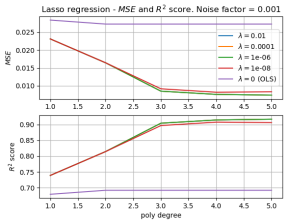


(e) $p = 4$.

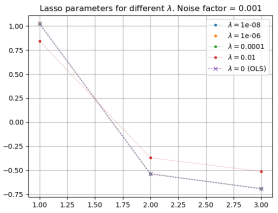


(f) $p = 5$.

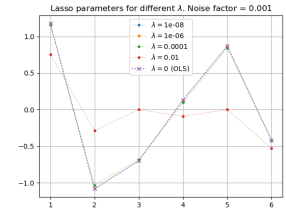
Figure 11: Parameters β for the Ridge regression with polynomials of different order p and error of the models using the bootstrap resampling technique on the Franke function. The noise has an amplitude of 0.5.



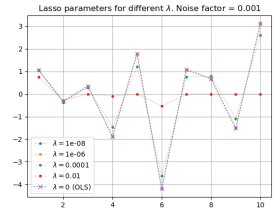
(a) MSE and R^2.



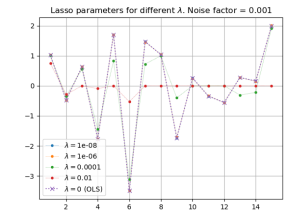
(b) $p = 1$.



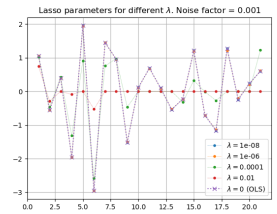
(c) $p = 2$.



(d) $p = 3$.

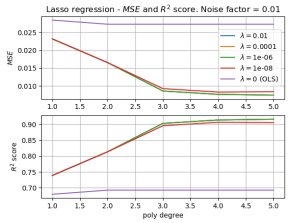


(e) $p = 4$.

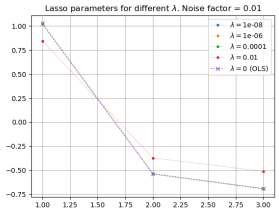


(f) $p = 5$.

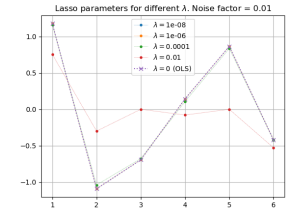
Figure 12: Parameters β for the Lasso regression with polynomials of different order p and error of the models on the Franke function. The noise has an amplitude of 0.001.



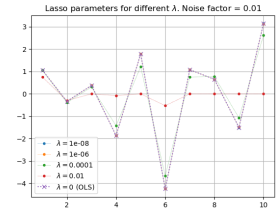
(a) MSE and R^2.



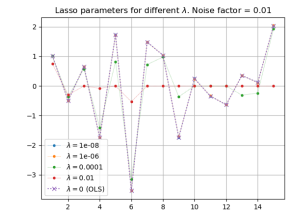
(b) $p = 1$.



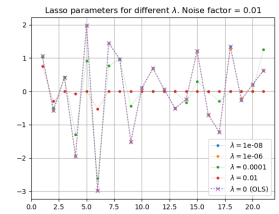
(c) $p = 2$.



(d) $p = 3$.

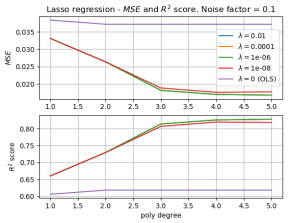


(e) $p = 4$.

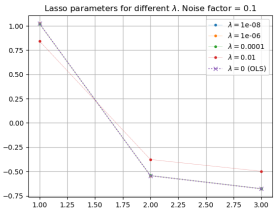


(f) $p = 5$.

