University of Caen

Machine Learning

Regression analysis

Students:

Andrea Espinosa

Vasily Sorokin

Vimal Vijayan

Machine Learning Project for Erasmus Mundus International NuPhys Master Programme

Abstract

In this work we present an assesment of the predictive ability of some of the most popular regression methods. Ordinary Linear Regression, Ridge and Lasso regression are tested for a two dimensional function. Influence of the lambda parameter is also studied in the case of the Ridge and Lasso methods. Quality of the fitting is determined considering mean squared error and R2 values. It is shown how accuracy depends on the order of the polynomial fitting that is performed and on the initial data noise. In all the cases quality of estimators is evaluated combining the different regression methods with 5-fold cross validation.

In the second part the real data was introduced and was assayed for polynomial fitting with ordinary least squares regression. Likewise, different regression method (such as Lasso and Ridge) and resampling techniques were applied for training. R2 vs. regularization parameter dependence was also studied.

Contents

1 Abstract

2 Introduction and Formalism

Information can be obtained from any given set of data by relating the data points in infinitely different ways. For instance, a given set of data points in a 2D space can be connected and averaged in infinte number of ways of straight lines, curves of different order of polynomials, or with more complex functions. This process of estimating a dependent variable that gives an average dependency relation to a given set of data is known as regression. Once we obtain the model using the given data, the fundamental objective is to predict the future outcomes with our model and analyze how well it can predict the real ones. Depending on the type of regression used, it can be a simple linear regression or a complicated polynomial regression (spline), etc.
It is normally considered the simple approach of linear regression where the model function is parametrized by the polynomial of degree $n-1$ for n given points. In specific, we also consider three different types of Linear regression methods, Ordinary Least square, Ridge regression and the Lasso regression.

The basic idea is that for a given a set of data that consists x_i i = 1,...n input data and the corresponding output data y_i, we are trying to make a model $\tilde{y}_i$ that follows the condition,

$$y_i = \tilde{y}_i + \epsilon_i \tag{1}$$

where ϵ is the error associated with each specific point. We assume that y is the exact value, which is not true but an average value in real experiments. Writing everything in terms of vector format,

$$y = \tilde{y} + \epsilon \tag{2}$$

being each of them is a $n \times 1$ column matrix.

Our model $\tilde{y}$ is constructed from the regression technique based only on the input data and a set of p parameters denoted by β_i, for p independent variables $1, x_{i1}, x_{i2}, ..., x_{ip}$. The only unknowns of the model are the β parameters, whereas independent variables of the model is manually fixed. It can be polynomials, trigonometric functions, exponential functions, etc.

As we are considering linear regression, in matrix notation $\tilde{y}$ is written as,

$$\tilde{y} = X\beta \tag{3}$$

where X is a $n \times p$ matrix and β is a $p \times 1$ column matrix.
Using Eq.(1),

$$y = X\beta + \epsilon \tag{4}$$

To obtain the unknown β parameters that best fits our data, it is defined a new function called cost function.

Cost function (related with the χ^2 function) is defined as the sum of the squared error (deviation of the model from the real value) of each data,

$$C(\beta) = \sum_{i=0}^{n} (y_i - \tilde{y}_i)^2 \tag{5}$$

To make the cost function dimensionless, it is defined the χ^2 function in terms of the variance of y_i as,

$$\chi^2 = \frac{1}{n} \sum_{i=0}^{n-1} \frac{(y_i - \tilde{y}_i)^2}{\sigma_i^2} \tag{6}$$

.

And finally, the model is obtained by adjusting the set of β parameters that minimizes this cost function (so called the ordinary least square method).

In spite of the fact the definition seems to be simple, the minimization procedure is not an easy task in general, as functions in general don't possess analytical forms in all the cases. The minimization is,

$$\left. \frac{\partial C(\beta)}{\partial \beta_j} \right|_{j=1,2,..,p} = 0 \tag{7}$$

that gives for β,

$$\beta = (X^T X)^{-1} X^T y \tag{8}$$

Once the structure of the independent variables X is defined, being $X^T X$ is invertible, it is always possible to obtain the β parameters. In other words, a model is essentially based on the definition of the matrix X, which in specific known as the design matrix [1]. The invertibility of the matrix X is not always possible in many situations, for which the problem is overcome by regression methods given by Ridge and Lasso.

