UNIVERSITÉ DE NANTES

Autonomous Work 2 Report
Learning Theory

Camille-Amaury JUGE

Master 2 Data-Science.

Summary

1. Introduction

In this section we will briefly remind the subject. First of all, we had to deal with Polynomial Regression, which is a generalization of the Linear Regression Theory. Nextly, we had to implement and understand the basics of validation on a model using indicators and cross-validation. Then, our goal was to discover and find the right way to use polynomial regression and how to best evaluate the result of those models.

Thus, we propose a basic approach following the question raised by the subject :

- We will firstly explore data using visualization methods and try to give intuitions on the subject.

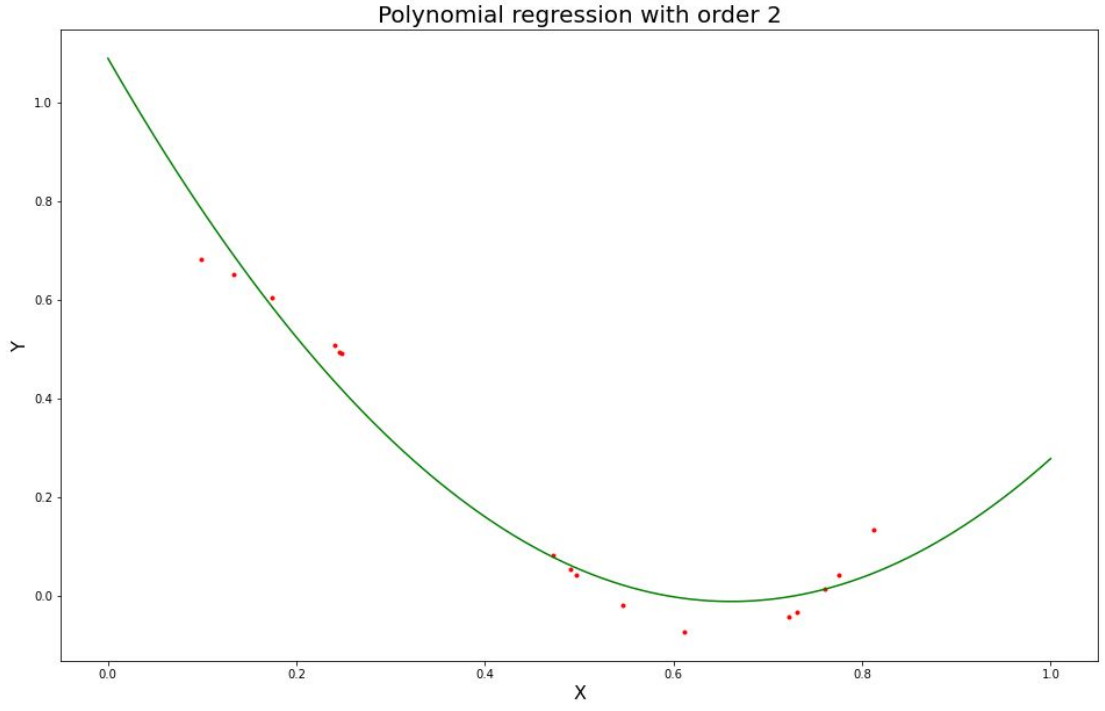
- Finally, we will explore an improved way to evaluate our model and give accurate conclusions.

N.B **:** All the experiments and plots displayed in this report will be available using a python compiler and the “crossval.py” file provided. Those are all made by Camille-Amaury Juge.

