Signals, Patterns, and Symbols Report - The Unknown Signal

Jordan Taylor

1. Introduction and Background

This assignment involved taking various concepts of statistics and data science for use in a data modelling scenario. The project's aim was to read a .csv file containing a variable size set of data points, separate it into chunks, then determine the best fitting function to describe the plot – with the objective of minimising the error between the data and the calculated model.

There was a further challenge; to determine an unknown signal. ***Failure could result in nuclear war…***

The project utilises the Anaconda virtual environment with Python 3.7.4 64-bit. This enables several useful packages to be installed including pyplot, numPy, and pandas.

This assignment makes use of the matrix form of least squares regression. It enables code to be re-useable and concise. It may be easily extended to support multiple dimensions of analysis. This is an abstraction of residual minimisation. Partial derivatives with respect to each regression parameter are taken and set to zero.

In a generalised case, a set of polynomial regression coefficients are generated in a 2-D array a_{LS} where X is the design matrix, an N-D array containing values of x from order 0 to order n.

$$\mathrm{a}_{LS} = \left(\mathrm{X}^{\mathrm{T}}\mathrm{X}\right)^{-1}\mathrm{X}^{\mathrm{T}}\,\mathrm{y}$$

2. Functionality and Design

2.1. Final Solution Overview:

The final version of the project attempts to meet all the specified success criteria. It may test the suitability of a set's regression up to any n degree polynomial without overfitting. A variation of k-fold validation is used to find an average case square error for each model. Additionally, the solution successfully plots the unknown signal where appropriate.

2.2. Initial Fixed Implementation:

The project's scope increased over time as successive versions of the source code were created. The initial implementation focused on setting up a prototype program to model regressions up to n^3.

The implementation first involved reading an input file, separating the x and y points into 20 length subsets if necessary, and iteratively processing each subset of data. Rudimentary error checking was used to choose the closest fit function. The best estimates were output to the console.

2.3. Unbounded Polynomial Implementation:

At this point, the program was only able to determine if a line segment appeared to be linear, quadratic, or cubic (and with a limited degree of accuracy). I realised that I would need to start broadening the script's scope to handle more complex functions and begin to determine the unknown signal. This involved changing the body and parameters of several functions to accept a high degree input.

For instance, the least squares coefficient function was changed from checking a specific line type to iterating through to the specified order, expanding the design matrix using the column stack operation.

2.3. Training Sets and K-Fold Validation:

At this point, regression functions were only calculated once. All points were compared to the line segment. This produced problems with overfitting. Line segments would almost always be modelled using a high degree polynomial.

To rectify this, the concept of cross validation was introduced. Data was split into training and test sets. The least squares regression of the training data was calculated and the square error derived from the difference between the modelled line and test set. Removing several points at a time helped mitigate the issue of function overfitting.

However, this still yielded a large deviation in the predicted order of the line segment due to reducing the size of the original dataset.

Figure2: The output of the best fit polynomial degree for each line section in adv_1.csv. The prediction changes over trials.

```
PS D:\SPS Coursework> python cwMk4_withErrorOutput.py "train_data/adv_1.csv"
3
4
3
194.64600169477552
PS D:\SPS Coursework> python cwMk4_withErrorOutput.py "train_data/adv_1.csv"
3
3
3
198.23215982586981
```

Not only were the results imprecise, but they were wildly inaccurate compared to the apparent order of the line segments.

The process was repeated multiple times for testing each order of polynomial – essentially a naïve approach to generalisation. Each time a random subset of the data was chosen for training. This was computed by shuffling the list of x and y datapoints with the same random seed, then taking the first n elements. Accuracy of fit appeared to improve with the number of trials.

2.4. Sin – The Unknown Signal:

Using the functionality provided by the state of my program after implementing a robust validation system, I set about experimentally determining the unknown signal (see section 3 for more detail). This was discovered to be sin. The least squares and regression function in the code were expanded to include a case for generating a sin-based plot.

3. Tests, Experiments & Critical Analysis

3.1. Discovering the Unknown Signal:

Before implementing the least squares sin regression, all basic datasets except basic_5 output a sum error of close to 0 (w.r.t floating point precision). I deduced that basic_5 contained the unknown signal.

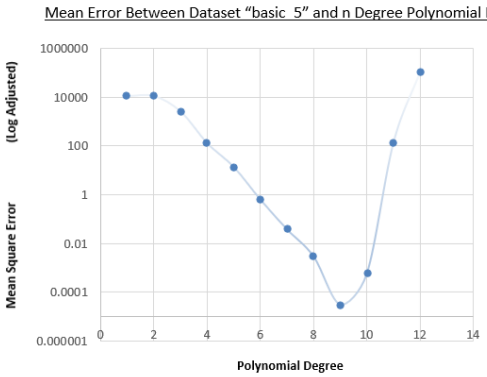
Figure3: The output sum squared error of each basic dataset.

```
PS D:\SPS Coursework> python cwMk4_withErrorOutput.py "train_data/basic_1.csv"
1.681062589025976e-27
PS D:\SPS Coursework> python cwMk4_withErrorOutput.py "train_data/basic_2.csv"
6.467081701001855e-27
PS D:\SPS Coursework> python cwMk4_withErrorOutput.py "train_data/basic_3.csv"
2.9225169587180204e-18
PS D:\SPS Coursework> python cwMk4_withErrorOutput.py "train_data/basic_4.csv"
2.761250335560573e-13
PS D:\SPS Coursework> python cwMk4_withErrorOutput.py "train_data/basic_5.csv"
2612.9757784503645
```