The idea is that, when the design matrix is not invertible, that is when the column vectors are linearly dependent, the matrix becomes singular. So, the basis inverse component of the β parameter $X^T X$ cannot be determined. This problem is overcome by a technique where we simply add additional diagonal terms to $X^T X$ make it possible to find the inverse.

$$X^T X \rightarrow X^T X + \lambda I \tag{9}$$

where I is the identity matrix, and λ is called the penalty parameter.

The β parameter is then minimized to obtain the correct model. In addition, these penalty terms can be tuned in a way such that, it is possible to neglect the variables that has less variance and to decrease the degrees of freedom [2]. In lasso method, it can be

exactly tuned to make them zero whereas the minimization is done by gradient methods.

Once a model is created, the validity and the quality of the model is tested by various methods known as resampling methods. Some of the methods are, cross validation, bootstrap, jack knife, blocking. These methods works on the principle of splitting the whole data into two parts, training and test data. For instance in k-fold cross validation, the data is divided randomly into sets of data of k-groups. Only one set is took for the testing and the rest of them is used for the training. The process is repeated each time including all possibilities until k-times. Finally, the mean values of all the quantities are computed to assess the model. It is computationally expensive for large amount of data, for which other methods such as bootstrap method is more conveniently used.

2.1 Model function

To make our model, first we require the data. The data can be taken from anywhere, for instance from an experiment or can be generated from a function. Here, we took a specific function that is normally used for interpolations known as the Franke function. It is given by,

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

We create our data from this function by adding random noise and build our model based on it. The function is defined in the range $x, y \in [0, 1]$. Since we have two predictors x and y, first we make an ordinary least square regression on this data with a polynomial fit with x and y dependence of all possible combination of different orders up to degree 5.

We also try to analyze it with adding exponential factors that is more suitable to the functional form and try to improve the model of our regression. Then, using similar procedure, we extend it to Ridge and Lasso regression.

3 Implementation and Results

The code was written in Python 3.7 programing language and was interpreted via Anaconda distribution. Calculations were performed on a MacBook Air (Early 2015). portable computer. It has a Intel Core i5 CPU @ 1.6 GHz processor with 8.00 GB of RAM.

The overall code structure represents a set of functions for regression (with and without resampling; linear regression, Ridge, Lasso methods are employed) and plotting which are later called in the program.

Metrics computed in the following sections represent estimates ones as they are calculated on just a given sample of data and not for the whole population.

Here below is present the main parts of the code for the project 1.

Algorithm .1: Project 1 implementation

```
1: Functions;
def OLS(x,y,z);
def plot(x,y,z);
def OLS with k-fold CV(x,y,z,CV number);
for k in CV number do
    Compute MSE, R2 score
end
def Ridge(Lasso) with k-fold CV(x,y,z,CV number,lambdas);
for k in CV number do
    for l in lambdas do
        Compute MSE, R2 score
    end
end
2: Program;
define: CV number, x, y, z = Franke Function + noise, lambdas;
for degrees in range(3,6) do
    for noise in range (0,3) do
        Functions ‘
    end
end
```

3.1 Ordinary Least Square on the Franke function with resampling

First of all, the Franke function was fitted with the ordinary least squares method (OLS). To do so, the dataset ($x_{train}, y_{train} \in [0, 1]$) was defined by random numbers computed with the uniform distribution. OLS regression analysis was performed with polynomials in x_{train} and y_{train} of 3^{rd}, 4^{th}, and 5^{th} orders. The *sklearn.preprocessing.PolynomialFeatures* module was used to generate a new feature matrix consisting of all polynomial combinations of the features with degree less than or equal to the specified degree. z_{train} points were found as Franke function values with the given x_{train} and y_{train}:

$$z_{train} = f(x_{train}, y_{train}) + \mathcal{N}(0,\, \sigma^2), \tag{11}$$

where $x_{train}, y_{train} \in [0, 1]$, $\sigma = 10^{-f}$. Noise was added to simulate a real experiment where it is always present. It was chosen randomly for all points (200 were selected) with the standard normal distribution $\mathcal{N}(0,\, \sigma^2)$.

Consequently, OLS fit was carried out applying *sklearn.linear_model.Linear Regression* to fit a given sample (X_{train},z_{train}) where X_{train} represents a concatenated matrix of x_{train}

and y_{train} (all terms, including cross-coupled ones) as the Franke function is defined in 3 dimensions and data points should be of the 2D size. Once the fit was available, prediction $z_{predict}$ on the same data points (in case of OLS without resampling) was performed.