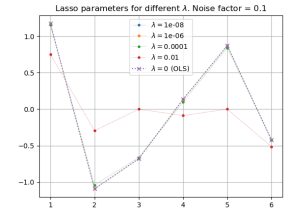
Figure 13: Parameters β for the Lasso regression with polynomials of different order p and error of the models on the Franke function. The noise has an amplitude of 0.01.



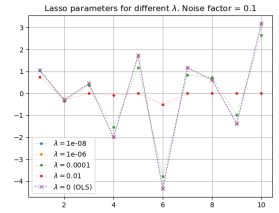
(a) MSE and R^2.



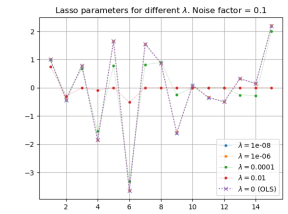
(b) $p = 1$.



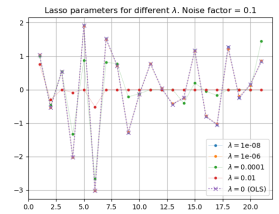
(c) $p = 2$.



(d) $p = 3$.

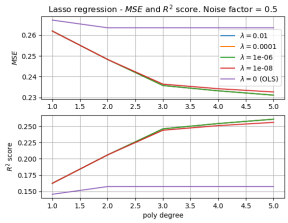


(e) $p = 4$.

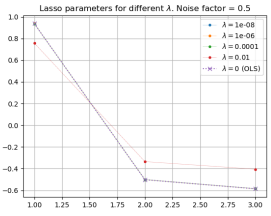


(f) $p = 5$.

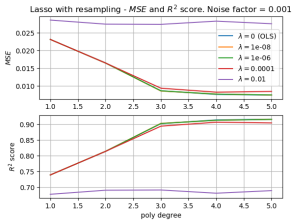
Figure 14: Parameters β for the Lasso regression with polynomials of different order p and error of the models on the Franke function. The noise has an amplitude of 0.1.



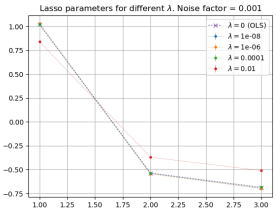
(a) MSE and R^2.



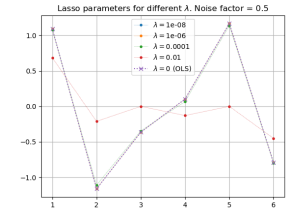
(b) $p = 1$.



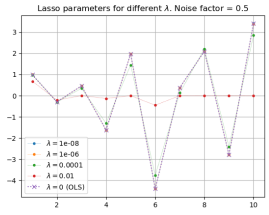
(a) MSE and R^2.



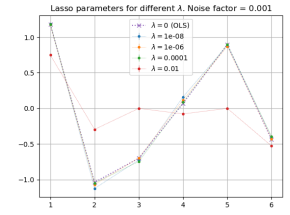
(b) $p = 1$.



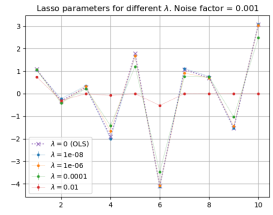
(c) $p = 2$.



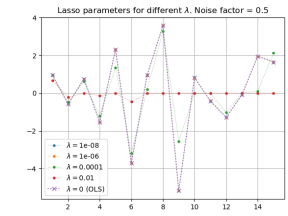
(d) $p = 3$.



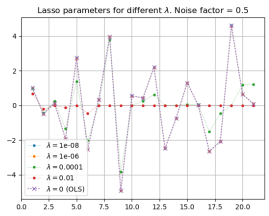
(c) $p = 2$.



(d) $p = 3$.

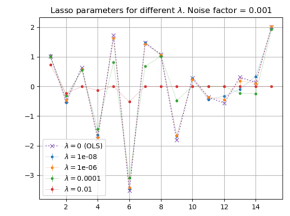


(e) $p = 4$.

