Symbols, Patterns and Signals

CW1: An Unknown Signal

Stefan Valentin Popescu (bk18529)

March 27, 2020

1. Least Squares Method

The "least squares" method is a form of mathematical regression analysis used to determine the line of best fit for a set of data, providing a visual demonstration of the relationship between the data points. The most common application of this method, which is sometimes referred to as "linear", aims to create a straight line that minimizes the sum of the squares of the errors that are generated by the results of the associated equations, such as the squared residuals resulting from differences in the observed value, and the value anticipated, based on that model.

1.1. Linear Least Squares

The goal is to minimize the residual $R(a,b) = \sum_{i=1}^{N}(\mathrm{y_i} - (a + bx_i))^2$. The parameters **a** and **b** which minimize *R* can be found by taking the partial derivatives w.r.t. (a,b) and setting them to zero. The results are: $a_{LS} = \bar{\mathrm{y}} - b_{LS}\bar{\mathrm{x}}$ and $b_{LS} = \frac{\sum_i \mathrm{x_i y_i} - N\bar{x}\bar{y}}{\sum_i x_i^2 - N\bar{x}^2}$ where $\bar{\mathrm{x}}$, $\bar{\mathrm{y}}$ represent the mean.

1.2. Matrix form

This coursework uses the matrix form solution for Least Squares. A column vector Y keeps the y coordinates, X is a *Nx2* matrix with the first column of ones and the second containing x coordinates and A is the column vector containing the parameters. The residual *R* takes this form now: $|\mathrm{Y\text{-}XA}|^2$. To minimize the vector we solve the equation $|\mathrm{Y\text{-}XA}|^2 = 0$ and obtain $\mathrm{A} = (X^T.X)^{-1}.X^T.Y$. The matrix form allows Least Squares method to be easily extended to polynomial fitting.

1.3. Non-linear Least Squares

When extending to a non-linear system the form of X changes. For the fundamental functions (sin, cos, log, exp etc.) the only change happens to the second column of X where the desired function is applied on each x coordinate. For a higher degree polynomial the shape of X changes. X becomes a Vandermonde matrix where the first column is one, the second represents x coordinates, the third represents x coordinates squared and so on. The shape of A will also change. The number of parameters will vary depending on the degree of the polynomial.

$$X = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_{n-1} \end{bmatrix}$$

As the parameter lies inside f(x) we can't modify the matrix X to include this parameter. Therefore composed functions (eg. $\mathrm{f}(e^x)$) can't be represented using Least Squares, only functions of the form af(x) + bg(x) + ch(x)+… can be modeled.

2. Analysis of results

File name	Maximum degree 2	Maximum degree 3	Maximum degree 4	Maximum degree 2 + threshold	Maximum degree 3 + threshold	Maximum degree 4 + threshold
Basic_1	1.68 e-28	1.68 e-28	1.68 e-28	1.68 e-28	1.68 e-28	1.68 e-28
Basic_2	6.21 e-27	6.21 e-27	6.21 e-27	6.21 e-27	6.21 e-27	6.21 e-27
Basic_3	15.74	1.43 e-18	1.43 e-18	15.74	1.43 e-18	1.43 e-18
Basic_4	0.0072	1.38 e-12	1.38 e-12	0.0072	1.38 e-12	1.38 e-12
Basic_5	1.05 e-25	1.05 e-25	1.05 e-25	1.05 e-25	1.05 e-25	1.05 e-25
Noise_1	11.84	10.98	10.53	12.20	12.20	12.20
Noise_2	809.23	797.91	727.20	850.90	849.55	782.25
Noise_3	482.21	477.69	440.86	483.06	482.90	446.11
Adv_1	218.59	198.23	194.50	218.85	199.72	199.58
Adv_2	3.65	3.65	3.39	26.15	3.68	3.47
Adv_3	986.05	974.50	948.02	357636	1019.43	91521

Table 1: Error results on different polynomial degrees

2.1. Implementation

The program consists of two functions, one dealing with polynomials (including linear) and one dealing with the unknown function. The generic polynomial function is given a bound for the polynomial degree. The squared error is calculated on each segment for each polynomial degree, starting with one(linear). The minimum error is selected for each segment. This error is then compared with the error given by the unknown function applied on that segment. The smallest of the two is added to the total error.

2.2. Explaining the results

The results for the submitted version are represented in the highlighted column of *Table 1*. The table can be interpreted as the left side and the right side. The first three columns represent the results with no threshold added. The program chooses the smallest error every time, this explains the outcomes slightly smaller in this part. On the other half the program is conditioned to choose linear over a higher degree polynomial in some cases, to avoid overfitting.