The first noise added to the Franke function was normally distributed with $\sigma = 1$. However, in this case none of the methods (even including those with resampling techniques discussed later) succeed in providing precise fitting because of large noise. Henceforth, the model was also trained decreasing noise tenfold to obtain more comprehensible results for the task 1.

Variance was calculated to determine confidence intervals for regression coefficients $\hat{\beta}$. The variance-covariance matrix is defined as the following expression:

$$\mathrm{Var}(\hat{\beta}) = \left(\hat{X_{train}}^{T} \hat{X_{train}}\right)^{-1} \sigma^2, \tag{12}$$

where diagonal elements represent variances of the coefficients β. A confidence interval is range of numbers within which the true population parameter is thought to fall with a prespecified probability, denoted by 1-α (α was chosen as 0.05 which corresponds to the probability $p = 95\%$) [3], [4]. The estimation of confidence intervals (as they are calculated within a given sample) was calculated taking $z - score$ for a given observation to be equal to 1.96 (for the chosen probability of 95%):

$$C.I. = \pm 1.96 \sqrt{Var(\hat{\beta})/n}, \tag{13}$$

where $Var(\hat{\beta})$ stands for diagonal variance elements and n is a sample size. Tables 1,2 comprise confidence interval values for polynomials of 3^{rd} and 5^{th} degrees to show the behavior with increasing complexity of fitting polynomials (the table for the 4^{th} degree can be found in the $Project1\_output.txt$ file).

Features	**Coefficients β**	**Confidence intervals C.I.**	**abs(C.I./β)*100%**
1	1.07	0.01	0.9
x	-1.19	0.05	4.2
y	0.87	0.04	4.6
x^2	0.21	0.09	42.9
xy	1.60	0.07	4.4
y^2	-5.25	0.08	1.5
x^3	-0.10	0.05	50.0
x^2y	0.37	0.05	13.7
xy^2	-1.25	0.05	4.0
y^3	3.77	0.05	1.3

Table 1: Confidence intervals for OLS (unsampled), Degree = 3, $\sigma = 0.1$

Features	**Coefficients** β	**Confidence intervals (C.I.)**	**abs(C.I.**/β**)*100%**
1	0.36	0.02	5.6
x	9.19	0.18	20.0
y	4.92	0.20	4.1
x^2	-40.44	0.97	2.4
xy	-15.66	0.69	4.4
y^2	-20.61	0.96	4.7
x^3	60.26	2.26	3.8
x^2y	49.21	1.59	3.2
xy^2	21.16	1.62	7.7
y^3	26.40	2.13	8.1
x^4	-36.05	2.41	6.7
x^3y	-54.57	1.84	1.5
x^2y^2	-9.57	1.45	15.2
xy^3	-38.01	1.82	4.8
y^4	-7.67	2.18	2.8
x^5	6.93	0.96	13.9
x^4y	15.13	0.80	5.3
x^3y^2	17.26	0.75	4.3
x^2y^3	-10.02	0.80	8.0
xy^4	25.66	0.77	3.0
y^5	-3.51	0.87	24.8

Table 2: Confidence intervals for OLS (unsampled), Degree = 5, $\sigma = 0.1$

After that mean squared error and R2 score metrics were calculated using the functionaliy of the $scikit-learn$ (see Tables 3, 4) [5], [6].

$$MSE(\hat{z}, \tilde{\hat{z}}) = \frac{1}{n}\sum_{i=0}^{n-1}(z_i - \tilde{z}_i)^2. \tag{14}$$

R^2 is defined as

$$R^2(\hat{z}, \tilde{\hat{z}}) = 1 - \frac{\sum_{i=0}^{n-1}(z_i - \tilde{z}_i)^2}{\sum_{i=0}^{n-1}(z_i - \bar{z})^2}, \tag{15}$$

where:

- $\bar{z}$ is the mean value of $\hat{z}$
- $\tilde{z}_i$ is the ith predicted value
- z_i is the corresponding true value (in a test set in case of Lasso and Ridge regressions).

