Least Squares Regression Report: An Unknown Signal

Taharka Okai

April 2021

1 Linear regression

Consider the case where a pair (x, y) related by f where $y = f(x)$ can be estimated by a 'guess' function $\hat{y} = \hat{f}(x)$. The likelihood of the output given the estimate, assuming that the likelihood is described by a normal distribution, yields the following:

$$P\left(y|\hat{y}\right) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{y-\hat{y}}{\sigma}\right)^2\right]$$

However nothing is known about the value of $\hat{y}$ as the function describing the estimate is still unknown. It is possible to "design" a function by expressing $\hat{f}(x)$ as a matrix product between a vector containing functions of x, or features and different scaling factors for each feature, or weights. Below is an example:

$$\text{Take } \hat{f}(x) = 2x + 3. \text{ Then } \hat{f}(x) = \boldsymbol{x}^T\boldsymbol{w} \text{ where } \boldsymbol{x} = \begin{bmatrix}1 & x\end{bmatrix} \text{ and } \boldsymbol{w} = \begin{bmatrix}2\\3\end{bmatrix}$$

Note that the above can be extended where the input space is a vector. Using this structure, the likelihood can be rewritten in terms of the feature and weight vectors. The above only considers a case where a single estimator for y is considered, but when working with vectors, the product of all likelihoods for each dimension in y represents the overall likelihood given $\hat{\boldsymbol{y}}$.

Using log values of the likelihood not only makes the calculation of probabilities more stable during computation - as the probabilities tend to be small - but also reduces the product of the likelihoods in the equation to a sum (because $\log\prod_i x_i = \sum_i \log x_i$), making the differentiation step easier to calculate.

$$P\left(\boldsymbol{y}_i|\boldsymbol{x}_i, \boldsymbol{w}_i\right) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{\boldsymbol{y}_i - {\boldsymbol{x}_i}^T\boldsymbol{w}_i}{\sigma}\right)^2\right]$$

$$\mathcal{L}(\boldsymbol{w}) = \log\left(\prod_i P\left(\boldsymbol{y}_i|\boldsymbol{x}_i\right), \boldsymbol{w}_i\right) = -\frac{1}{2}\sum_i\left(\log 2\pi\sigma^2 + \frac{1}{\sigma^2}\left(\boldsymbol{y}_i - \sum_j \mathbf{X}_{ij}\boldsymbol{w}_j\right)^2\right)$$

Using the equation above it is possible to maximise the likelihood using the single free parameter in this equation, $\boldsymbol{w}$. Finding the solution to $\frac{\partial\mathcal{L}(\boldsymbol{w})}{\partial\boldsymbol{w}} = \mathbf{0}$ produces $\hat{\boldsymbol{w}}$, the value of the weight vector that maximises $P(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})$:

$$\frac{\partial\mathcal{L}\left(\boldsymbol{w}_k\right)}{\partial\boldsymbol{w}_k} = -\frac{2}{2\sigma^2}\sum_i\left(\left(\frac{\partial\boldsymbol{y}_i}{\partial\boldsymbol{w}_k} + \sum_j \frac{\partial\boldsymbol{w}_j}{\partial\boldsymbol{w}_k}\mathbf{X}_{ij}\right)\cdot\left(y_i - \sum_j \mathbf{X}_{ij}\boldsymbol{w}_j\right)\right)$$

$$0 = -\frac{1}{\sigma^2}\sum_i\left(0 + \mathbf{X}_{ik}\right)\cdot\left(y_i - \sum_j \mathbf{X}_{ij}\boldsymbol{w}_j\right) = \sum_i \mathbf{X}_{ik}\left(y_i - \sum_j \mathbf{X}_{ij}\boldsymbol{w}_j\right)$$

Since $\boldsymbol{w}_j$ is independent of $\boldsymbol{w}_k$ when $j{=}k, \frac{\partial\boldsymbol{w}_j}{\partial\boldsymbol{w}_k}{=}1$, and 0 otherwise

Thus remains:

$$\mathbf{X}^T\mathbf{X}\boldsymbol{y} - \mathbf{X}^T\mathbf{X}\hat{\boldsymbol{w}} = \mathbf{0}$$
$$\hat{\boldsymbol{w}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\boldsymbol{y}$$

To finally calculate the value of $\hat{y}$ and estimate y, simply substitute the maximising weight vector into the equation for $\hat{f}(x)$.

$$\boldsymbol{y} \approx \hat{\boldsymbol{y}} = \boldsymbol{x}^T\hat{w}$$

2 Method

2.1 Nomenclature

The features describing the general characteristic of a function's properties, like the shape of its graph, will be referred to as the 'class' of the function.

The function on x with the highest coefficient or exponent of fitted function's class will be referred to as the 'order', where $\sin(3x)$ is of greater order than $\sin(x)$. This comparison does not apply between function classes, however.

2.2 Observations

Firstly the files provided were parsed to retrieve data and assigned to input and output vectors, extended as described in the previous section. Then functions were chosen that would supposedly fit into the three given types, linear, polynomial or unknown.