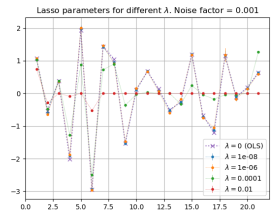


(f) $p = 5$.

Figure 15: Parameters β for the Lasso regression with polynomials of different order p and error of the models on the Franke function. The noise has an amplitude of 0.5.

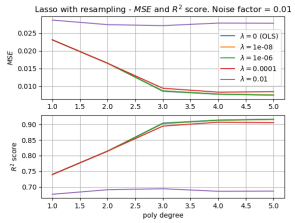


(e) $p = 4$.

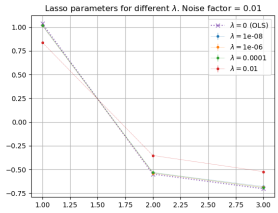


(f) $p = 5$.

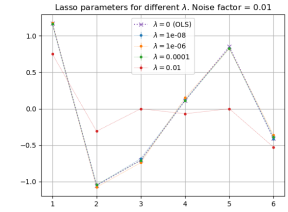
Figure 16: Parameters β for the Lasso regression with polynomials of different order p and error of the models using the bootstrap resampling technique on the Franke function. The noise has an amplitude of 0.001.



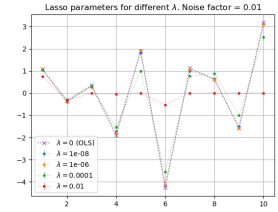
(a) MSE and R^2.



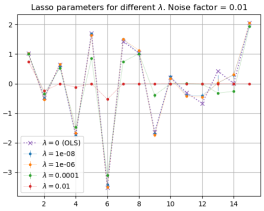
(b) $p = 1$.



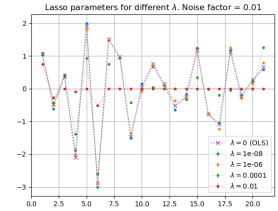
(c) $p = 2$.



(d) $p = 3$.

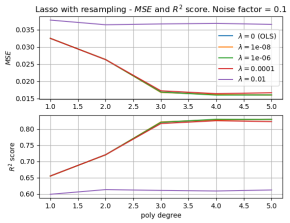


(e) $p = 4$.

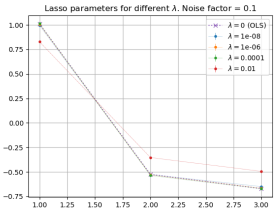


(f) $p = 5$.

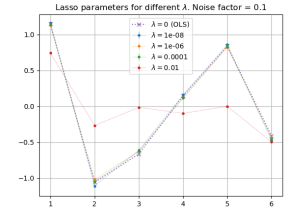
Figure 17: Parameters β for the Lasso regression with polynomials of different order p and error of the models using the bootstrap resampling technique on the Franke function. The noise has an amplitude of 0.01.



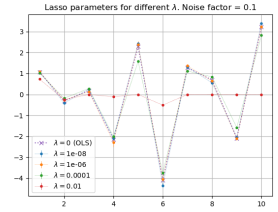
(a) MSE and R^2.



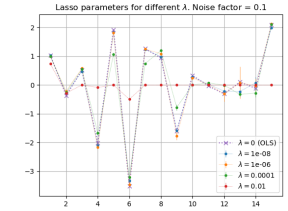
(b) $p = 1$.



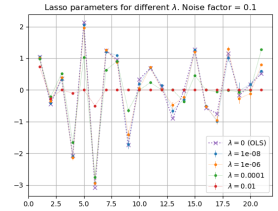
(c) $p = 2$.



(d) $p = 3$.

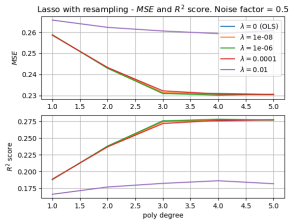


(e) $p = 4$.