Degree	Metrics	Value
3	MSE	1.02
	$R2\ score$	0.07
4	MSE	1.00
	$R2\ score$	0.08
5	MSE	0.99
	$R2\ score$	0.10

Table 3: Metrics for OLS (without resampling), σ =1

Degree	Metrics	Value
3	MSE	0.014
	$R2\ score$	0.81
4	MSE	0.012
	$R2\ score$	0.85
5	MSE	0.011
	$R2\ score$	0.86

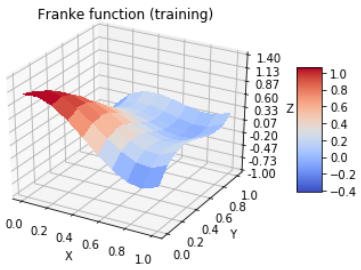
Table 4: Metrics for OLS (without resampling), σ =0.1

Then, the same assessment parameters were calculated once again but the dataset preliminarly underwent 5-fold cross validation (CV). Training data was divided into training and testing sets and then fitted on the training data. The model was developed on the training data resampling it 5 times which resulted in 5 values of metrics. Later on, their mean values were computed applying the model on the test data (20% of all points). Results are presented in the Table 5.

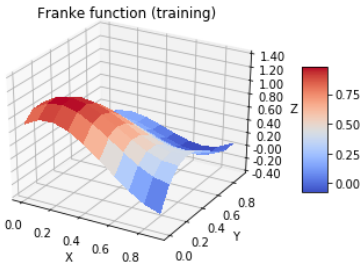
Degree	Metrics	Value
3	MSE	0.016
	$R2\ score$	0.777
4	MSE	0.014
	$R2\ score$	0.804
5	MSE	0.014
	$R2\ score$	0.804

Table 5: Metrics for OLS (with 5-fold cross.val. resampling), σ =0.1

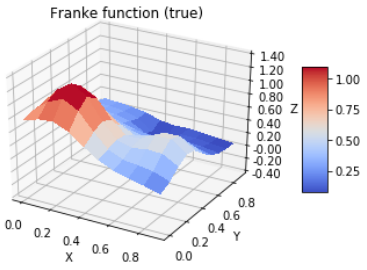
Plots were made creating a function $plot(x, y, z)$ which input parameters represent a smaller data set in order to spare computer's RAM and make faster computations. Visualization of results is performed for the unsampled case (standard OLS, $\sigma(noise) = 0.1; 1$, training data) and for the Franke function without noise (true data). Plots Fig.4a,2a,3a which represent the case of $\sigma(noise) = 1$ differ noticeably with those with a tenfold less noise (see Fig. 4b, 2b, 3b).



(a)

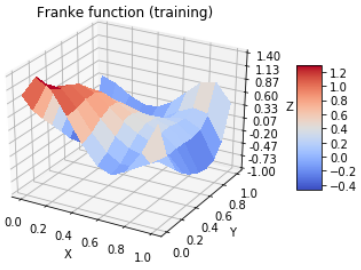


(b)

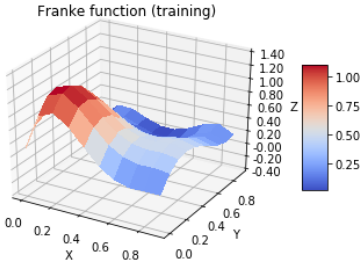


(c)

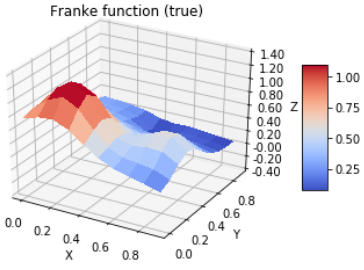
Figure 1: Degree 3 OLS fit without resampling: (a) σ=1, (b) σ=0.1, (c) Franke function



(a)

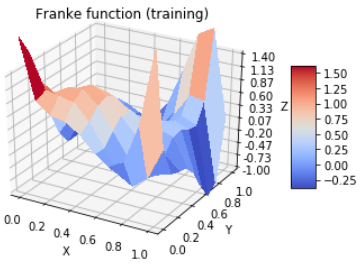


(b)

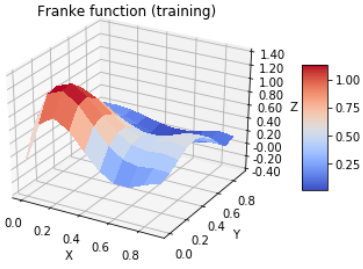


(c)

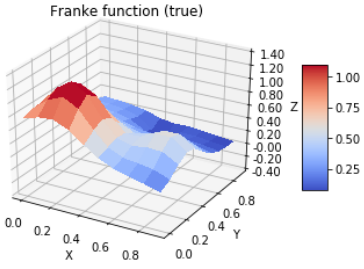
Figure 2: Degree 4 OLS fit without resampling: (a) σ=1, (b) σ=0.1, (c) Franke function



(a)



(b)



(c)

Figure 3: Degree 5 OLS fit without resampling: (a) σ=1, (b) σ=0.1, (c) Franke function

3.2 Ridge Regression on the Franke function with resampling

The next steps of the Project 1 suggest introducing other regression techniques which fit the data with a regularization parameter. It is used to change the function shape in order to avoid correlated variables and make fit less variant to noise addition.

