CW1: An Unknown Signal

Rares Bucur

1. Introduction

This assignment involved taking various concepts of statistics and data science for use in a data modelling scenario. The program takes as input a .csv file containing a set of data points and decides whether each line segment represents the correct type of function, in this case, if it is a linear, polynomial or an unknown function. I decided to implement my solution with the Chebyshev method because it was more straight forward and efficient, dealing with non-linear functions as well, from the beginning of the process.

2. Functionality

Firstly, I started by defining the basic functions of the program. I used the linear regression method presented in the notebooks and began by implementing the "***fit_Wh(X, Y)***" function which calculates the inverse of the matrix (***np.linalg.inv(X.T.dot(X)).dot(X.T).dot(Y)***). Following the steps in notebook 3, I am acquiring the coefficients for the linear and polynomial functions using the Chebyshev function "***np.polynomial.chebyshev.chebval***". After that, I am creating a matrix of different coefficients and returning its transpose.

I am determining how many line segments are in every .csv file and based on the number of segments, the new matrices *(**"new_X", "new_Y"***) are created.

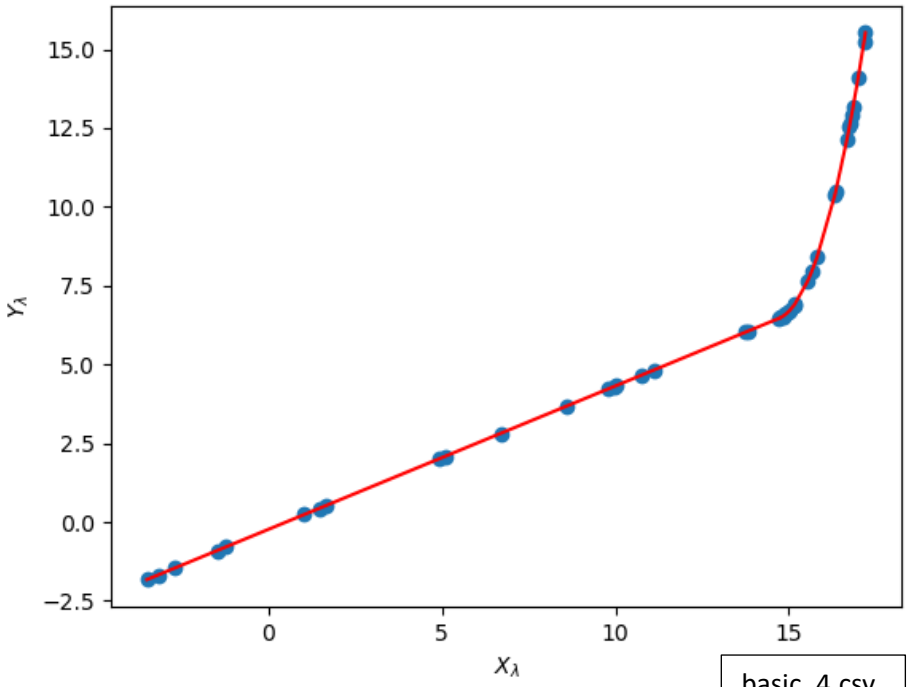
For approximating the function type and dealing with overfitting because line segments would almost always be modelled using a high degree polynomial, I used cross-validation. After trying multiple solutions, I decided to split the data points as follows: the first 15 points into a training set ***("X_train", "Y_train"***) and the last 5 points into a test set ***("X_test", Y_test")*** based on which the cross error will be calculated.

Depending on the order, I am fitting the lines nicely with the help of the trained and tested data sets. I am calculating the cross validation error using the formula from the notebooks, in my case: *"**cross_validation_error = ((Y_test – Yh_test) ** 2).mean()**"*. Then, I am collecting the errors in a vector named *"**errors**"*.

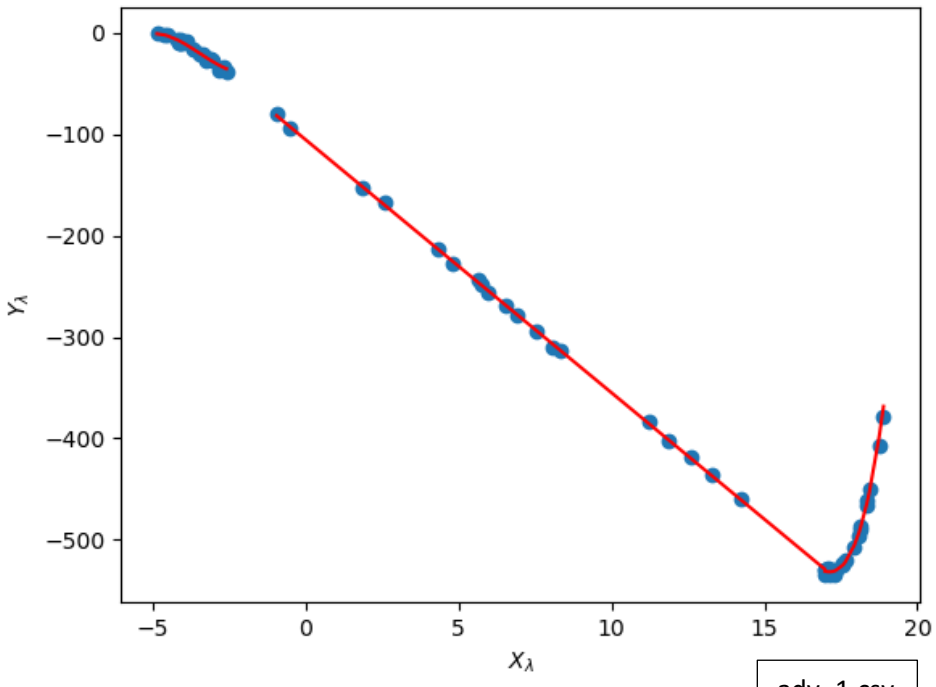
With the aim of determining the order of each line segment as well as which type of function each line represents, I am extracting the lowest error from the vector to later compare it with the *"**cross_validation_error**"* of the sin function.