The errors of *basic* files stay constant while the others vary more. The data points in the *basic* files are closer to a specific model function (linear or cubic). This explains why the errors are constant and small. On the left side of the table, the errors of files *noise* and *adv* decrease as the degree of the polynomial increases. By increasing the degree, the line gets closer to the center of each data point and the squared error gets smaller and smaller. This phenomenon is referred to as overfitting. The same thing happens in the right side of the table, except for file *adv_3* (Figure 1) where the best fit is obtained by setting the polynomial degree bound to three, meaning that certain segments follow the cubic function.

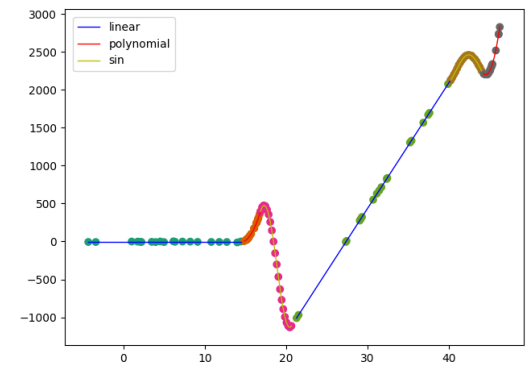

Figure 1

3. Generalization and overfitting

3.1. The threshold

In order to avoid overfitting for some segments which follow a linear model but are chosen as a higher degree polynomial by the program, a threshold is imposed. After some testing, the threshold was set to 1.2*polynomial error. Every time the linear error is smaller than this threshold, the linear is selected for that segment. The result can be observed on the first segment in Figure 2 and Figure 3. In the first plot, the line is wavy, an indicator of overfitting, while in the second plot, a straight line is chosen as a best fit. While overfitting is avoided, the error increases.

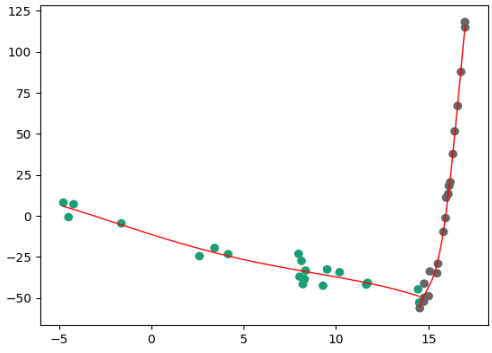

Figure 2: Result on *noise_2* with no threshold

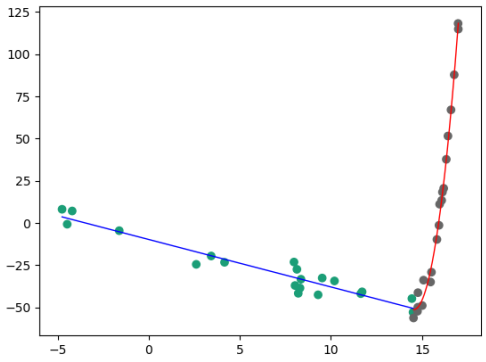

Figure 3: Result on *noise_2* with threshold

3.2. Choosing the unknown function

When choosing the unknown function, the *basic_5* file was very useful. This file gives very large errors for linear and polynomial functions, those representing an underfit. Therefore, it must follow a different type of function. After verifying the elementary functions (*exp*, *log*, *sin*, *cos*), it has been observed that the ***sin*** function gave the smallest error and therefore was the best fit. The program selects this function, this is why the error in the *Basic 5* row in *Table 1* stays constant.

3.3. Possible extensions

One possible extension can be the implementation of Weighted Least Squares. This method attaches non-negative constants to data points. It is used when the data violates the assumption of homoscedasticity, when there are certain areas of interest or when the data points should not be treated equally.

Another possible extension could be the prevention of overfitting with cross-validation. The idea is to use the initial training data to generate multiple mini train-test splits and then use these splits to tune the model. The standard k-fold cross-validation partitions the data into k subsets called folds, then iteratively trains the algorithm on k-1 folds while using the remaining fold as the test set. This allows keeping the test set as a truly unseen dataset for selecting the final model.

4. Conclusion

This coursework represented a good way of understanding the Least Squares regression and the idea of finding the “best” fitting model for a given train data set. The notions of generalization, overfit and underfit became more clear as well. The challenges of finding the unknown function and the maximum degree of the polynomial function to avoid overfitting were also widening my understanding of the topic.