Unknown Signal

Elan Virtucio

1 Introduction

It is very useful to be able to model a given set of data points to an appropriate degree of accuracy. This can allow for predictions to be made for an output given some input.

In this instance, a set of data points is given which follows an unknown signal. The task given was to reconstruct this signal and alongside it, calculate the sum squared error or residual sum of squares (RSS) to give an idea of how well the model represents the data. There are different segments of this signal and each one can be modelled by either a linear function, a polynomial function of a given degree or some other function that is unknown.

Although it's possible to model a set of data, ultimately, the results will be highly dependent on the accuracy and correlation of the data and therefore problems may arise such as overfitting. The aims of this project were to minimise these effects and address the limitiations of modelling a data set.

2 Implementation

The program is *lsr.py* and it takes in Comma Separated Value (CSV) files consisiting two columns for the x and y data points respectively. The files can contain one or multiple line segements and each line segments consists of 20 data points. The segments are split up and for each one, a model is fitted and it's RSS is calculated. The RSS of each line segment are summed up to produce the total RSS.

The regression method used was the matrix form of the Least Squares Regression (LSR) and to account for any form of overfitting, the use of a k-fold cross-validation (CV) was used. There are two implementation that can be used, a fixed k-fold or a random k-fold. For example, if $k = 5$, in a fixed k-fold, the parts are split up like so; $[[0,1,2,3],[4,6,7,8],\ldots,[16,17,18,19]]$ where the numbers represent the index of the data point. Whereas, in a random k-fold, the parts can be split up like so; $[[8,3,19,7],[10,2,14,4],\ldots,[15,1,0,8]]$, and this can differ per run.

The program iterates through a list of defined models as described in Table 1 where $\hat{y}$ is the resulting regression function. These models are a list object of their own and contains the properties that'll allow it to model a data set using LSR. These properties are it's name, the relevant function required to extend the X vector with the relevant feature vectors, where X consists of the x data points, and additionally, the equation to use. The data is modelled using each of this models and the RSS is calculated for each one or to be more precise, the CV error, which corresponds to the average RSS value that was calculated during the CV process. The model that produced the minimum error gets chosen as the model to use for that data set.

Table 1: Models

Name	X	$\hat{y}$
Linear	$\begin{bmatrix} 1 & x_0 \\ \vdots & \vdots \\ 1 & x_{N-1} \end{bmatrix}$	$a_0 + a_1 x_i$
2nd to 10th Degree Polynomials	$\begin{bmatrix} 1 & x_0 & \dots & x_0^d \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N-1} & \dots & x_{N-1}^d \end{bmatrix}$	$a_0 + a_1 x_i + \dots + a_d x_i^d$
Exponential	$\begin{bmatrix} 1 & e_0^x \\ \vdots & \vdots \\ 1 & e_{N-1}^x \end{bmatrix}$	$a_0 + a_1 e^{x_i}$
Sinusoidal	$\begin{bmatrix} 1 & sin(x_0) \\ \vdots & \vdots \\ 1 & sin(x_{N-1}) \end{bmatrix}$	$a_0 + a_1 sin(x_i)$
Cosinusoidal	$\begin{bmatrix} 1 & cos(x_0) \\ \vdots & \vdots \\ 1 & cos(x_{N-1}) \end{bmatrix}$	$a_0 + a_1 cos(x_i)$

[1] $N = number\ of\ data\ points\ \therefore\ N = 20.$

[2] $a_j \in A = (X^T X)^{-1} XY \mid Y = \begin{bmatrix} y_0 \\ \vdots \\ y_{N-1} \end{bmatrix}$

[3] $d \in \mathbb{N} \mid 2 \leq d \leq 10$

[4] $i \in \mathbb{Z} \mid 0 \leq i < N$

2.1 Running the Program

lsr.py takes in CSV files as arguments. There are other optional arguments that can be passed: `--plot` shows a visual plot of the fitted line, `-k=10` uses 10 as the value for k in the k-fold CV process, `-v` makes the output more verbose by outputting the model that was used per line segment, `--random-k-fold` uses the random implementation of k-fold CV, `--no-cross-validation` runs the program without using CV.

3 Results

The following results that will be mentioned were obtained using the lab machines. There were multiple different runs of the program accross all of the train data files that were provided. This included changing the k-fold CV methods used including changing the k values as well running it without CV. With these runs, it was surmised that the three function types that were used were linear, cubic and sinusoidal. This was because across all of the basic train files provided, these were the models that the program fitted consistently and additionally, they visually produced acceptable fits for the data. There were other models that were fitted with the advance and noise files, however, considering the specification of the project of having only a linear, a polynomial of a fixed degree, and some additional function in the line segments, these were ruled out as cases of overfitting.

These redundant models were then commented out for future runs of the program and therefore were not part of the iteration of the list of models to fit the data in. An instance of this is shown in Table 3

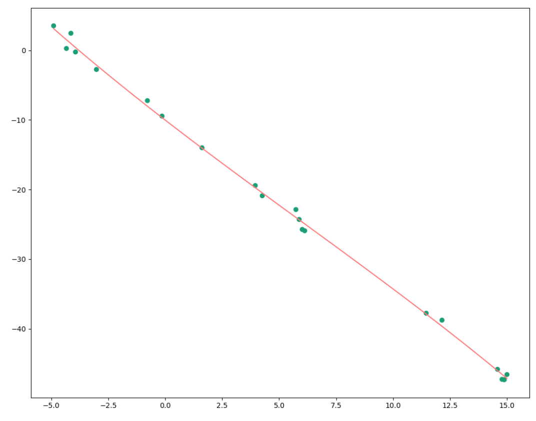
Table 2: Results of training data files using a fixed 10-fold CV

File (.csv)	RSS	Fitted Models *w.r.t.* the Line Segments
basic_1	2.229×10^{-27}	Linear
basic_2	2.055×10^{-27}	Linear, Linear
basic_3	3.805×10^{-18}	Cubic
basic_4	8.645×10^{-11}	Linear, Cubic
basic_5	2.157×10^{-25}	Sinusoidal
adv_1	220.4	Sinusoidal, Linear, Cubic
adv_2	3.685	Sinusoidal, Linear, Sine
adv_3	1019	Sinusoidal, Cubic, Sinusoidal, Linear, Sinusoidal, Cubic
noise_1	12.21	Linear
noise_2	849.6	Linear, Cubic
noise_3	482.9	Linear, Cubic, Sinusoidal

Table 3: Results of noise training data files without CV

File (.csv)	RSS	Fitted Models *w.r.t.* the Line Segments
noise_1	10.99	Cubic
noise_2	797.9	Cubic, Cubic
noise_3	477.7	Cubic, Cubic, Sinusoidal

Figure 1: Plot of fitted line of *noise_1.csv* without CV



4 Conclusion