Week 7: Assignment

Submit Assignment

Due Aug 18 by 11:59pm **Points** 100 **Submitting** a file upload

Instructions

Please make a python module called week_7_homework.py and put the answers to the problems below in it. All the problems below ask you to write functions or classes. Put ***only*** the functions in the file "week_7_homework.py" do not put any testing code in that file. To test your functions, create another python file week_7_homework_test.py. In this file you can import that functions from your homework file as follows:

```
import week_7_homework as w7h

# then we can test the functions/classes like so:

result = w7h.my_function()
print(result)

object_instance = w7h.My_class()
```

Please write the following functions. ***Make sure you name the functions/classes exactly as typed below.***

Problem 1

Write a function called list_is_in. This will take two arguments, `list1` and `list2`. It return a new list that is the same length as list1, and contains True or False depending on if each item in the list is found in list2 (see the example below). Make sure you code this in a way that is O(n) and not O(n*m) (where n is the length of list1 and m is the length of list2. You do not need to use anything outside of the built-in functions (you should not import any python packages).

```
list1 = [1, 2, 3, 4, 5, 6]
list2 = [1, 6, 10, 11]
is_in = list_is_in(list1, list2)
print(is_in)
# [True, False, False, False, False, True]
```

Problem 2

Write a function called mean_normalize. This function will take in an numpy array. We will assume the array represents temperature data, where each *column* is a different weather stations, and each row is an hourly temperature measurement. We want to mean normalize the data, which means we want to subtract the mean of the temperature data for each station from that stations data, and then divide by the stations std. For example, if the temperature series for one station was [32.3, 45.6, 39.9], we would subtract each value by the mean, which is 39.26 to get [-6.96666667, 6.33333333, 0.63333333]. Then we would divide by the standard deviation of this series, which is 5.448 to get [-1.27872403, 1.16247639, 0.11624764].

You can test the function with the following code:

```
import numpy as np

input_array = \
    np.array([[-0.47752694,  2.10771366,  0.59367963],
              [-0.99518782, -0.56534531,  0.01539558],
              [ 0.44353958,  1.31930398,  2.42232459]])

normalized_array = mean_normalize(input_array)

print(normalized_array)

# [[-0.22599586  1.02887216 -0.40627705]
# [-1.09600761 -1.35471375 -0.96997878]
# [ 1.32200347  0.32584158  1.37625583]]
```

Problem 3

Write a class called temperature_model. The __init__ method will accept a filepath to a csv file of numbers. The __init__ load the file into a numpy array using the loadtxt function from numpy (see the example below). Add a method called mean_normalize(). This method will return the array that was loaded, but mean normalized, using the exact same process you used in problem 2.

You can test this by loading the file 'temperature.csv' that is linked **here**.

Example of loading a csv file into a numpy array:

```
import numpy as np

filepath = "/dummy/file/path.csv"

my_array = np.loadtxt(filepath, delimiter=',')
```

The shape of this file is (100, 87600), you can treat it as 100 stations with 87600 hours of temperature data. You can transpose it to a matrix of

100 columns and 87600 rows with the following line:

B = A.T, where A is the (100, 87600) array, and B will now be a (87600, 100)

If you do not want to transpose it, and want to do something like the following to mean normalize:

station_means = A.mean(axis=1)

B = A - station_means

You will get an error, the reason is as follows:

When you want to subtract (or divided, etc..) a vector form a matrix it needs to line up with the correct dimension. What I mean is this, when you have the shape (100, 87600) this means it is a matrix with 100 rows and 87600 columns. And, you have a vector that is shape (100,), when you have a a 1-D vector like this, you can think of it as a row (rather than a column) - we have a row with 100 elements.

So we have a matrix with 100 rows and 87600 - columns, visualize that shape in your head. Now, we have a row that is 100 elements long that we want to subtract from the matrix. That won't work.. because our matrix has rows that are 87600 elements long! Obviously, we don't want to subtract a row of 100 elements from our matrix, we want to subtract a column of 100 elements from the matrix. Said another way,

we want to subtract a column of 100 elements from each of the 87600 columns. So we need to change the row of 100 elements to a column.

I.e. we want to have a matrix of (100, 87600) and column vector (100, 1). We can then subtract the column vector from each column of the matrix. How can we reshape the vector (100,) to be (1,100)?

We can do this:
M = A.mean(axis=1)
print(M.shape) # (100,)
M = M.reshape(1, -1).
print(M.shape) # (1, 100)

B = A - M

The way the reshape method works, is you just specify the new dimensions you want as the arguments and a '-1' indicates that you want to use all resining dimensions. So when do M.reshape(1, -1), we are specifying that we want there to be 1 row (that is the 1 in the first argument) and that we want the number of columns to be as many elements are left after making 1 row (this is what the -1) means.

Expirement with this by doing
M.reshape(2, -1).shape
M.reshape(4, -1).shape

Submission

Each problem is worth 30 points, and you get 10 points for turning something in (that is completely blank). Click on the blue button in the top right corner to submit your assignment. Submit the file with the functions, week_7_homework.py

Click Next (below) to progress through the course.