Symbols, Patterns, Signals Coursework 1: An Unknown Signal Report

Lik Wong

20th May, 2020

1. Introduction

For this task, the Least Square Method (LSM) is utilised to approximate a line given a sequence of coordinates. The objective of the task is to minimise the Residual Sum of Squares (RSS) or the sum of squared errors in order to provide the most accurate solution.

The program implemented reads from a file with a set of points and uses LSM once for each function type, namely linear, polynomial and sine per segment (20 points). Afterwards, it decides which type of function each line segment will be by selecting the one with the least RSS. This is done on training data initially to work out function types and improve on accuracies, which we then move on to more advanced and broader data sets.

2. Method

The general equation for LSM in its matrix form is as follows with the input coordinates' respective x and y components:

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad X = \begin{bmatrix} 1 & x_1^1 & x_1^2 & \cdots & x_1^p \\ 1 & x_2^1 & x_2^2 & \vdots & x_2^p \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n^1 & x_n^2 & \cdots & x_n^p \end{bmatrix}$$

Then we calculate the coefficient vector using:

$$A = (X^T * X)^{-1} * X^T * Y$$

and regression line $\hat{y} = a_0 + a_1 x + a_2 x^2 + \cdots + a_p x^p$ where each a_i is an element of A.

From the above, n is the number of points and p is the power of the regression line. Hence, this implies that the equation works for any polynomial function by changing the value of p. In this case, powers for linear and cubic are chosen for our solution ($p = 1\ or\ 3$).

Nonetheless, given that the unknown function is not polynomial, the previous equation won't satisfy it. Through using a combination of trial-and-error and observation on the files with an unusually large RSS (basic_5.csv, adv_3.csv and noise_3.csv), the unknown function was determined to be sinusoidal. We have to, therefore, alter the general equation:

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad X = \begin{bmatrix} 1 & \sin(x_1) \\ 1 & \sin(x_2) \\ \vdots & \vdots \\ 1 & \sin(x_n) \end{bmatrix}$$

Then we calculate the coefficient vector using:

$$A = (X^T * X)^{-1} * X^T * Y$$

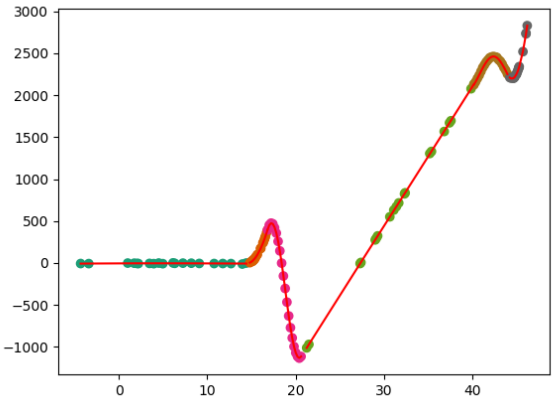
and regression line $\hat{y} = a_0 + a_1 \sin(x)$ where each a_i is an element of A.

3. Analysis

The result from all the datasets with their respective function types are listed in the table below:

File Name	RSS (to 3 d.p.)	Function Type for Each Segment (in order)
basic_1	1.681×10^{-27}	linear
basic_2	6.467×10^{-27}	linear, linear
basic_3	2.923×10^{-18}	cubic
basic_4	2.761×10^{-13}	linear, cubic
basic_5	2.496×10^{-25}	sine
adv_1	198.232	cubic, cubic, cubic
adv_2	3.651	sine, cubic, sine
adv_3	1018.635	cubic, cubic, sine, cubic, sine, cubic
noise_1	10.985	cubic
noise_2	797.917	cubic, cubic
noise_3	477.699	cubic, cubic, sine

There are a few notable features shown on this table. Firstly, it is apparent that all the basic files have a measly RSS. This indicates that the functions chosen and implementation are definitely correct. Secondly, certain segments that are cubic in the advanced and noisy data sets can be seen by eye that it should be linear. Illustrated in graph plot of adv_3.csv below, the first and fourth segment are cubic when it is visibly linear. With this in mind, we could potentially be overfitting for these data sets.



Plot of adv_3.csv

Furthermore, one question we can ask ourselves is does increasing or decreasing the power of the polynomial function improve the results? More specifically, how did we deduce that the cubic function provides us with the most accurate model? To answer this, observe the outcome where we test different values of p on files that has cubic segments:

File Name	RSS (3 d.p.)			
	Quadratic ($p = 2$)	Cubic ($p = 3$)	Quartic ($p = 4$)	Quintic ($p = 5$)
basic_3	15.743	2.923×10^{-18}	1.176×10^{-14}	5.020×10^{-9}
basic_4	7.269×10^{-3}	2.761×10^{-13}	1.522×10^{-4}	8.050
adv_1	218.598	198.232	381.162	379.655
adv_2	3.653	3.651	3.397	3.444
adv_3	986.583	1018.635	91487.034	99497.794
noise_1	11.850	10.985	10.537	9.687
noise_2	809.236	797.917	727.200	821.603
noise_3	482.214	477.699	440.869	504.115

= lowest RSS

Generally, it is clear that cubic performs the best consistently out of all the other types. Take quadratic for example, the only merit for it is in adv_3.csv where it has the lowest RSS across the board but does poorly on every other file, a case of underfitting. Quintic suffers from a similar problem to quadratic, with noise_1.csv bearing the best result and everything else below par. To the extent where the RSS in adv_3 is almost 98 times larger than that of cubic. This is due to the fact that certain segments are using the linear model instead. Finally, for quartic, on the surface, it has the same number of lowest RSS as cubic in the table. Nevertheless, it shares the same undesirable trait with quintic where it uses a linear model for its segments in adv_3.csv, resulting in an almost identical RSS. Notably, we can see a pattern here where higher values of p won't necessarily enhance our results and may lead to overfitting.

4. Future Improvements

As for improvements for future implementations, we could add a check for overfitting by using the type with the lowest RSS only if its value is lower than the next lowest RSS type by an arbitrary percentage.

Moreover, in order to enhance our accuracy, we could regularise our model by using a k-fold cross validation.

5. Conclusion

In conclusion, in terms of generalisation of new data, the 3 chosen function types, linear, cubic and sine, produce the most accurate output. Between changing the power of the polynomial function or switching to a completely different type for the unknown function (to exponential, cosine or tangent), there is no better alternative to the situation as it would only lead to a worse fit. Though we do have to acknowledge the fact that we may be already overfitting in some of the files and there are room for improvements for our model.