Data-Driven Computer Science

Coursework Report

**Introduction:**

The goal of this project was to read some points that would define a signal, and then approximate the function, plotting it and calculate the error.

I know that the linear segment could be linear, polynomial, or unknown function, so I decided to use the cross-validation method to approximate the function.

**Implementation:**

Firstly, I calculated the number of line-segments present presented in the .csv file and then, for every line segment, I am randomizing and then splitting the data in 4 groups. 3 of the represent the training part and the other one the data for testing. ( KFold() function)

**Approximating the coefficients and least square:**

Secondly, I am determining the right type of function. To do that, I use the linear regression and k-fold cross validation methods presented in courses.

* I obtain the coefficients for the linear and polynomial signal as we did in notebook 3; using a function of type Chebyshev (numpy.polynomial.chebyshev.chebval()).

Cheb function is returning the coefficients of a function with order c.

def cheb(*xs*, *c*):

coefs = c \* [0] + [1]

return np.polynomial.chebyshev.chebval(xs, coefs)

* I am creating a weight matrix with coefficients for different orders using the chebx function, and the linear regression formula. The chebx function is using cheb() to generate a matrix of coefficients from “x” values.

def chebx(*x*, *order*):

xs=cheb(x, 0)

for c in range(order-1):

xs = np.vstack([xs, cheb(x, c+1)])

return xs.T

def fit\_Wh(*X*, *Y*):

return np.linalg.inv(X.T.dot(X)).dot(X.T).dot(Y)

weight = fit\_Wh(chebx(x\_train, order), y\_train)

* Now, to obtain “y” of our guessed function we have to multiply the weight matrix with chebx of “x”.

yf = chebx(x\_test, order).dot(weight)

The most important thing is how I am deciding the order of the function and if it is polynomial, linear, or unknown.

**Choosing the order of polynomial:**

To determine the right order, I am using cross-validation. With the data being divided in train and test, I am calculating “the cross error” for every order from 1 to 19 and I am appending it in an array called error.

while(order < 20):

weight = fit\_Wh(chebx(x\_train, order), y\_train)

yf = chebx(x\_test, order).dot(weight)

cross\_error = ((y\_test - yf) \*\* 2).mean()

error = np.append(error, cross\_error)

order += 1

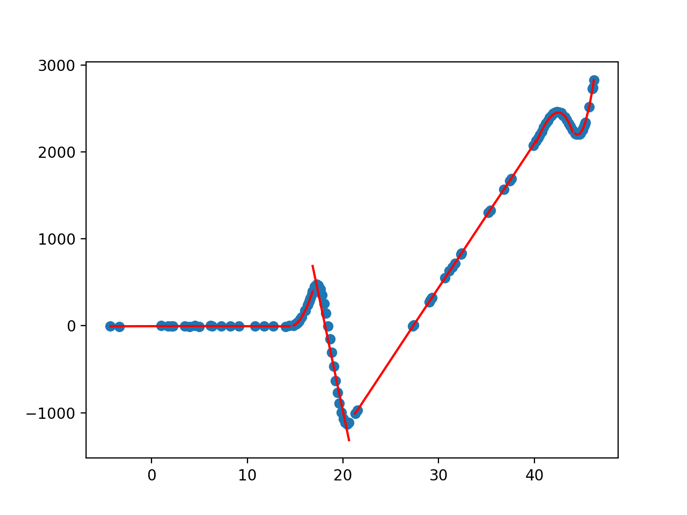
Now, to decide which is the closest order, I am iterating the array as long as the value decreases.

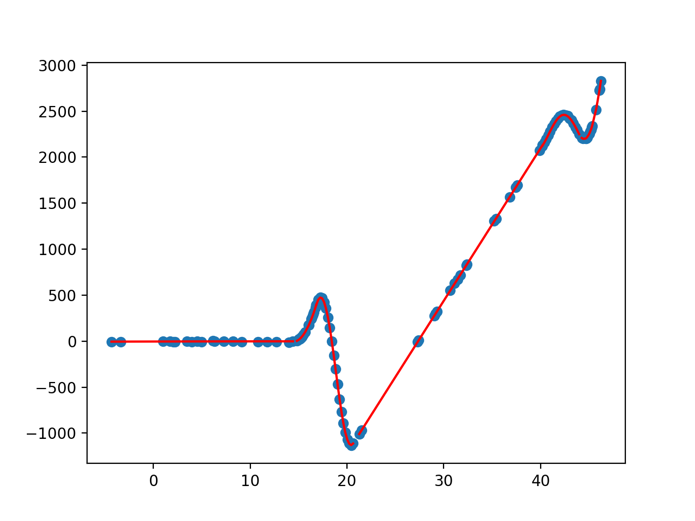
* For example, if the array is [4, 3, 2, 3, 1] then the order will be 3 because when I compare 2 with the second 3, it is smaller.
* If the order is 1, then the function is linear.
* If the order is bigger than 1 then is polynomial.

**Determining the unknown function:**

After I implemented this, I observed that some functions are not polynomial or linear. So, I begun trying different types of values to see which of them is suiting best. For the examples that or given, the best choice is “a \* sin(x) + b “.

**Those are the plots for the adv\_3.csv file. In the first one I used only polynomial and linear function, and in the second one, I used sinusoidal too.**

****

****

**Deciding between polynomial and sinusoidal:**

In the end, to decide between polynomial and sinusoidal function I am calculating the cross error for the sinusoidal function and compare it with the cross error of the closest order polynomial. If the error is smaller for the sinusoidal, I am choosing the sinusoidal, else I will go with the polynomial.