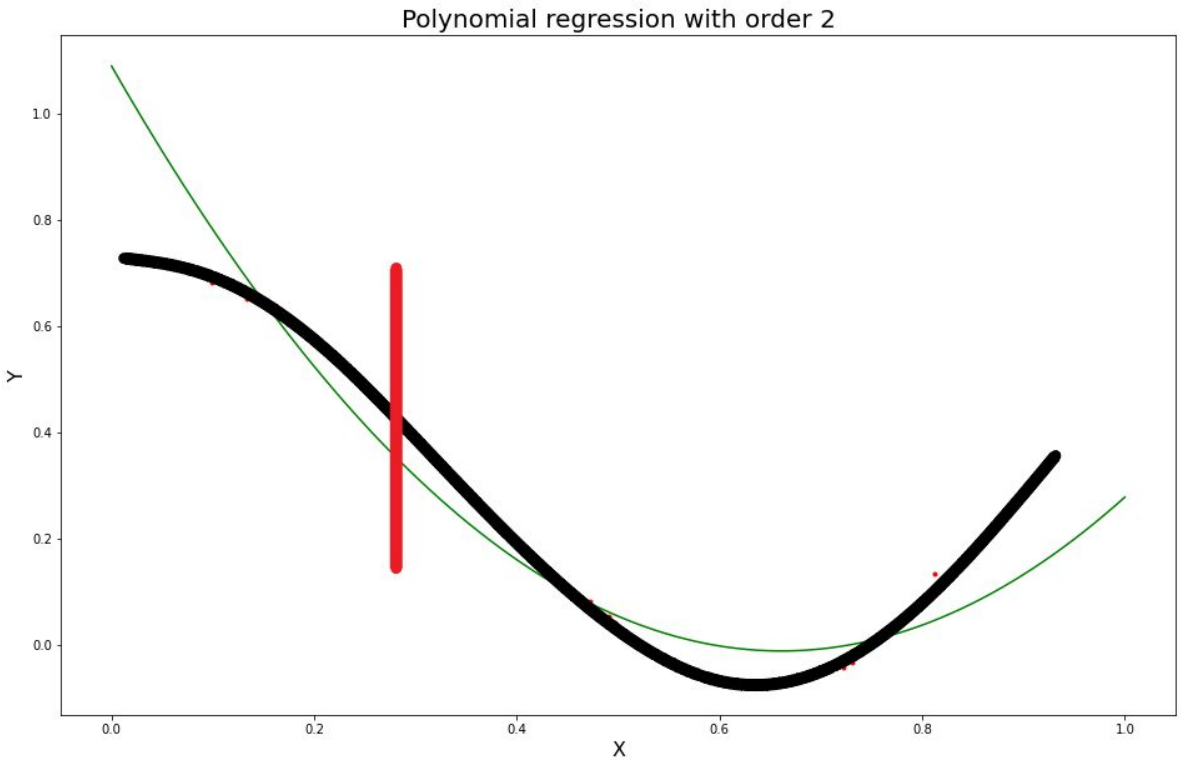
2. Visualization

In this section, we will try to develop a first impression of our dataset and on how polynomial regression is working on it.

A. Intuition



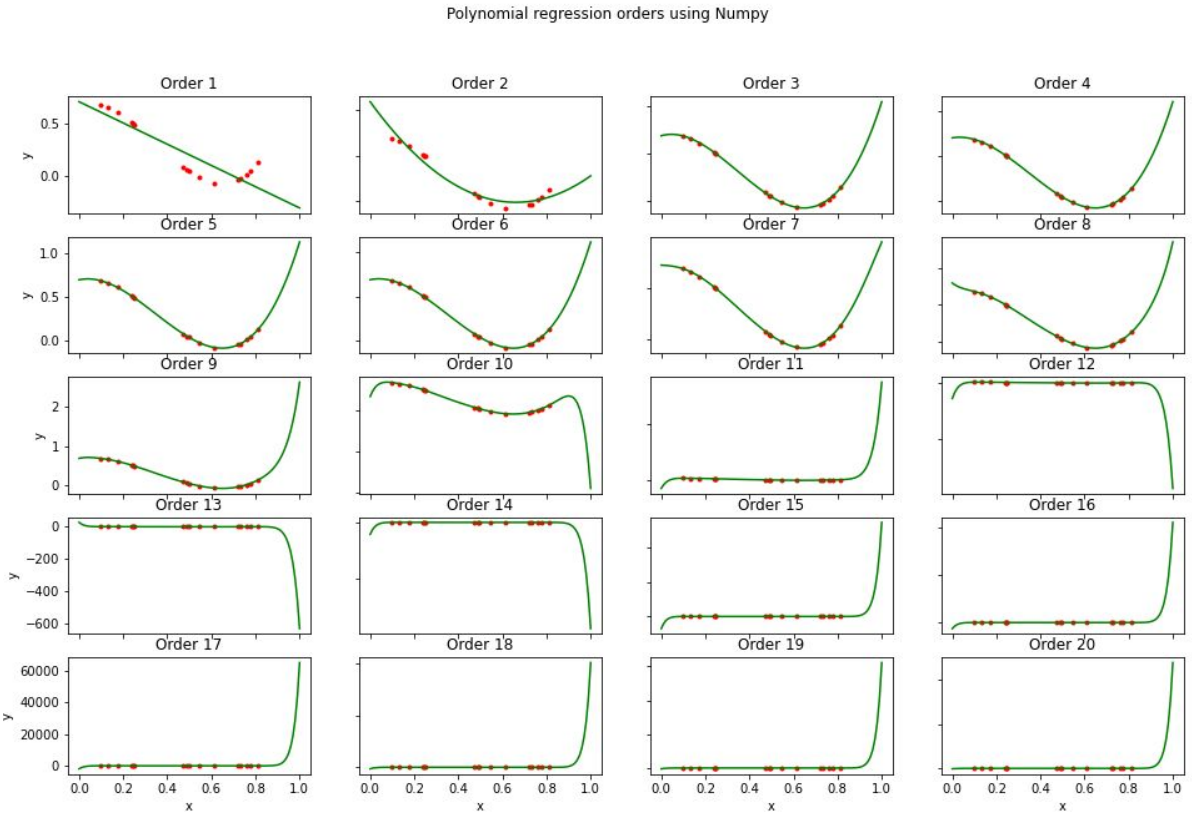
In this first example, we used a method to compute the Polynomial Regression with the initial script we were given. As we can see, a second order polynomial regression seems not to fit well the data. In order to be clear, our first group of data (represented by the points before the red line) seems to depict a concave function, while our second group (represented by the points after the red line) depicts a convex function (regarding figure below).



