Polynomial Regression and Stochastic Gradient Descent for Data Analysis

This document entails the purpose of every written function, as well as the information learned throughout this project.

temp_block.py

temp_block.py contains the class TempBlock. TempBlock stores three things, all of which are entered at time of the creation of the object. TempBlock is used to store all relevant information from one row of the excel file.

xtemp- Stores the item given to it (should be temperature) into ‘myTemp’.

xtime- Stores the time given to it (I have done so in measurement number) into ‘myTime’

xclassification- Stores a classification- valid classifications presented in this are ‘s’, ‘p’, and ‘e’. ‘s’ represents start, ‘p’ represents peak, and ‘e’ represents end. Adding a number after the letter will indicate which start, end, or peak it is in relation to read in data. For instance- if it is the first start recorded in a data set, you should note it as ‘s1’.

temp, time, classification- return myTemp, myTime, and myClassification, respectively.

setClassification- allows retroactive change of a classification.

file_read_in

file_read_in contains the method readIn: its purpose is to read in the entirety of a excel file. The excel file is **expected** in the following format. The excel file opened is at location ‘loc’. It opens the first sheet in this excel file. Skip row 1, as it is the plot title. Skip row 2, as it is the column identifiers. For all following rows, until a blank row is reached, a tempblock is created and added to a list known as ‘tempprofile’.

Temperature is found in column **3.**
Time is found in column **1.**
Classification is found in column **4.**

Note: If all of column 4 is empty, the program gets confused. To subvert this, add a letter in the third column that starts with neither s, e, or p. This adds in a- for all purposes- empty classification and allows the fourth column to be read.

readIn returns tempProfile, which at this point is a list of temperature blocks that includes every temperature, time, and classification in order.

read_in_read_out

A brief test program that reads in the entirety of an excel sheet written in-code on line 3. It will return all temperatures in that excel sheet, in order.

polyRegression

The main program.

We will start with all helper functions.

create_function

Create function takes two arguments- expArray and xArray.

expArray is a list of scalers. They indicate how the function is created. For instance, take the list [2, 3, 4]. This would create a function 2 + 3x +4x^2.

xArray is a distance across which you want this function applied. [2 : 6] will apply the earlier argument for all numbers between 2 and 5.

In total: this function takes any number of scalers to create a function of any number of degrees, progressing from x^0 to x^n, where n is the length of expArray. That function is then applied to the range X, and the new list that contains all applications are returned.

half_split

No longer necessary in current code format, but code is still present in case of potential future use. half_split takes any number of integers and returns all integers that land directly between them. For instance, [0, 10, 20] returns [5, 15]. The original application was to find half points between all given peaks.

find_start

Takes three arguments- numberArray, upDifference, and downDifference. numberArray is a list of heights (in this case temperature data) that the program iterates through. upDifference is the positive height difference across three data points needed to indicate a total increase. downDifference is the negative height difference across three data points before the program will look for another height.

The program returns changeArray- a list of points containing detected starts.

find_acceleration

Takes the height over a number of points. This height is then extrapolated into velocity, by finding the change between one point and its neighbor. It expects there to be an equidistant change over all points. It then does the same for all velocities, finding an index of highest acceleration and returning it.

find_steepest

Contains arguments array and add. Steepest is meant to take a snippet of a larger list of heights, and return the index in the terms of the longer list of when the change between two points is greatest. Array is the list of heights, and *add* is the total offset the snippet is.

pair_start_and_end

Takes arguments startPoint, endPointArray, and peakPoint. startPoint and peakPoint are integers, while endPointArray is a list of all endPoints in a format of tempBlocks. If an endpoint falls between a peakPoint and a startPoint, that endpoint is returned. If there is no such endpoint, then '0' is returned instead.

polynomial_regression

This is the big one. It requires one argument, and optionally takes more.

tempArray2D- a list of all tempBlocks, in order. This is **required**.

startList- a list of all starts, which will be displayed in **green**. If not entered, it will be an empty list.

endList- a list of all ends, which will be displayed in **red**. If not entered, it will be an empty list.

Start and Endlist are meant for groundtruthed data, found in excel sheets.

peakList- a list of all peaks, which will be displayed in **blue**. If not entered, it will be an empty list.

calculatedStartList- a list of calculated-in-program starts, which will be displayed in light green. If not entered, it will be an empty list.

counter- the starting point of the counter of total derivations of the entirety of data. This counter goes up to eight. If not entered, it starts at 7, which means no total derivations will show up.

Lines 58-65. These lines create x and y, which graph time and temperature from tempArray2D respectively. This is then displayed in black.

Lines 68-99. These lines go through startList, endList, peakList, and calculatedStartList. They display all points of the respective lists in their respective colors.

Lines 102-106. These lines create a number of derivatives of the entire graph from 'counter' to 7.

Lines 111-118. These lines compare calculated starts to peaks. It grabs a start, and pairs it to the next peak. It adds this pair to startPointPairs.

Lines 123-136. These lines take every pair from 111-118. They use a machine learning program- polyfit- to create a derivative of degree 5. This derivative is then fed into create_function, along with the distance between start and end, and then returned as a function across this range. It is then placed on the graph in **dark green**.
Furthermore, these lines check every pair to see if an end from endList is between them. Why? If there is an end between them, this is a ground truthed segment. If it is ground truthed, it is added to traininglist- data that the machine learning program will use to train itself. If it is not, it is added to testlist- data that the machine learning program will use to test itself.

Line 157- This is the place that the machine learning takes place.

Main()

tempData is taken from the constant on line nine. It checks through every tempblock for classification. If the first letter is an ‘s’, it is put in the startList- list of starts. If the first letter is an ‘e’, it is put in the endList- list of ends. If the first letter is a ‘p’, it is put in the peakList- list of peaks.

This program calculates its own peaks, using the scipy.signal program. To do this, however, we need to ‘strip’ the data- strippedData is a list of only temperatures.

We then find all peaks across these temperatures- with the prominence of 2 and the distance of 20.

We then calculate all starts across these temperatures- using the find_start program, using .4 for upDistance and .2 for downDistance.

Now that we have these, we put them back in tempBlock format.

Finally, we call polynomial_regression. After this program is done, we add a legend to the graph, and show the graph.

Output_excel is not functioning yet and will require further aid. Its purpose is to create an excel file outputting important information.