Hw 8: Curve Fitting & Linear Regression

Python Decal - Fall 2024

Due 11/23/24 at 11:59 pm

Curve fitting is a super important skill. These problems should help you get more comfortable with:

- curve fitting (using scipy.optimize.curve fit, pandas, and matplotlib) (you'll need this for your final project!)
- covariance matrices and error
- data cleaning, processing, and visualization

1 Curve Fitting Guided Problem

The following problem will walk you through how to fit a tricky curve to a set of data. (source)

This problem looks super long, but it's not :)

(1a) Use Pandas to do the following:

- 1. Read in the file "GlobalLandTemperaturesByState.csv".
- 2. Filter the table to include only the columns for the date, temperature, and state.
- 3. Filter the table to include only years after 2000.
- 4. Filter the table to include only the rows corresponding to Wyoming, Nebraska, or South Dakota. Check: your table should be 495 rows and 3 columns

(1b) Modify the table such that it contains the average temperature over all three states for each date. It should have two columns: date and average temperature.

- 1. Look into the function "groupby".

(1c) Use matplotlib.pyplot to plot the data from the table you created above.You can pass pandas columns directly into matplotlib without needing to turn them into arrays.

- 1. Date on the x axis, average temperature on the y axis.
- 2. Label the axis and give the graph a title. Now that the data is imported and plotted so you have an idea of what it looks like, let's get into the curve fitting.

(1d) scipy.optimize, unsurprisingly, can only do math with numbers. The date column of the table is currently composed of strings.

- 1. Fix this and convert the string date into numerical values however you see fit, and make it a column in the dataframe. Your numerical values should capture all parts of the date provided in the string (don't use just the year, etc).

(1e)scipy.optimize requires: a model equation, and an initial guess of parameters. For this section:

- 1. Define an appropriate model equation. Use a generic form like mx+b: there should be four parameters.
- 2. Make an initial guess at the parameters and save them in an array.
- This part is really important. A dataset with a non polynomial pattern was chosen for a reason: your initial guesses matter, particularly the period.
- If you're stuck, eyeball the length of the period (it should make physical sense, and remember that the units are in years) and keep this in mind: cos(2x) means the function covers two periods in the space of 2pi. 2 * period = 2pi, so each period is pi long.

(1f) Run scipy.optimize's curve fit function! Remember that it outputs a tuple containing two arrays: the parameter array and the covariance matrix.

- If while attempting this, you get one of the following errors:
- Something about maximum depth
- Something about not being able to estimate the covariance
- A line that does not fit the data at all You may need to re-examine your guesses for the initial parameters (particularly the period). This is why plotting the data before fitting it is critical.

(1g) Do the following:

- 1. Replot the data.
- Plot the line outputted by curve fitter on the same graph as the data.
- Make sure they mostly match up!

(1h) Do they match? If so, you made it! Your curve fitting code looks great. Non polynomial functions are tricky to fit, so congratulations on curve- fitting sinusoidal temperature variations! One last step (okay, two):

- 1. Use the covariance matrix to calculate the errors for each parameter. Recall that the errors are located down the diagonal of the matrix.
- E.g: Parameter 1's error is the square root of location [0,0] in the matrix, 2's error is at location [1,1], etc.

(1i) The final step! Print out your results!

- 1. Print out each parameter AND its corresponding error with format: parameter +/- error.
- Print out the final equation!

Congrats :) you did it!

Part 2: Advanced Plotting

2 Random Plotting Practice

This problem focuses on generating and visualizing random data in two different types of plots. First, you will create three lists of random numbers, each containing 50 elements, with values ranging from 0 to 200. Then, you'll create two instead of vertical subplots:

- 1. The left plot should display the first list of random data as a blue line with a line width of 5, and the second list in green with a dotted line style.
- 2. The right plot should be a scatter plot of the third list, using purple triangles as markers. The x-coordinate should correspond to the position of each element in the list.

Make sure to add appropriate titles and legends to each subplot.

3 Monte Carlo

Did you know that you can estimate the value of π using random numbers? Crazy statement but really cool.

First, you need to generate a large number of random points inside a 1x1 square with one corner at the origin (essentially you are making a unit circle).

Second, check how many of these points fall within a distance of 1 from the origin (in other words, a radius = 1). These points will lie inside a quarter circle, centered at the origin.

By using the ratio of points inside the quarter circle to the total number of points, we can estimate π! based on the relationship between the area of a quarter circle and the square.

Your task: - Calculate an estimate of π using this method of $N = 10, 10^3, 10^5, 10^6$. Print these results for each value of N. What do you notice as N gets bigger? - For $N = 10^4$, plot the points within the quarter circle in one color and the points outside in another color. Make sure the entire plots is a square. Display the estimate value of π on the plot. Don't forget to include a title and legend.