Thus, we can easily be persuaded that the order of the polynomial function which will best represent the data is at least greater than 2. It is maybe a bit late to say that the order 3 should be a good solution and that's why we are going to explore other orders to improve our intuition.

B. Intuition Analysis

We will now generalize the previous visualization to provide us more information using the polynomial regression from the first to twentieth one (regarding figure below)



As the plots show, our first intuition seems to be a valid one. Both Linear Regression and Second Order Polynomial Regression don't fit the data well. But an interesting thing observed here is that over the ninth order, the Polynomial Regression is not accurate at all too. It means that, in order to find our best model order, we should concentrate our research on low orders rather than greater ones.

```
> Order(1) => squared error : 0.23444624114878998
> Order(2) => squared error : 0.045527890679536144
> Order(3) => squared error : 4.675072465134565e-05
> Order(4) => squared error : 9.390247544466868e-06
> Order(5) => squared error : 8.785086324615368e-06
> Order(6) => squared error : 8.784226147040216e-06
> Order(7) => squared error : 8.271588653006233e-06
> Order(8) => squared error : 7.620955413879173e-06
> Order(9) => squared error : 7.057269388462692e-06
> Order(10) => squared error : 6.9141569109996424e-06
> Order(11) => squared error : 5.160400704497333e-06
> Order(12) => squared error : 2.1883091964800006e-06
> Order(13) => squared error : 1.92032832432285e-06
> Order(14) => squared error : 1.838792251016609e-06
> Order(15) => squared error : 2.145315984408519e-10
> Order(16) => squared error : 2.8601417003342813e-12
> Order(17) => squared error : 7.896361080604821e-13
> Order(18) => squared error : 1.348125583422131e-13
> Order(19) => squared error : 2.0961940919521353e-13
> Order(20) => squared error : 1.7451988158208622e-14
```

We now compute the Squared Error on each of those orders (regarding figure above), and we can see that the greater the order is, the lower the error is. But it doesn't mean that the 20 order is the best one following what we have seen before. In fact, It probably says that those high orders overfit our data and we will be bad at generalizing the behavior of the function we try to approximate.

Moreover, we could follow the elbow method to see that there is a real improvement between the second and third order, we divide the squared error by 1000. This is the greatest difference between the squared errors orders (with the 14th and the 15th ones). And it may confirm our intuition that the third order seems to fit pretty well our data.

We will then explore a better method to ensure that our intuition is not wrong.

3. Evaluation

In this section, we will try to prove that the first order seems to be the best solution to approximate the data.

A. Cross-Validation

In our path to reach the best solution, we had to understand and implement cross-validation methods. The aim was to avoid overfitting problems that we raised in Section 2, by firstly using the Hold-out method. It is basically separating the dataset with a fixed percentage between Training dataset (which will be used to compute the Polynomial Regression) and Testing dataset which will be used to compute the squared errors. It means that we use data that has never been seen during the training and then, we better evaluate the ability of the model to generalize and not overfit the data.