First, an array of regularization parameters λ is created: [10^{-5}, 10^{-3}, 0.1, 10]. The *Ridge.sklearn.linear_module* was used to perform Ridge regression. To study the dependence on λ while varying eventually the strength of the noise in your expression, two loops

were required: over all lambda values and 5 times of cross-validation resampling. When $poly.fit\_transform$ was applied on the data, the intercept term (an array of ones) was not created as Ridge module creates it by itself.

As loops run over, two-dimensional arrays (k - the number of resamplings and λ - regularization parameter) of MSE and $R2\ score$ are filled. For each row corresponding to a certain λ mean values of the metrics were calculated. Afterwards, best regularization parameters were chosen based on the maximum value of $R2\ score$ (the higher it is, the more precisely a model describes a given dataset).

Tables 6, 7 represent behavior of metrics for different λ in case of two σ(noise) values: 0.1 and 0.01 ($\sigma = 1$ is omitted as it results in $R2\ score$ values close to zero, though, assessment parameters for this case can be found in the $Project1\_output.txt$ file).

Degree	**Metrics**	$\lambda_1 = 10^{-5}$	$\lambda_2 = 10^{-3}$	$\lambda_3 = 0.1$	$\lambda_4 = 10$
3	MSE	0.017	0.017	0.020	0.030
	$R2\ score$	0.793	0.777	0.73	0.60
4	MSE	0.014	0.016	0.019	0.030
	$R2\ score$	0.836	0.79	0.76	0.60
5	MSE	0.014	0.015	0.017	0.030
	$R2\ score$	0.863	0.80	0.76	0.60

Table 6: Metrics for: Ridge regression, k(cross.val.)=5, σ=0.1

Degree	**Metrics**	$\lambda_1 = 10^{-5}$	$\lambda_2 = 10^{-3}$	$\lambda_3 = 0.1$	$\lambda_4 = 10$
3	MSE	0.007	0.007	0.012	0.024
	$R2\ score$	0.90	0.90	0.82	0.65
4	MSE	0.005	0.006	0.010	0.024
	$R2\ score$	0.93	0.91	0.85	0.65
5	MSE	0.003	0.005	0.009	0.024
	$R2\ score$	0.96	0.93	0.87	0.65

Table 7: Metrics for: Ridge regression, k(cross.val.)=5, σ=0.01

3.3 Lasso Regression on the Franke function with resampling

This part is essentially a repeat of the previous two ones, but now with Lasso regression. The way of fitting is the same as in the part 3.2. The only difference which affected the choice of lambdas was *convergence warning* which appeared when a lambda was lower than 10^{-5}. The set of lambdas was also the same and it was creates as the following array: $\lambda = [10^{-5}, 10^{-3}, 0.1, 10]$. Here below are visualized the results for 2 values of noise σ: 0.1 and 0.01 (see Tables 8, 9).

Degree	**Metrics**	$\lambda_1 = 10^{-5}$	$\lambda_2 = 10^{-3}$	$\lambda_3 = 0.1$	$\lambda_4 = 10$
3	MSE	0.017	0.024	0.086	0.086
	$R2\ score$	0.791	0.70	-0.05	-0.05
4	MSE	0.017	0.021	0.086	0.086
	$R2\ score$	0.794	0.74	-0.05	-0.05
5	MSE	0.016	0.020	0.086	0.086
	$R2\ score$	0.802	0.75	-0.05	-0.05

Table 8: Metrics for: Lasso regression, k(cross.val.)=5, σ=0.1

Degree	**Metrics**	$\lambda_1 = 10^{-5}$	$\lambda_2 = 10^{-3}$	$\lambda_3 = 1$	$\lambda_4 = 10$
3	MSE	0.008	0.015	0.075	0.075
	$R2\ score$	0.889	0.77	-0.05	-0.05
4	MSE	0.008	0.013	0.075	0.075
	$R2\ score$	0.887	0.82	-0.05	-0.05
5	MSE	0.007	0.012	0.075	0.075
	$R2\ score$	0.894	0.82	-0.05	-0.05

