Project 2

ECE 8540 NON-LINEAR REGRESSION

September 13, 2022

Bradley Selee
Clemson University
Department of Electrical and Computer Engineering
bselee@clemson.edu

1 Purpose

This project extends the first project by fitting non-linear models instead of linear models. In this project, a non-linear regression fit was calculated in the form of $y = ln(ax)$. This was accomplished by an iterative approach using a variation of the Netwon-Raphson Method on given sets of data. Initial guesses were formed to begin the iterative approach and when the right guess was chosen, the algorithm converged to a finite value.

2 Methods and Materials

For this project, my implementation was created in Python. The only libraries used were NumPy for matrix operations, Pandas to load the datasets into a dataframe, and Matplotlib to plot all the graphs. The code was developed in Windows Subsystem for Linux (WSL) using Visual Studio Code as the text editor.

2.1 Code Design and Implementation

Initially, the datasets were visualized using a scatter plot to have a better understanding of the trend of data. With the shape of the data, the basis equation $y = ln(ax)$ was formed. Next, the first and second derivative of the basis equation were taken with respect to a. Inside a function, a loop is performed that impelents the equations from the Newton-Raphson method to find the next a in the sequence. The loop runs until the maximum number of iterations is reached; if this occurs, the initial guess was wrong and the function has to be repeated with a new guess. If the change in a values is below a small threshold, then the function is successful and the user can see the convergence value along with the number of iterations to converge. This function was repeated for all three datasets.

3 Results

There were three datasets given for this project. All three were in the shape of the natural log function. Below shows the derivation of the Newton-Raphson iteration, initial data visualization, the valid initial guesses for each dataset, and the final convergence value of a.

3.1 Newton-Raphson Algorithm Derivation

The algorithm is defined as

$$a_{n+1} = a_n - \frac{f(a_n)}{f'(a_n)}$$

In order to find the function $f(a_n)$ the derivative of E must be taken

$$E = \sum_{i=1}^{N} (y_i - ln(ax_i))^2$$

With the derivative being

$$f(a_n) = \sum_{i=1}^{N} 2(y_i - ln(ax)) \frac{-1}{a}$$

$$f(a_n) = \sum_{i=1}^{N} \frac{-2}{a}(y_i - ln(ax_i))$$

and setting the function equal to 0

$$f(a_n) = \sum_{i=1}^{N} \frac{-2}{a}(y_i - ln(ax_i)) = 0$$

$$f(a_n) = \sum_{i=1}^{N} \frac{1}{a}(y_i - ln(ax_i))$$

Next, the derivative of $f(a_n)$ must be taken

$$f(a_n) = \sum_{i=1}^{N} a^{-1}(y_i - ln(ax_i))$$

$$f'(a_n) = \sum_{i=1}^{N} -a^{-2}(y_i - ln(ax_i)) + (-a^{-1} \cdot a^{-1})$$

$$= \sum_{i=1}^{N} -a^{-2}(y_i - ln(ax_i)) - a^{-2}$$

$$f'(a_n) = \sum_{i=1}^{N} \frac{-1}{a^2}(y_i - ln(ax_i)) - \frac{1}{a^2}$$

3.2 Data Visualization

With all data, it is a good idea to visualize it to understand trends and patterns that may occur. In this case, it will help to form the basis equation. Figure 3.1 shows a scatter plot of Data A (0), Data B (1), and Data C (2)

From this, the data takes the form of the natural log so the basis equation can take the form of $y = ln(ax)$

