Data Driven Coursework: An Unknown Signal

Pragya Gurung (wq19451)

1. Introduction

The goal is to determine the function types of line segments in files using the Least Squares Method (LSM) and calculating the total residual error. LSM is used to find the coefficient vector that is used to generate the calculated $\hat{y}$ for a line segment. The calculated $\hat{y}$ and the original Y provided in the train files are then used in the Sum of Squares Equation (SSE), which we aim to minimize. For nosier files, we may need to consider that overfitting might be an issue.

2. Method

Obtaining the estimates of the parameters for each function whilst minimizing the SSE values is done by using LSM. The equation goes as follows:

$$X = \begin{bmatrix} 1 & X_1^1 & X_1^2 & \dots & X_1^p \\ 1 & X_2 & X_2 & \dots & X_2 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & X_n & X_n & \dots & X_n \end{bmatrix} \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$A = (X^T * X)^{-1} * X^T * Y = [a', b' \dots]$$

Y is the input coordinates extracted from the .csv files, X is a $p \times n$ matrix (p being the power) and A is a coefficient vector. The coefficients and the X coordinates are used to calculate the regression line of the segment for each function type. The formula for the regression line varies depending on the function type.

For functions that are linear, the regression line is $\hat{y} = a_0 + a_1 x$ and similarly the regression line for the polynomials is $\hat{y} = a_0 + a_1 x + a_2 x^2 + \cdots + a_p x^p$, where was a_i is an element of the coefficient vector A and p is the power.

As for the unknown function, it required a different equation. Determining the last function type (sine) was a trial-and-error process that is detailed further in this report (*3.3*). The new regression line equation for this function is $\hat{y} = a_0 + a_1 \sin(x)$.

After calculating $\hat{y}$ for all three function types, the program then decides which of the three best fits the line segments. It does this by selecting the function that produced the lowest residual error calculated using SSE:

$$S = \sum_i (\hat{y}_i - y_i)^2$$

3. **Analysis**

3.1 The results

Presented below in *Table 1* are the results for the submitted code. The errors for the basic files are very small as the data points are closer to the model function. It suggests that the functions chosen for each segment were correct.

Table 1: SSE values and function type for each file

File (.csv)	SSE (15% threshold)	Function Types
Basic_1	1.688e-28	linear
Basic_2	5.148e-27	linear, linear
Basic_3	1.318e-17	polynomial
Basic_4	4.547e-12	Linear, polynomial
Basic_5	1.050e-25	Sine
Adv_1	199.726	Polynomial, linear, polynomial
Adv_2	3.685	Sine, linear, sine
Adv_3	1008.638	Linear, polynomial, sine, linear, sine, polynomial
Noise_1	12.207	Linear
Noise_2	849.553	Linear, polynomial
Noise_3	482.909	Linear, polynomial, sine

3.2 Handling overfitting

Previously, for noisier files, the program classified some linear segments as polynomial, an indicator for overfitting. For example, noise_2 was being classed as "Polynomial, polynomial" when it is visibly "Linear, polynomial". Figure 1 shows that the first line is wavy, an indicator of overfitting. To resolve this issue, there were two options - imposing a threshold or applying cross-validation.

A threshold was imposed first. With this method, the program classifies a line segment as a polynomial if and only if the percentage difference between the linear and polynomial SSE values is lower than a certain percentage. Through trial and error, the threshold percentage was found to be around 15%.

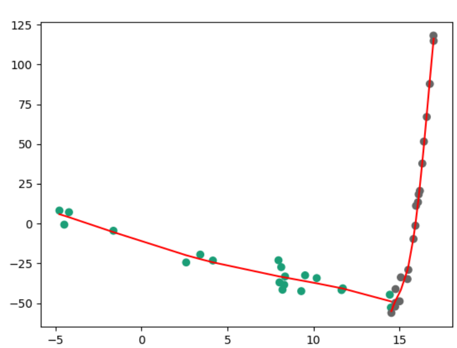

Figure 1: noise_2.csv no threshold

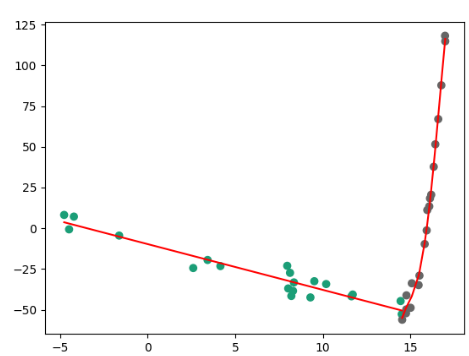

Figure 2: noise_2.csv with threshold 15%

Leave-one-out cross-validation (LOOCV) was attempted next to see if it was better at determining the function type that best fit the line segments. The number of folds is equal to the number of instances in the dataset. It will use one point as the validation set and the rest as training; this is repeated for each instance. LOOCV was appropriate because the dataset was small, and it prioritizes the accuracy of the estimates of model performance over computational costs.

The LOOCV method produced similar results as the threshold method. The program no longer overfitted for most of the files, except for adv_3. Specifically, the first line segment in the file was being classed as sine when it is visibly linear.