Table 9: Metrics for: Lasso regression, k(cross.val.)=5, σ=0.01

4 Analysis

4.1 OLS

As was proposed in the first task of the project, the first noise distribution which was added to the $Franke\ function$ was standard normal distribution $\mathcal{N}(0,\ 1)$. It can be seen from the Table 3 R2 score values are close to 0 which means high difference between the trained model and real function. The same order values were also observed in case of resampling techniques (see the $Report1\_output.txt$ file for more data). To interpret better the results and make wise comparisons, the σ(noise) was lowered and the results were found to fit much better the true function (see Table 4).

Later on the 5-fold cross-validation was applied to the dataset. Resampling techniques are used to avoid overfitting and provide results less variant upon the changes in the training data (i.e. to different random errors). Thus, a computed model is more reliable. As one can see in the Table 5 obtained $R2\,score$ values are lower than in the case of the unsampled data, which may look unclear, from one side. From the other side, it should be considered resampling is performed on less data (80% of the whole dataset) which results in lower $R2\,score$ values and higher MSE ones, however, the goal is to build a more stable model which results one should trust more.

Confidence intervals β were also calculated and results are presented for 3^{rd} and 5^{th} degrees (unsampled case) in the Tables 1, 2. One may notice that the ratio $C.I./\beta$ happens to reach 40-50% in same case for the 3^{rd}, while for the 5^{th} one it is less than 25% for

all coefficients (for this randomly distributed data). It can be concluded the higher a polynomial degree is, the better a model fits the true function (at least until the 5^{th} degree). But further analysis should carried out in order to find a degree when higher complexity starts becoming a drawback rather than an advantage for building a model (there is always a tradeoff between model fit and model complexity).

4.2 Ridge

In the case of Ridge regression method, the same analysis was performed as in the previous exercise. It can be noticed Ridge slightly increases R2 score value for the lowest value of lambda studied(see Table 6) with respect to the OLS 5-fold cross validation (Table 5). It can be explained since addition of a regularization parameter λ makes the model more stable to noise and, therefore, a predictor describes better the true function.

Lambda dependence on noise was also studied, the results are present for two values of σ (0.1 and 0.01). One can deduce the lower the applied noise is, the more precise fit is obtained, what is obvious as beta variance becomes smaller and there is less dispersion of data points.

Best lambda (for the highest value of R2 score) was found to be either 10^{-5} or 10^{-3} for different degrees of polynomials. When it approaches λ=1 and 10, the precision gets lower so does the R2 score. However, more analysis is required to choose a lambda more accurately, that is dividing small lambdas region into more points (with *logspace*) and looking again for highest R2 score values to improve the model selecting the best lambda.