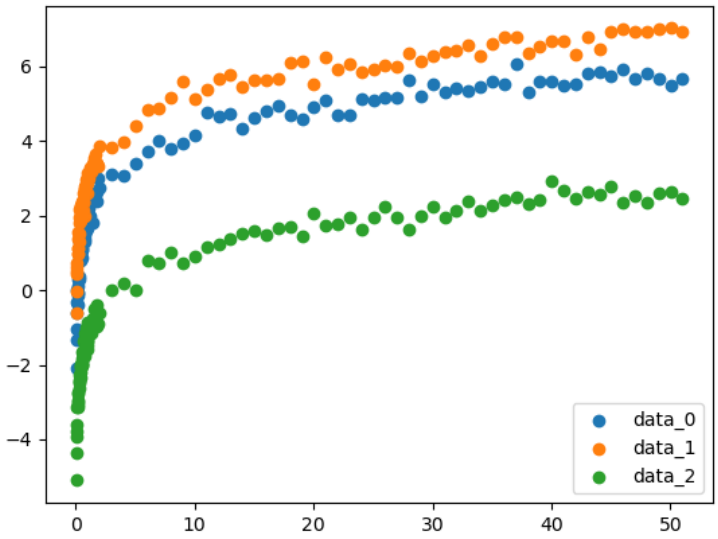

Figure 3.1: Scatter plot of the given data

3.3 Data A, B, and C Results

For each dataset, the initial value a_0 must follow

$$\begin{aligned} Data\ A: &\quad 0 < a_0 < 12 \\ Data\ B: &\quad 0 < a_0 < 32 \\ Data\ C: &\quad 0.00 < a_0 < 0.48 \end{aligned}$$

Table 3.1 shows the initial value chosen, the number of iterations taken to converge, and the convergence value for each dataset

Dataset	a_0	Final Value	Iterations
A	11.0	6.71136	13
B	7.0	18.99612	8
C	0.25	0.29000	5

Table 3.1 Results from each dataset

4 Conclusion

Overall, I think this project was successful. My implementation solves the derivatives for the Newton-Raphson method and iterates to a convergence value. It is important to

note how easily these functions can break if the wrong initial value is chosen. One initial value may blow up the function value and never converge, while another number could not exist after a few iterations. It takes a lot of trial and error to find good initial values. Data A and B were very easy to figure out, however data C took me a long time to figure out due to its small range of initial values that do not break the function. I believe this is not a stable method for general purpose model fitting because the functions can break too easily if an incorrect initial value is chose. An interesting extension to this project could be to use different methods of non-linear model fitting. If I did this project again I might have tried different datasets that would use a different basis equation. An fun equation to try might be an oscillating function like sine or cosine.

5 Appendix

The following appendix is the Python code for this lab.

```
"""lab2.py

Bradley Selee
ECE 8540
Project 2
Due: 09/13/2022
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

guess_a = 11 # converges to 6.71136. Good guesses are: x < 12 and x > 0
guess_b = 7 # converges to 18.99612. Good guesses are x < 32 and x > 0
guess_c = 0.25 # converges to 0.29000. Good guesses are x < 0.48 and x > 0.00
num_iterations = 1000 # Number of iterations incase the threshold isn't reached
threshold = 0.0000001 # The stopping condition

def scatter_plot(df, iteration):
    plt.scatter(df.iloc[:, 0], df.iloc[:, 1], label=f'data_{iteration}')
    plt.legend(loc="lower right")


def non_linear_regression(df, initial_guess):
    """Non linear regression with the basis function y=ln(ax)"""
    X = df.iloc[:, 0]
    y = df.iloc[:, 1]

    error = 1

    a = initial_guess
```

```
    print(f'Initial guess: {a}')
    for index in range(num_iterations):
        numerator = sum((1 / a)*(y - np.log(a * X)))
        denominator = sum(((-1 / a**2) * (y - np.log(a * X))) - (1 / a**2))
        new_a = a - (numerator / denominator)
        print(f'{a:.5f}\t{new_a:.5f}') # the f prevents scientific notation
        if abs(new_a - a) < threshold:
            print(f'Non-linear regression took {index+1} iterations to
                converge\n')
            error = 0
            break
        a = new_a
    if error:
        print(f'Error: Unable to find the true value. The max number of
            iterations has been reached {num_iterations}.\n')

def main():
    # Read data files into dataframe and visualize the values in a scatter plot
    data_a = pd.read_csv('log-data-A.txt', sep=' ', header=None)
    data_b = pd.read_csv('log-data-B.txt', sep=' ', header=None)
    data_c = pd.read_csv('log-data-C.txt', sep=' ', header=None)
    all_data =[data_a, data_b, data_c]
    for index, data in enumerate(all_data):
        scatter_plot(data, index)

    print('\nData A:')
    non_linear_regression(data_a, guess_a)
    print('\nData B:')
    non_linear_regression(data_b, guess_b)
    print('\nData C:')
    non_linear_regression(data_c, guess_c)

    plt.savefig('scatter_plot.png')
    plt.show()


if __name__ == '__main__':
    main()
```

References

No further references were needed for this project.