Table 2: Results from using cross validation that were different from the previous table

File (.csv)	SSE (with LOOCV)	Function Types
Adv_3	1006.899	**Sine**, polynomial, sine, linear, sine, polynomial

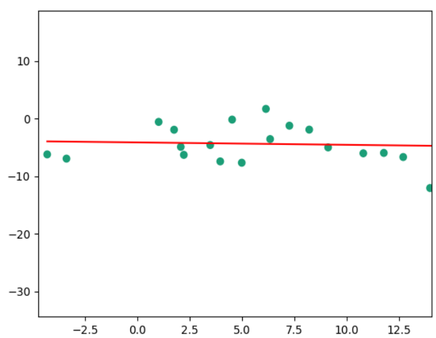

Figure 3: Adv_3 segment 1 with threshold 15%

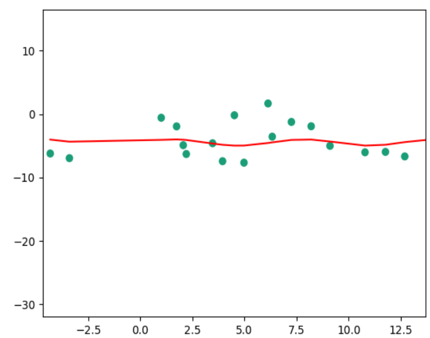

Figure 4: Adv_3 segment 1 with LOOCV

The comparison can be observed in Figures 3 and 4, showing only the first segment of the file. In Figure 5, the line is wavy, once again indicating that the program is overfitting. Cross-validation did not completely overcome the overfitting problem in the model selection - it just reduced it. For this reason, the threshold method was implemented over LOOCV in the final submitted code.

3.3 The unknown function

When determining the unknown function, the file basic_5.csv was very helpful. This is because it contained only 1 segment for which both linear and polynomial produced large errors, indicating neither functions were a correct fit.

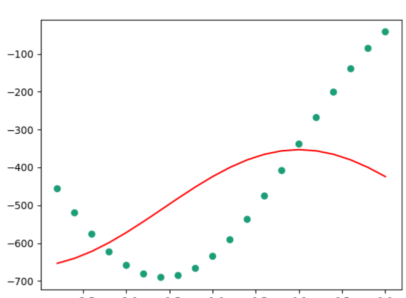

Figure 5: Cos Function

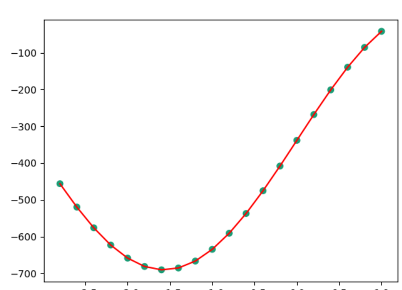

Figure 6: Sine Function

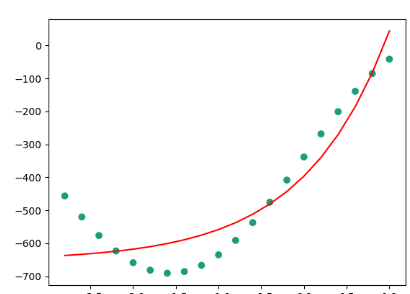

Figure 7: Exponential Function

Through a trial-and-error, sin, cos, and exponential functions were all attempted. Figures 5-7 presents the plotted graphs for each function, and it is apparent that the sine function produced the best line of fit of the data. It also produced smallest error. This was then later confirmed as other files containing the unknown function produced lower residual errors when using sine.

3.4 Order of polynomial

Determining the order of polynomials was also a trial-and-error process. Orders 2-5 were hardcoded into the program and each file that contained polynomial segments were tested. It was found that the program yielded the lowest errors for all files with orders of 3 or 4 (highlighted in green).

Table 3: SSE values when the p (power) is changed

File (.csv)	SSE			
	Quadratic (p=2)	Cubic (p=3)	Quartic (p=4)	Quintic (p=5)
Basic_3	15.743	1.318e-17	1.172e-13	1.0146e-09
Basic_4	7.269e−3	4.547e-12	1.669e-4	16.773
Adv_1	218.855	199.726	199.223	386.216
Adv_2	3.685	3.685	3.477	3.529
Adv_3	1014.409	1008.638	654856.715	655057.065
Noise_2	850.901	849.553	782.254	1267.131
Noise_3	483.063	482.909	446.113	487.593

However, the cubic consistently performed the best overall. Although quartic has the same number of low SSE values, the values for the files adv_3 and basic_4 are significantly larger than that for cubic. Quartic's adv_3 SSE value is almost 649 times larger than Cubic's, and basic_4 is almost 3.67e+12 times larger. For this reason, the chosen order was cubic.

4. **Conclusion**

To conclude, this coursework shows a good way of understanding LSM and SSE to find the best fitting model for the given data. There was no better alternative to the trial-and-error method used to find the order of polynomials as the order is fixed for all files; this is the same for the unknown function. The challenge with noisier files was reducing overfitting, and it's acknowledged that there are multiple ways of overcoming this. In this case, implementing a threshold produced the most accurate results.