CW1: An Unknown Signal

Originally by: Michael Wray & Davide Moltisanti

1 Introduction

The year is 1983, the strategy of Détente (de-escalation via verbal agreement) has failed. Both the United States of America and the Soviet Union have been stockpiling nuclear weapons and recent political events have brought the world on the cusp of total annihilation. As an operator for a nuclear early warning device you receive an unknown signal...

2 Task

In this coursework you will be given example points which follow an unknown signal. You will be required to reconstruct the signal and give the resulting error. Your mark will be based on a report you write about the code, submitted to the "Data-Driven Computer Science Coursework" submission point. You will also submit the code as a single python file to "Code submission point". We will check that the code satisfies the basic requirements (in the implementation section) and for plagarism, but we won't assess its performance.

You are required to work on this coursework individually. You may consult with your peers on the concepts and spirit of the assignment/solution, but you may not exchange code or passages of text for the report.

The deadline is April 30th at noon UK time.

3 Implementation

Your final solution will use the linear least squares regression code that you have created in the worksheets which will need to be extended in order to cope with non-linear curves. Your solution should:

- Read in a file containing a number of different line segments (each made up of 20 points)
- Determine the function type of line segment,
  - linear

  - polynomial with a fixed order that you must determine
  - unknown nonlinear function that you must determine
- Use maximum-likelihood/least squares regression to fit the function
- Produce the total reconstruction error
- Produce a figure showing the reconstructed line from the points if an optional argument is given.

Note: All Code written for the least squares regression sections must be written by yourself. The only external libraries your solution should use are:

- Numpy
- Pandas
- Matplotlib

Builtins e.g. from `os`, `sys`, or `argparse` are fine. But Scipy/Torch are not!

4 Report

The aim of this report is to demonstrate your understanding of methods you used, the results you obtained and your understanding of issues such as overfitting. The report should be no more than 4 pages long, using no less than 11 point font, with at least 2 cm margins. You should include:

- Equations for linear regression.
- Justification for your choice of polynomial order. Include plots.
- Justification for your choice of unknown function. Include plots.
- Description of your proceedure for selecting between linear/polynomial/unknown functions, accounting for overfitting.

5 Train/Test Files

The input files are Comma Separated Value (CSVs) which consists of two columns for the x and y dimensions respectively. Each input file will be split into different line segments each made up of 20 different points, so the first 20 points correspond to the first line segment, 21-40 the second line segment *etc.* Two consecutive line segments may/may not follow the same trend. A line segment will follow a linear, polynomial or another unknown function. An example train file is given below (basic_2.csv):

```
-2.9251047704944395,4.8735802196928955
-0.8554275589011144,8.298220787115126
0.19855925795710938,10.042225113162727
```

```
0.43421216060191,10.432153786922873
0.624770694735732,10.747465992520771
1.3614382007711343,11.96641038237006
1.5943359027154818,12.351780097835984
4.194060135884697,16.653475561819924
4.6779257660237565,17.45411532191076
5.023596843152395,18.026088181779052
7.288865840413205,21.774369344024162
7.485381750603618,22.099539063443302
7.932320124145074,22.839076261125182
9.697898711534522,25.760532815546593
9.79296969415338,25.91784427553536
10.338531977763036,26.820571869016863
10.47247053028027,27.042196476913674
11.373020960538451,28.532313640689885
12.448760928878032,30.31231233562508
12.518847008701732,30.428281932638267
12.653923639452994,30.27170891389091
12.717363846157376,30.149430959474596
12.922893495415568,29.753282422312626
13.82087982807715,28.02245685354787
16.940698414027672,22.009156225167718
17.888369346145012,20.182565974932835
19.383541905873596,17.300692607859034
19.669839705150725,16.748867336976346
20.098166635136398,15.923287731535833
20.564236539021604,15.024960354898298
20.621376796845194,14.914825249663732
21.505205350325678,13.211288120947877
27.70770290983017,1.2562716850382856
28.51062072023851,-0.2913138686887322
28.664485990065593,-0.5878817934724623
29.937739282405232,-3.042016420848668
30.093442528905104,-3.3421269574788823
30.886445731764947,-4.870602580881737
31.512235111191547,-6.076781582919168
31.755689686592167,-6.546028595495876
```

This corresponds to 40 points of x and y co-ordinates which make up the two different line segments of the file. Test files could consist of any number of line segments (*i.e.* longer than adv_3.csv) as well as have large amounts of noise.

6 Input/Output

The auto-marker will expect a program called **lsr.py** (**not a jupyter notebook**) which will be run as follows:

```
$ python lsr.py test_2.csv
```

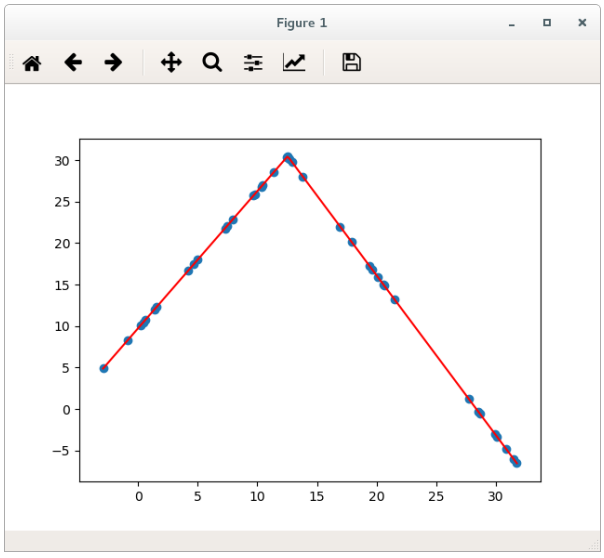
After execution your program should print out the total reconstruction error, created by summing the error for every line segment, followed by a newline character:

```
$ python lsr.py basic_2.csv
6.467081701001855e-27
$
```

Additionally, to further understand and validate your solution we wish for your program to be able to visualise the result given an optional parameter, `--plot`. This is only to visually check that your solution is working and will not be part of the automarker (styling of the plot itself doesn't matter). For example, running:

```
$ python lsr.py basic_2.csv --plot
6.467081701001855e-27
```

Will also show the following visualisation:



Note the presence of both the original points (blue dots) and fitted line segments (red lines).

7 Hints/Notes

- We have provided you a utilities file which contains two functions: *load_points_from_file* and *view_data_points*. The first will load the data points from a file and return them as two numpy arrays. The second will visualise the data points with a different colour per segment. Feel free to copy-paste this code into your final solution (it will be excluded from plagarism checks).
- The error you need to calculate is the sum squared error.
- Regardless of the number of segments your program needs to only output the total reconstruction error.
- The test files will include points approximating line segments generated from the three functions mentioned above, *i.e.* linear, polynomial or the unknown function.
- The order of the polynomial function **will not** change between different files.
- Having a very low error might not be the correct answer to a noisy solution, refer to overfitting in the lecture slides.