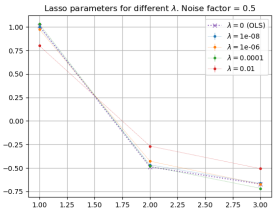


(f) $p = 5$.

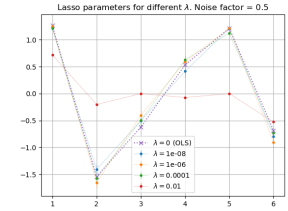
Figure 18: Parameters β for the Lasso regression with polynomials of different order p and error of the models using the bootstrap resampling technique on the Franke function. The noise has an amplitude of 0.1.



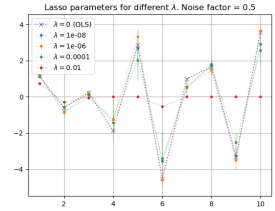
(a) MSE and R^2.



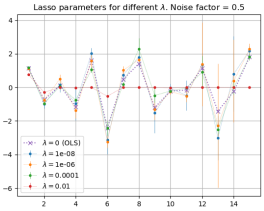
(b) $p = 1$.



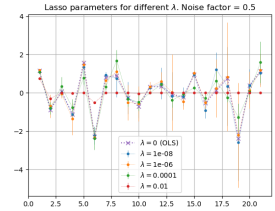
(c) $p = 2$.



(d) $p = 3$.

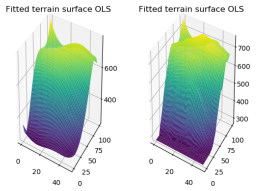


(e) $p = 4$.

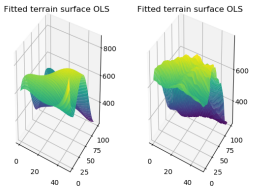


(f) $p = 5$.

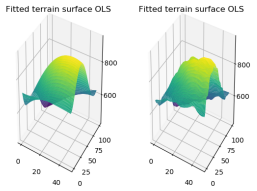
Figure 19: Parameters β for the Lasso regression with polynomials of different order p and error of the models using the bootstrap resampling technique on the Franke function. The noise has an amplitude of 0.5.



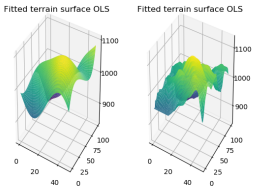
(a) Patch 1.



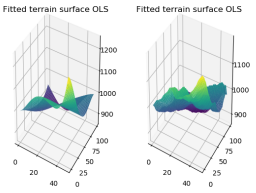
(b) Patch 2.



(c) Patch 3.

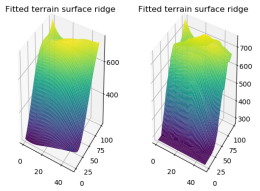


(d) Patch 4.

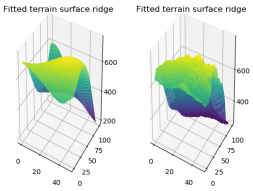


(e) Patch 5.

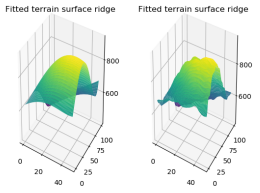
Figure 20: Plots of the patches of terrain using the linear regression.



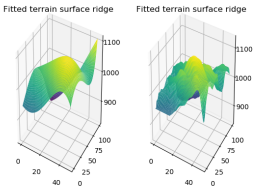
(a) Patch 1.



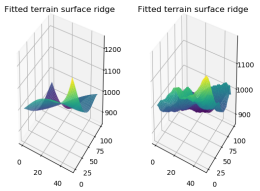
(b) Patch 2.



(c) Patch 3.

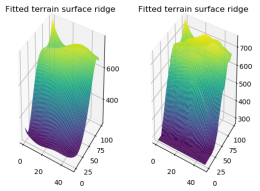


(d) Patch 4.