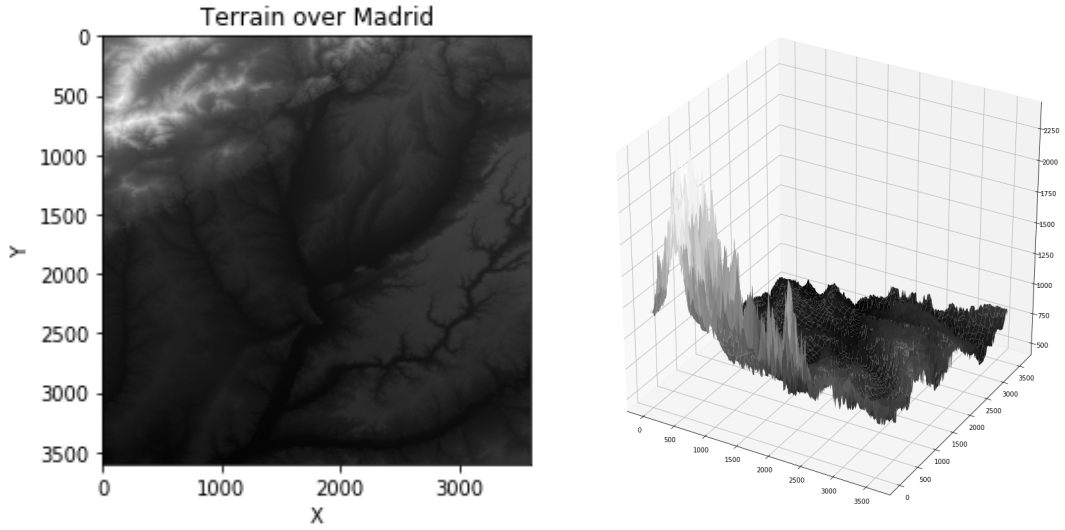
4.3 Lasso

The overall behavior of the Lasso method coincides with that of the Ridge one: the quality of fitting increases when a degree of polynomials increases and decreases when λ becomes lower. Lasso regression displayed quite the same results as the Ridge method for the lowest value of lambda used λ=10^{-5} in the case of σ=0.1 (Tables 6, 8). However, it caused lower $R2\ scores$ for higher values of lambda. For the smaller value of noise σ=0.01, once again, statistically significant results are those who belong to the case of $\lambda = 10^{-5}$ (Table 9). The results for $\lambda = 10^{-3}$ differ for more than 10% with Ridge method values (Table 7).

For λ values of 0.1 and 10 (for both values of noise) Lasso regression brought negative $R2\ score$ values while Ridge method resulted in 0.6 value for $\sigma = 0.1$. Therefore, it can be seen Lasso regression quite fast reaches a good solution for low values of λ, but causes negative $R2\ score$ values when we increase it too much. Ridge is more stable over a larger range of values for λ, but eventually also fades away.

5 Application of the model to real data

As we have comprehensively analyzed our model in the previous sections, now we would like to apply it to a real data. So, we consider the freely available digital terrain data taken from the earth explorer website [7], which is one of the world's largest civilian collection of images of the Earth's surface. It consists of satellite images and data, aerial photography, elevation and etc. Using this, in specific we consider the region of Madrid in Spain and try to make the regression analysis on this data (see Fig. 4).



(a) (b)

Figure 4: Madrid terrain: (a) 3D histogram, (b) view from upside

This final part deals with the parameterization of the digital terrain data. All three methods for linear regression were applied like in the previous part of the Project. Polynomial approximation and resampling techniques (5-fold cross-validation) were used to ass different models.

Real data processing was found to be computationally expensive as it includes fitting with polynomials of $10^{th} - 100^{th}$. In order to achieve precise results and reduce computation time, one value of regularization parameter λ was selected. To do so, a rough estimation was made taking the following λ values: $[10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}]$, and the highest $R2$ *score* was found to belong to the $\lambda = 10^{-6}$. This value was used for further computations (see Fig. 5).

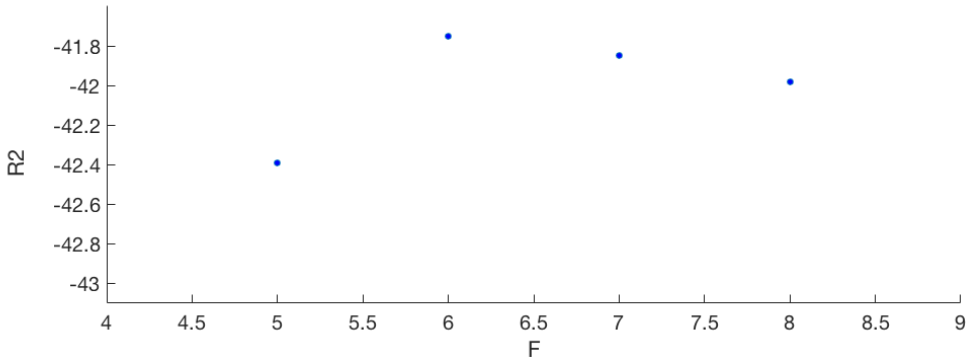


Figure 5: R2 score vs. $\lambda = 10^{-F}$, *Ridge regression with* $5 - fold\, CV$

Consequently, different regression techniques were applied on the training data as can be seen in the Table 10. Due to long computation times for degrees higher than 100, 3 points were considered: $20^{th}, 50^{th}$, *and* 100^{th} degrees. However, of the standard OLS $R2$ *score* was also computed for the 200^{th} polynomial degree and was found to increase with respect to smaller degrees. It may occur due to many peaks and uplands of the terrain data set which lead to very complex models needed to obtain better metrics. It can be suggested trying non-linear regression to fit the data better.

Resample techniques fail to display the same pattern as the unsampled one. In fact, the current data set includes 3601x3601 points which are hardly to be well-sorted and -analyzed after just 5 resamplings.