But rather to simply choose continuous indexes in the splitting process, we used a random sampling method which could be better in some cases : Let's imagine that we have every first group's points (regarding figure 2, section 2.A) only in our training set and second group's points in the testing set. We will not have accurate training due to its sorting. But we can also generalize this idea to a classification problem. Then, selecting random samples based on the percentage will help us to avoid this problem (at least the probabilities of having all the classes or the same group points are really low).

```
> Order(1) => Mean empirical se : 0.13081964160678863
> Order(2) => Mean empirical se : 0.025744032984352797
> Order(3) => Mean empirical se : 1.2168963772658352e-05
> Order(4) => Mean empirical se : 8.1154629594782e-06
> Order(5) => Mean empirical se : 9.620507654947486e-06
> Order(6) => Mean empirical se : 0.0002431098947409953
> Order(7) => Mean empirical se : 1.0780012343033075e-05
> Order(8) => Mean empirical se : 4.637292172385648e-05
> Order(9) => Mean empirical se : 0.01675899866805846
> Order(10) => Mean empirical se : 81.97152114409872
> Order(11) => Mean empirical se : 0.6843731529581916
> Order(12) => Mean empirical se : 0.0024669510123890553
> Order(13) => Mean empirical se : 262.3526706285454
> Order(14) => Mean empirical se : 2177679.480739961
> Order(15) => Mean empirical se : 11.561820906608661
> Order(16) => Mean empirical se : 0.004114603822957102
> Order(17) => Mean empirical se : 104.46923214732593
> Order(18) => Mean empirical se : 67.84539655155794
> Order(19) => Mean empirical se : 0.015939872930176215
> Order(20) => Mean empirical se : 1.398948247065996

Best Order(4) with se = 8.1154629594782e-06
```

Above, the results show clearly that our intuition was correct : orders greater than ninth order leads us to a really high squared error (we can even say seventh order). As we may have predicted, the best order is in this case 4 which is really near 3. But one thing to mention here is that a unique splitting of the dataset doesn't ensure us that our splitting is good. As we mention it, there is still a low probability of having a really bad training and testing set. To avoid this problem, we then propose to explore the K-Fold method.

N.B **:** If you try to execute the script, you won't probably have the exact same result due to randomness splitting.

B. Interpretations

The K-Fold method needs to have a k parameter which will decide on how much sub-dataset we will divide our initial dataset. Then, we obtain k subdataset and we will iterate k times with the idea of :

- Concatenating k-1 sub-datasets as the training set
- Taking the last sub-dataset as the testing set.
- Computing the error's metric

Then, we can take the average of all the stored error's metric of the k iteration (empirical error) and that would give us a more robust metric to evaluate the approximation of our function.

In our implementation, we also choose to use random sampling splitting for the same reason that was told before.

```
> Order(1) => Mean empirical se : 0.295756222917857
> Order(2) => Mean empirical se : 0.1058902527472993
> Order(3) => Mean empirical se : 0.0006428366441889181
> Order(4) => Mean empirical se : 0.0008610424637278296
> Order(5) => Mean empirical se : 0.001584479226830427
> Order(6) => Mean empirical se : 0.5896382304893379
> Order(7) => Mean empirical se : 8057.627419717876
> Order(8) => Mean empirical se : 3936.8890562286315
> Order(9) => Mean empirical se : 2061.198184865467
> Order(10) => Mean empirical se : 1149.7016186114834
> Order(11) => Mean empirical se : 680.7238563909784
> Order(12) => Mean empirical se : 428.09105273248673
> Order(13) => Mean empirical se : 288.4049934028777
> Order(14) => Mean empirical se : 212.58443063033152
> Order(15) => Mean empirical se : 177.1994191681461
> Order(16) => Mean empirical se : 171.90374053140653
> Order(17) => Mean empirical se : 193.7450758938678
> Order(18) => Mean empirical se : 244.57147004365294
> Order(19) => Mean empirical se : 329.90654281771543
> Order(20) => Mean empirical se : 458.56122723224604

Best Order(3) with se = 0.0006428366441889181
```

Thus, we obtain the third order as the best approximation of our function. But rather to conclude on this result, we will go further. In fact, if you do several repeat K-Fold Cross Validations you won't have 3 as the best order, we often find 4 or 5 as the best result and sometimes 3 when the random splitting enables it. So how is it possible ?

The explanation can come from two part :

- First thing already mentioned, we use randomness, then it makes the result change.

- Lastly, our dataset is really small as we can see on plots. It means that having to split the dataset will decrease performance of models and chances to have a quite constant result. The more you have data, the more you are likely to find the same order in each several K-Fold methods you apply.

N.B **:** The empirical error is always the Squared errors metric, we will probably find another result with another metric, but it should still remain approximately the same (not choosing the 20th order for example but rather the 5th or 6th maybe).