Data Driven Computer Science
Unknown Signal Coursework

George Edward Nechitoaia / dp19681

Challenges

The main challenges are to understand and analyse the data, to determine the polynomial order and the unknown function while finding a way to avoid overfitting and to build a program that computes the total reconstruction error of a file.

Fitting Method

The fitting method used during all analysis is maximum-likelihood /least squares regression:

$$W = (X^T \, . \, X)^{-1} . \, X^T \, . \, Y$$

W = weights matrix

X = independent train data matrix

Y = dependent train data matrix

X and W have different forms, depending on the fitting function. For example:

1. Liniar:

$$\widehat{y}_i = b * x_i + a$$
$$W = [a, b]$$
$$X = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ ... & ... \\ 1 & x_n \end{pmatrix}$$

2. Polynomial of 3rd order:

$$\widehat{y}_i = d * x_i^3 + c * x_i^2 + b * x_i + a$$
$$W = [a, b, c, d]$$
$$X = \begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ ... & ... & ... & \\ ... & & & \\ 1 & x_n & x_n^2 & x_n^3 \end{pmatrix}$$

3. Sinusoidal function:

$$\widehat{y}_i = b * sin(x_i) + a$$
$$W = [a, b]$$
$$X = \begin{pmatrix} 1 & sin(x_1) \\ 1 & sin(x_2) \\ 1 & sin(x_3) \\ ... & ... \\ 1 & sin(x_n) \end{pmatrix}$$

$$\widehat{Y} = [\widehat{y}_1, \widehat{y}_2, \widehat{y}_3 \dots \widehat{y}_n]$$

$$\widehat{Y} = dependent\ variable\ prediction\ matrix$$

Error method

The Sum Squared Error is computed in order to obtain the accuracy fitting of each function.

$$SEE = \Sigma_i \, (y_i - \widehat{y}_i)^2$$

y = real value

$\widehat{y}$ = predicted value

Determining the functions type

In order to find the functions type (polynomial order and unknown) , I use the following strategy to both understand the data and to analyse the errors and fitting trends:

1. I fit each segment of the files to multiple functions types.
   (liniar; 2,3,4,5,6-order polynomial; sin; cos; exponential)
2. In order to treat overfitting, I use "Leave-one-out cross validation". The motivation behind my choice is the limited size of the dataset. This cross validation method helps build 20 models, by taking for each model a different datapoint from within a segment as testset, avoiding any type of bias. The 20 errors are used to compute the respective function mean error.
3. Finally, it compares all the functions' mean error and proceeds further analysis. I build a table taking into account :
   a. How many times a function type is preferred (the one with the smallest function mean error)
   b. I compute the cost row, by following the rules:
      i. If the segment prefers linear, it will add only the linear mean error to the linear cost error.
      ii. If the segment prefers polynomial (of any order), it will add every polynomial function mean error to the respective polynomial cost error.
      iii. If the segment prefers unknown, it will add only every unknown function mean error by adding it to the respective unknown cost error.

I apply the strategy 2 times: firstly on all files (noise + noise-free data) and secondly only on noise-free files.

Table 1: The results for all files:

	liniar	2nd order	3rd order	4th order	5th order	6th order	sin	cos	exp
Best fit	10	5	3	0	0	0	7	1	0
Cost	43	86	96	678,448	1,147,819	12,684,691	104	972,831	277,388

Table 2: The results for only noise-free files ("basic") :

	liniar	2nd order	3rd order	4th order	5th order	6th order	sin	cos	exp
Best fit	4	0	2	0	0	0	1	0	0
Cost	2e-27	1.3	8e-13	2	19	1094	9e-27	39,348	7,303

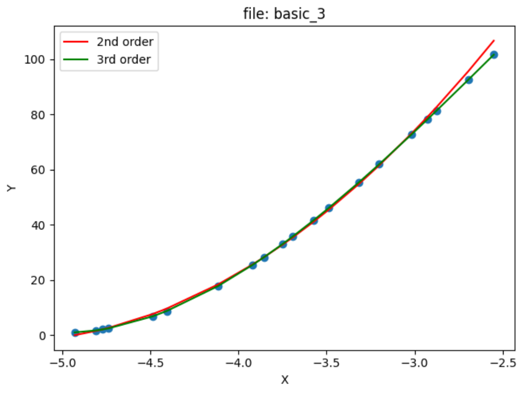
Finding the polynomial order

As it can be seen in tables, the Leave-one-out-cross validation method worked perfectly, there is no overfitting. This way we can exclude the 4th, 5th and 6th order immediately.

We need to choose between 2nd and 3rd order. We know the 2nd and 3rd order errors are taken into consideration only when the preferred function is a polynomial, but the only preferred polynomial functions are the 2nd and 3rd order. This gives us a good starting point in analysing the data. Despite the fact that Table 1 shows 2nd order appearing more times as the "Best Fit" than 3rd order, both of them have similar total errors - 86 / 96. This means that everytime a segment fits a polynomial, the errors between these 2 orders are quite close. (mean difference = (96 - 86) / (5+3) = 1.25 per segment). This previous information was gathered by analysing all data (noise-free files + noise files).

Moreover, Table 2 shows that when it comes to the noise-free files, the 3rd order is surprisingly better, with quite a big difference between errors and a 2 to 0 preferred segments.

Further, I plot and compute the errors for the "basic_3" file. Visually, it seems that this file follows a polynomial function.



