Reconstructing Signals via Least Squares Error Modelling

Jonquil Isys R. Mallorca

March 25, 2020

1 Introduction

The Least Squares Method (LSM) is a standard procedure in regression analysis to approximate a solution for a set of coordinates. It involves minimising the residual sum of squares (RSS) of every coordinate to find the best fitting line.

Cross-validation (CV) is a resampling procedure that splits a set of points, trains our model using the training points and checks the cross-validation error (CVE) with the test points. The variation we will use is k-fold cross-validation which repeats the process stated k times and gets the function type with the lowest mean cross-validation error.

We will use LSM and CV with the data given to train our model and observe the results. We expect our program to be able to pick a suitable model, based on the points plotted, with minimal RSS.

2 Explanation of the program

The program reads in multiple segments of points from a file[1]. Through LSM and CV, it will find the function type that appropriately fits the segment, alongside its sum of squares error (SSE). The SSE of each segment is then summed up to form the RSS.

The model has three equations of a line to choose from: linear (degree 1), cubic (degree 3) and sinusoidal (we discuss these selections in Subsection 3.2).

2.1 Linear & Polynomial Regression

Polynomial, and thus linear, regressions can be acquired through the general LSM in matrix form. We input the vector column of y-coordinates y from a file and a matrix X where each row represents the variables of the terms in the equation of a polynomial.

$$y_{(N\times 1)} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \; X_{(N\times(p+1))} = \begin{bmatrix} 1 & x_1^1 & \cdots & x_1^p \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_N^1 & \cdots & x_N^p \end{bmatrix}$$

[1]Note that utilities.py is not used as the functions related to it have been copied over to the main file and modified.

where

$$p = \begin{cases} 1, & \text{if polynomial is linear} \\ 3, & \text{if polynomial is cubic} \end{cases}$$

Using the input, we compute the coefficient vector $A = (X^tX)^{-1}X^ty$, and regression line $\hat{y}$, where $\hat{y}_i = a_0 + a_1x_i + a_2x_i^2 + \cdots + a_px_i^p$ and each a_i are components of A.

2.2 Sinusoidal Regression

Using similar steps in subsection 2.1, We input the y-coordinates y from a file and a matrix X where each row represents the variables of the terms in the equation of a polynomial.

$$y_{(N\times1)} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \quad X_{(N\times2)} = \begin{bmatrix} 1 & sin(x_1) \\ \vdots & \vdots \\ 1 & sin(x_N) \end{bmatrix}$$

Using the input, we compute the coefficient vector $A = (X^tX)^{-1}X^ty$, and regression line $\hat{y}$, where $\hat{y}_i = a_0 + a_1sin(x_i)$ and each a_i are components of A.

3 Results

3.1 Results from chosen regression functions

As demonstrated in Table 1, the model picks agreeable functions for each file segment, even when noise is present. Thus, the model correctly prioritises on selecting the most appropriate function type with relatively low RSS.

Table 1: RSS and sequence of function types for the line segments of each file

File	**RSS**	**Regression functions used**
basic_1	1.69×10^{-28}	Linear
basic_2	6.21×10^{-27}	Linear, Linear
basic_3	1.44×10^{-18}	Cubic
basic_4	1.39×10^{-12}	Linear, Cubic
basic_5	1.05×10^{-25}	Sine
adv_1	199.73	Cubic, Linear, Cubic
adv_2	3.69	Sine, Linear, Sine
adv_3	1019.44	Linear, Cubic, Sine, Linear, Sine, Cubic
noise_1	12.21	Linear
noise_2	849.55	Linear, Cubic
noise_3	482.91	Linear, Cubic, Sine

However, cross-validation cannot completely eradicate overfitting. This is evidently shown on Table 2 and Table 3. It was discovered that the program was frequently alternating between linear and sine functions to model the first segment as the mean CVE would differ depending on how the test set was chosen. We can see in Figure 1 that the first segment is obviously linear.

Table 2: Different RSS results for adv_3

File	RSS		
	(Optimal)	(Worse)	(Worst)
adv_3	1019.44	1017.70	1005.11

Table 3: Different regression functions used for adv_3

File	RSS (Optimal)	(Worse)	(Worst)
adv_3	Linear, Cubic, Sine, Linear, Sine, Cubic	Sine, Cubic, Sine, Linear, Sine, Cubic	Sine, Cubic, Sine, Cubic, Sine, Cubic

3.2 Comparison to other possible function types

The reasons why a cubic polynomial was preferred compared to other potential polynomials are displayed in Table 4. When $p = 2$, most files had either the same or higher RSS than when $p = 3$, indicating underfitting. In comparison, when $p = 4$, certain files had lower RSS than when $p = 2$; however, other files gained a large increase in RSS, most notably adv_3.csv.

Table 4: Comparison of RSS when degree of polynomial $p = 2, 3, 4$

File	RSS ($p = 2$)	RSS ($p = 3$)	RSS ($p = 4$)
basic_3	15.74	1.44×10^{-18}	2.60×10^{-12}
adv_3	1014.41	1019.44	91 521.02

In the last data segment of Figure 2, the program chose to use a linear function type to fit the data as it had the lowest CVE when $p = 4$. It is clear that using $p = 4$ or higher would lead to overfitting.

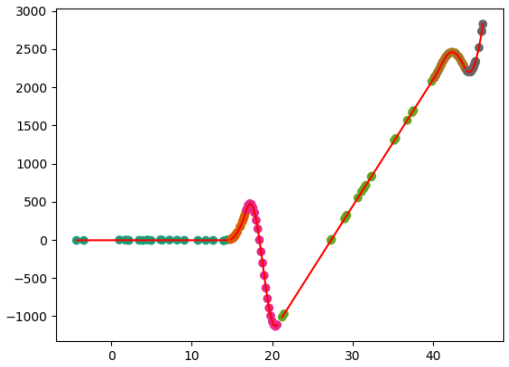

Figure 1: Graph of adv_3.csv when $p = 3$

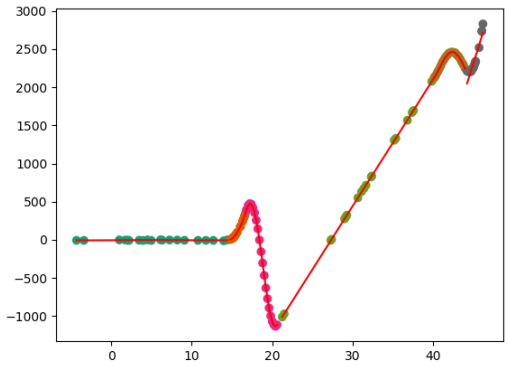

Figure 2: Graph of adv_3.csv when $p = 4$

4 Conclusion

Choosing linear, cubic and sinusoidal regressions, with CV, as the three primary ways of discerning the nature of every segment is sufficient in generalising the data while minimizing overfitting/underfitting, with respect to the training files given.

As a future improvement, we could use regularisation with cross-validation to further the accuracy of our program.