So, for example if I have a vector of approximated errors [4, 3, 5, 2, 3], the program takes the value "2" being the lowest error from the vector.

At this point, the program was only able to determine if a line segment appeared to be linear or polynomial:



basic_4.csv



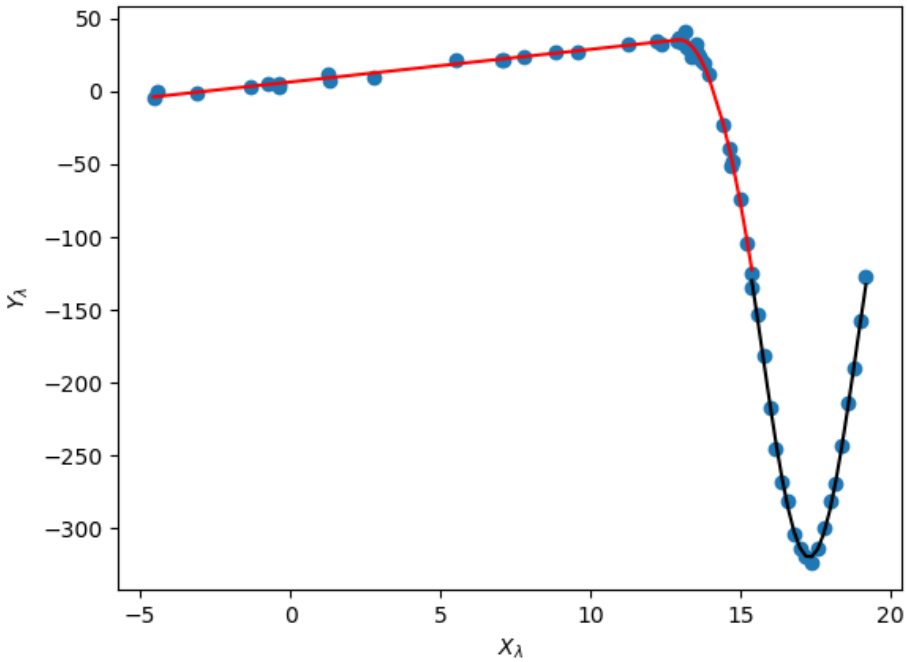
adv_1.csv

In this case of the *"**basic_4.csv**"* file, we have one linear function and one polynomial function of third order with a total error of *5.411054383189859e-11*.

In the example of the *"**adv_1.csv**"* file, we have two polynomial functions of third order and one linear function in the middle with a total error of *393.0007820531433*.

After plotting all the datasets and seeing all orders of the polynomial functions, I stated that in most of the cases and in all the basic cases, the order of the polynomial function is **3**, so this must be the true function based on majority of cases tested. In some files where there might be some noise, the polynomial function would plot better if it were of order 2.

Using the functionality provided by the state of my program after implementing a robust validation system, I set about experimentally determining the unknown signal. This was discovered later to be of type sin. The Chebyshev and regression function in the code were expanded to include a case for generating a sin-based plot. Therefore, with the unknown function being of type *"f(x) = a*sinx + b"*, I am determining the signal using *"np.sin()"* (*"**Yh_test = chebX(np.sin(X_test), 2).dot(fit_Wh(chebX(np.sin(X_train), 2), Y_train"))***) *and making a cross validation error for it.*



noise_3.csv

As seen above, the file *"**noise_3.csv**"* has one linear function, one polynomial and one sinusoidal of second order.

After successfully finding and plotting each function, I took the lowest error from the vector of cross validation errors and compare it to the cross error calculated only for the sinusoidal function. If the error from the vector is lower than the sin cross error then, the program plots the linear and polynomial functions in color red, otherwise, it plots the sinusoidal functions as well in color black.

3. Conclusion

This project helped me experience how easy it is to misrepresent a dataset by using an overfitting model. The usage of training and test sets helped solve this issue.

A downside to my approach of this assignment would be that I did not use a K-fold implementation for the cross-validation method, which would have better examined the dataset to model it more consistently. It was too late when I realized that a solution to this problem would be to shuffle the dataset before making the matrices and make the tests based on those matrices with a new flushed dataset every time the code is run.

To conclude, I believe that this assignment has been a success. To my knowledge, all datasets are modelled as accurately as possible based on my approach with dealing with overfitting by using the cross-validation method.