Function	Form	$\hat{\boldsymbol{w}}$
Linear	$a + bx$	$\begin{bmatrix} a & b \end{bmatrix}^T$
Polynomial	$a + bx + cx^2 + dx^3 + \cdots$	$\begin{bmatrix} a & b & c & d & \cdots \end{bmatrix}^T$
Exponential	$a + e^{bx}$	$\begin{bmatrix} a & b \end{bmatrix}^T$
Sinusoidal	$a + \sin(bx) + \cos(bx) + \sin(cx) + \cos(cx) + \cdots$	$\begin{bmatrix} a & b & c & \cdots \end{bmatrix}^T$

Table 1: Functions to fit for $\hat{\boldsymbol{w}}; \hat{\boldsymbol{y}}$

A large set of Python lists were generated containing function objects with the specified parameters, for example calling `Model.polynomial(3)` would result in a feature vector matching the polynomial of order 3. A 'training set' would be a hard coded subset of the available functions mentioned in Table 1, for example, { linear, polynomial(2), polynomial(3), exponential, sinusoidal(2) }. Additionally, upon controlled test runs, the maximum order for the sinusoidal and polynomial functions were reduced to 3 and 5 respectively, as those models were seldom chosen as suitable models for the provided dataset.

An observation that the exponential model was not chosen once by the algorithm indicates one of three possible things - either the exponential model was a poor choice for fitting these specific functions, the model was not extensive enough (i.e. not enough terms) or the model was not described correctly (i.e. $a + e^{bx}$ is the wrong form.)

2.3 Determining function class

Running these tests resulted in the plots shown in Figure 3. The algorithm progressively finds the model with the least cross validation error for each segment, meaning that the model with the least residuals gets chosen. This aims to produce accurate looking plots. However this does not yet account for possible overfitting, as part of the assignment is to ensure that the order for each class of function is consistent between files.

2.4 Determining function order

The strategy for determining the common unknown and polynomial classes over each file was to be that for each segment with a specific set of class (i.e. polynomial, sinusoidal), the program would determine whether the order was disparate between files. For example if one segment was fitted to a quadratic and another to a cubic - the same class but different order. In this specific cases, the disparate orders would be collected in a list and the algorithm run again for these segments, except instead of minimising error across function classes, the program would attempt to minimise error over order, to choose the best order that minimises error over all files considered. The results of

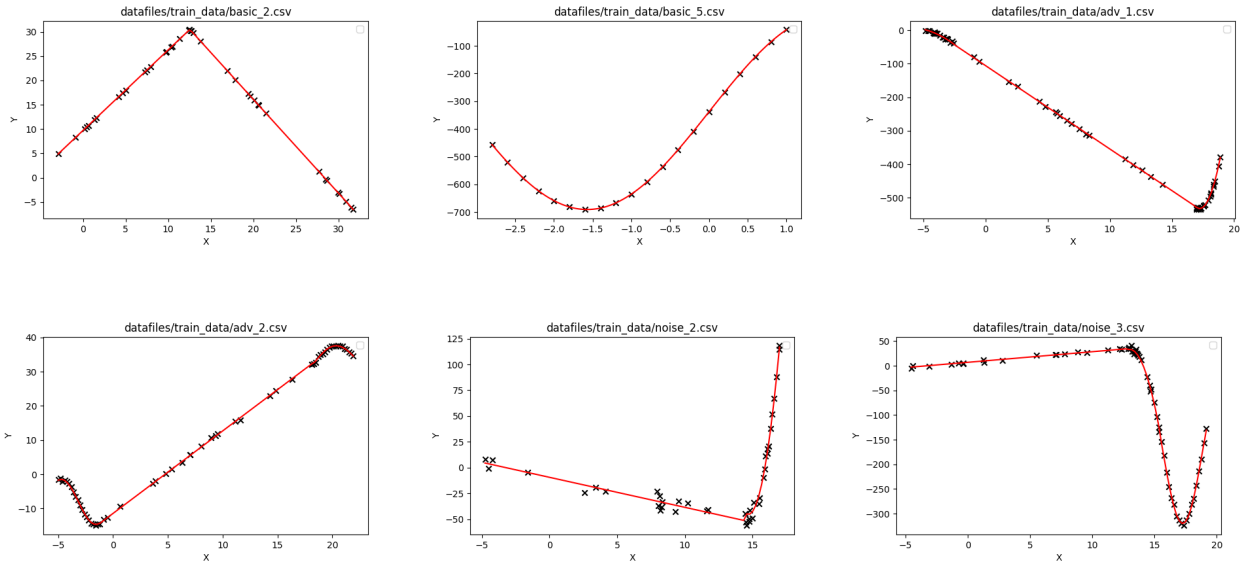

Figure 1: Some unaccounted-for fitted model results (no class determination)

this are shown in Table 2. In this instance, the functions chosen would be the order 4 polynomial and the order 2 sinusoid.

Function class	Sum-squared Error
Polynomial order 3	3514.69
Polynomial order 4	3283.77
Sinusoid 'order' 1	1199.00
Sinusoid 'order' 2	1124.00

Table 2: The Sum-squared error over segments of multiple order classes

3 Shortcomings

A possible caveat in the algorithms cross validation process is that a systematic approach to the selection of 50% of the points, a number chosen arbitrarily, was carried out. This may have steepened the correlation of the function between the points in the training set and the points in the validation set. The attempt to counterbalance this was to alternate between the cross-validation and training sets and select the function with the least cross validation error.

While this did create stability in the output (because the set of points removed from the training set per iteration was the same), this is a non-standard technique and may cause some biases in the fitting. For example, the fact that the input space is not uniformly distributed (i.e. the difference between adjacent points on the x-axis can vary wildly). Perhaps too many points were removed from the validation set for the results to be close to the true answer.

Observable oversights of the approach to determining the order of the function include the lack of evidence supporting the claim that this is a suitable way to mitigate overfitting. In hindsight, a reduced sum squared error at a higher order may be indicative of overfitting, so this method is by no means entirely reliable. Due to programming bugs and time restraints, graphs for the selected order for the functions were unfortunately not included in this document.