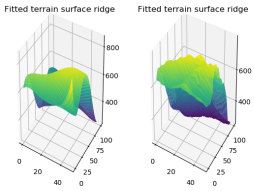


(e) Patch 5.

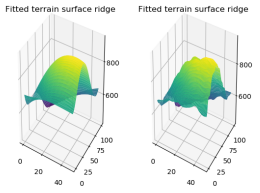
Figure 21: Plots of the patches of terrain using the Ridge regression with $\lambda = 0.01$.



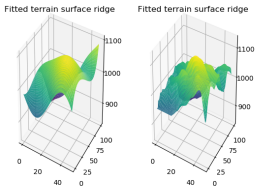
(a) Patch 1.



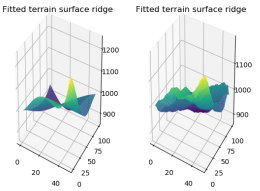
(b) Patch 2.



(c) Patch 3.

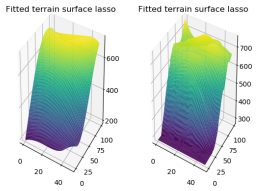


(d) Patch 4.

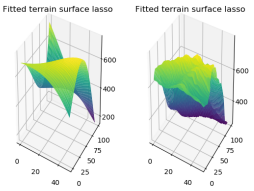


(e) Patch 5.

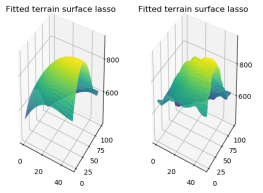
Figure 22: Plots of the patches of terrain using the Ridge regression with $\lambda = 0.0001$.



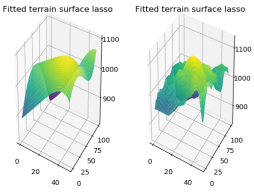
(a) Patch 1.



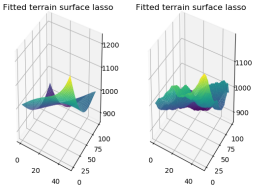
(b) Patch 2.



(c) Patch 3.

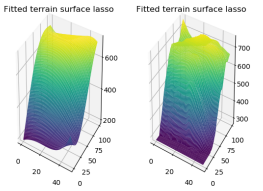


(d) Patch 4.

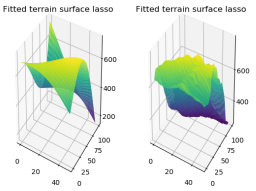


(e) Patch 5.

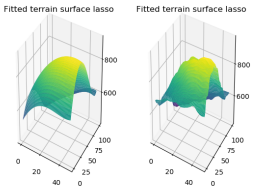
Figure 23: Plots of the patches of terrain using the Lasso regression with $\lambda = 0.01$.



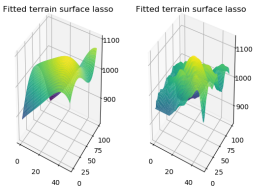
(a) Patch 1.



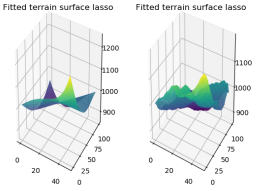
(b) Patch 2.



(c) Patch 3.



(d) Patch 4.



(e) Patch 5.

Figure 24: Plots of the patches of terrain using the Lasso regression with $\lambda = 0.0001$.

[1] M. H. Jensen, *Data Analysis and Machine Learning - Lecture notes Fall 2018*, University of Oslo - Department of Physics, 2018.

[2] Trevor Hastie, Robert Tibshirani, Jerome Friedman *The Elements of Statistical Learning - Data Mining, Inference, and Prediction*, Springer.

[3] Wessel N. van Wieringen *Lecture notes on ridge regression*

[4] *Website for the terrain data:* `https://earthexplorer.usgs.gov/`