I use hold out cross validation with 17 points for train and 3 points for the test, both for treating overfitting and for visual purposes. The errors are:

2nd order: 37

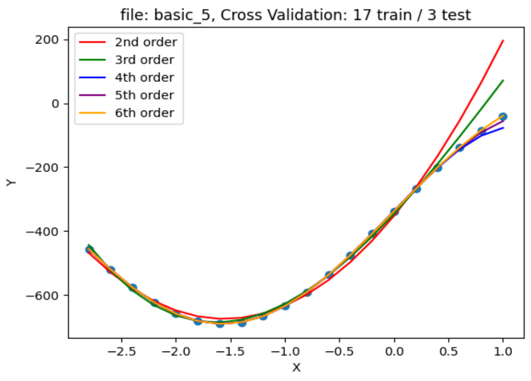
3rd order: $2 * 10^{-17}$

From both the plot and the errors, it is clear that the 3rd order is more accurate while generalising new data.

Taking all these in account, I strongly motivate that the right choice for the polynomial order is the 3rd order.

Finding the unknown function

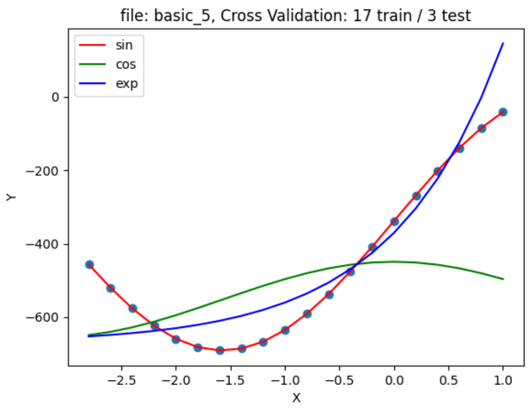
I choose to analyse 3 "unknown" functions: sin, cos and exponential. I motivate this choice by looking at the "basic_5" file, which is noise-free. This file does not follow neither a linear function, nor any polynomial function (as the plot on the right shows, 5th and 6th order are excluded).



So now we know that this file describes the unknown function. My assumption is that the line follows a sinusoidal line.

I used hold out cross validation with 17 train points and 3 test points.

It is visually clear from the plot on the right that sin is the best fit.



We look in the Tables from above and analyse the obtained data. Both Table 1 and Table 2 shows that the majority prefers sin. The difference between errors is enormous with sin being the winner in both tables.

Taking these into account, I strongly believe that sin is the right choice for the unknown function.

Final fitting and reconstruction error

I choose to go with: linear, 3rd order and sinusoidal functions. The final challange is to build a program that chooses which function is the best fit for each segment. This way the “Total Reconstruction Error" of the input file will be computed.

The approached strategy is straight-forward, the program :

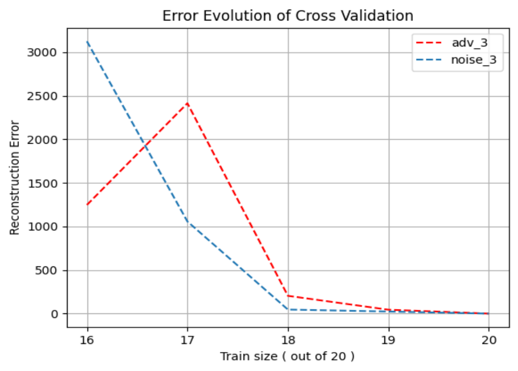
1. Fits each segment with all 3 functions (subject to cross validation, to avoid overfitting).
2. Calculates the test error for each function.
3. Chooses the function with the smallest test error.
4. Computes the reconstruction error of that segment, fitting the chosen function.
5. Adds up all segments’ reconstruction errors in order to obtain the total reconstruction error
6. Optional: Plots a figure showing the reconstructed line

When choosing the validation method, I analyse:

Hold out:

I plot the total reconstruction error evolution for 2 complex files (adv_3 and noise_3) , against different train set sizes.

It is clear that a train set smaller than 17 is inclined to overfitting. This happens mostly from the limited number of points in the dataset. Both 18 and 19 train sizes look like good choices for hold out cross validation.

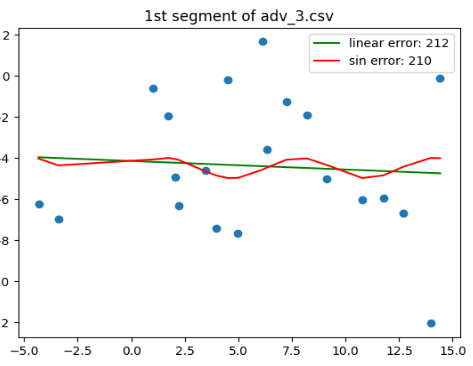


A big disadvantage of hold out method is the fact that it builds a single model which relies strongly on the order of the points in the dataset, making it extremely biased.

An advantage is the computational efficiency, precisely because it builds only one model.

Leave-one-out:

By applying the hold out on the first segment of “adv_3” the result is linear, which visually seems all right. But, when I apply leave-one-out, this respective segment prefers sinus function. The segment and both sin and linear functions are plotted on the right. The errors are similar in this case.



Leave-one-out treats the disadvantage of the hold out by considering all possibilities of splitting the dataset in 19/1. It builds and considers 20 models every time, being slower but much more accurate and reliable. Considering that there are only 20 points in a segment, I find the leave-one-out’s reliability and computational cost trade off being totally acceptable. With these being said, I strongly believe that leave-one-out cross validation is the right method to avoid overfitting.