I decided to run further tests on this dataset. The program's maximum regression polynomial was set to n^{12}, gathering a large list of error data. The mean of each error was plotted against its respective order*(Figure 4).*

It can be seen that up to n^9 , the error (and hence the accurracy of the plot) improves with the polynomial degree. I attributed the subsequent rise in error magnitute (n^{10} and beyond) to heavy overfittting miscalculations and ignored this part of the plot.Basic_5 was ruled out being exponential due to how its result fluctuates– behaviour not typical of an exponential function.

Figure4: Mean error between dataset "Basic_5" and n degree polynomial fit.



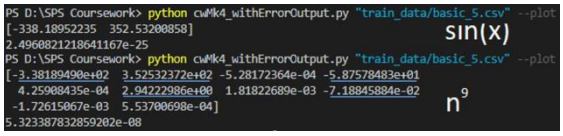
Further study of basic_5's plot shows hints that it may be an oscillating pattern (ruling out it being exponential). Reading the x axis, it appeared that a quarter of the potential pattern was shown over an interval of ~1.5; which is approximately ($\pi/2$). This is the behaviour of a trigonometric function. I proceeded to expand the matrix least squares to

plot sinusoidal regressions. The script now yielded an error of ~0. Success!

3.2. Confidence in the Unknown Signal and Determining the Polynomial:

Recalling the Taylor Expansion, real functions may be expressed as asummation of an infinite series of polynomials. I altered the script to output the coefficients of the line of regression for the sinusoidal and n^9 curves.

Figure5:The coefficients of the least squares regression models for basic_5. Similar Taylor coefficients are underlined.



Having obtained the equation of the sin function, it was possible to calculate a numerical estimate using the Taylor series:

Figure6:Taylor Expansion of $f(x) = 352sin(x) - 338$:

$$-338.2 + 352.53\mathrm{x} - 58.76\mathrm{x}^3 + 2.94x^5 - 0.070x^7 + \cdots$$

The expansion coefficients closely mirror that of the n^9 estimate. Therefore, I conclude with greater certainty that the unknown signal is sin(x).

Missile flight paths may now be extrapolated. Recommend deployment of anti-missile countermeasures.

The assignment sheet stated that all non-linear polynomial plots were to be of the same degree. Now, with confidence in sin, it was possible to determine this order. I ran the program on each of the basic graphs up to n^5 and set it to output the best fit for the least residual error. The only value printed was n^3. Therefore, I conclude that n^3 is the polynomial.

3.3. Potential Improvements:

I believe that my solution is logically correct with respect to modelling datasets; however, the source code is inefficient in places. The high number of cross-validation repetitions means that the program has a relatively long runtime. This could be improved by finding an effective trade-off between the number of trials and model accuracy. Furthermore, the validation algorithm could be altered to calculate the sum of a fixed number of point-wise operations, rather than a larger amount of set-wise procedures.

Presently, the code contains hardcoded cases for calculating sinusoidal and polynomial regression coefficients, with sin treated as an order exception. The script could be generalised to take lambda function as a parameter, thus reducing repetition and allowing any type of function to be modelled.

Currently the solution has no robust measures of regularisation (and because no noisy line segments yielded zero loss with high test set error), it was deemed that the given datasets did not require such a technique. However, the program could be adapted to model larger and more erratic datasets. One method that could be implemented is *"Tikhonov-regularized least squares"*, in which a gradient loss function would be solved to find an optimal weighting.

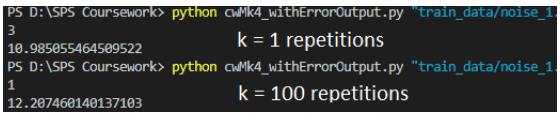
3.4. Conclusion:

From this assignment I have experienced how easy it is to misrepresent a dataset by using an overfitting model. The usage of training, validation, and test sets in calculating a trend predictor helped mitigate this issue.

An interesting find was high degree polynomial models would go beyond merely overfitting data and drastically miscalculate the data's behaviour, resulting in a high sum square error *(as demonstrated in Figure 6).*

Another interesting find was how, when computed with few cross-reference trials, seemingly linear relationships with an added factor of noise would have erroneously labelled models *(Figure 8).*

Figure7: Erroneous model orders for noise_1. The function should be linear.



To conclude, I believe that this assignment has been a success. To my knowledge, all datasets are modelled as accurately as possible and with consistency.

Due to the ingenuity of SPS, the Earth has been saved.