Degree(polynomial)	**OLS**	**OLS resamp.**	**Ridge**	**Ridge resamp.**	**Lasso**
20	0.05	-52.43	0.0045	-42.11	0.0003
50	0.12	-3841.87	0.0066	-43.15	0.0003
100	0.09	$-1.54*10^{10}$	0.0075	-58.76	0.0003

Table 10: R2 score values (k(CV)=5, $\lambda = 10^{-6}$)

$R2$ *score* (OLS) = 0.196 (degree = 200).

Moreover, the most prominent outcome (after that of the unsampled OLS) was observed with unsampled Ridge Regression which behavior followed OLS's one ($R2$ *score* increases with increasing polynomial degree) but results are less statistically significant as they around 10 times lower than OLS $R2$ *scores*. With the given λ and the number of resamplings $k(CV) = 5$ other models yielded in large negative values. An attempt was made to increase the number of resampling but, to the contrary, it caused a drastical decrease in $R2$ *scores*. The Lasso method with resampling techniques was ignored in the end, as it needed 30-60 minutes to generate each value, while, for example, the resampled Ridge method required max. a couple of minutes even for the degree 100.

It can be summed up the trained model can hardly be described with a low-complexity model ($R2 : = 0.196$ for the 200^{th} polynomial degree). It also requires much time for

troubleshooting and code processing. On may try to consider possible peaks on the flat terrain by introducing the delta-function in polynomials.

6 Conclusion

The main aim of this project was to study in more detail various regression methods, including the Ordinary Least Squares (OLS) method, Ridge regression, and, finally, Lasso regression. The methods were in turn combined with resampling techniques.

Training points were described with the $Frankefunction$ plus normally distributed random noise. Linear regression was performed for 3^{rd}, 4^{th}, and 5^{th} degrees of polynomials and best metrics values were in the case of the 5^{th} degree.

It has been shown Ridge regression with 5-fold cross validation improves cross-vaildation on OLS for the regularization parameter $\lambda = 10^{-5}$. The same method but with $\lambda = 10^{-3}$ resulted in almost the same metrics, so did the Lasso regression for $\lambda = 10^{-5}$. However, the Lasso regression did not converge for smaller λ and also did not succeed in fitting for $\lambda = 0.1, 10$ bringing negative $R2\,score$ values.

For the dataset which describes the Madrid's terrain (its level above the sea), it is challenging to perform a precise fit due to the data inhomogeneity. Nonetheless, OLS method is guessed to obtain higher values of $R2\,score$ by increasing the polynomial degree (for degree = 200, $R2\,score$=0.196). Ridge regression repeated the OLS's behavior but with 10 times lower values. Another non-linear method should be implemented to train upon the given dataset to ensure higher precision and smaller complexity. Also, peak regions may be separated and treated diversely.

Resampling methods are processes of repeatedly drawing samples from a data set and refitting a given model on each sample with the goal of learning more about the fitted model. They are to estimate descriptive statistics and confidence intervals from sample data when parametric test assumptions are not met. Their usage prevents from over- and underfitting, thus ensuring higher stability to external conditions such as noise. Methods which introduce a regularization parameter (e.g. Ridge and Lasso regressions) assist in avoiding correlated variables. Both resampling and regression techniques are carried out much easily using $scikit-learn$ and $numpy$ modules in Python.

References

[1] Morten Hjorth-Jensen, Regression methods https://compphysics.github.io/MLErasmus/doc/pub/Regression/html/Regression.html

[2] Ryan Tibshiraniurl, Modern regression 2: The lasso http://www.stat.cmu.edu/~ryantibs/datamining/lectures/17-modr2.pdf

[3] Lisa Sulliva, Confidence Intervals http://sphweb.bumc.bu.edu/otlt/MPH-Modules/BS/BS704_Confidence_Intervals/BS704_Confidence_Intervals_print.html

[4] Tirthajyoti Sarkar, Machine Learning with Python: Easy and robust method to fit nonlinear data https://towardsdatascience.com/machine-learning-with-python-easy-androbust\-method-to-fit-nonlinear-data-19e8a1ddbd49

[5] Trevor Hastie, Robert Tibshirani, Jerome Friedman. The Elements of Statistical Learning (2009) Springer

[6] Aurélien Géron. Hands-On Machine Learning with Scikit-Learn and TensorFlow (2017) O'Reilly Media.

[7] https://earthexplorer.usgs.gov