CodeLab – 1

Basics of Python (Numpy and Pandas) & Data Processing

Radar sensors have previously been successfully used to classify actions such as walking, carrying an item, and discriminating between people and animals' gaits, drones, and bird targets. All of this analysis used the phenomenon called Micro-Doppler, which is the additional movements a target has on top of its bulk velocity. For example, a person may walk forward at 3 m/s, but their arms and legs oscillate back and forth as they move at this speed. This movement created a signature that was coined as Micro-Doppler [1].

DopNet is a large radar database that contains Frequency Modulated Continuous Wave (FMCW) and Continuous Wave (CW) radar measurements of different gestures.

To evaluate how effectively a radar recognizes gestures, this challenge provides data that can be used to apply classification methods. A database of gestures has been created and uploaded here using the Ancortek Radar system. This database includes signals from 4 distinct types (Wave / Pinch / Click / Swipe). The data has been pre-processed so that the signatures have been cut into individual actions from a long data stream, filtered to enhance the desired components, and processed to produce the Doppler vs. time-domain data. The data is then stored in this format for it to be read in, features to be extracted and the classification process to be performed.

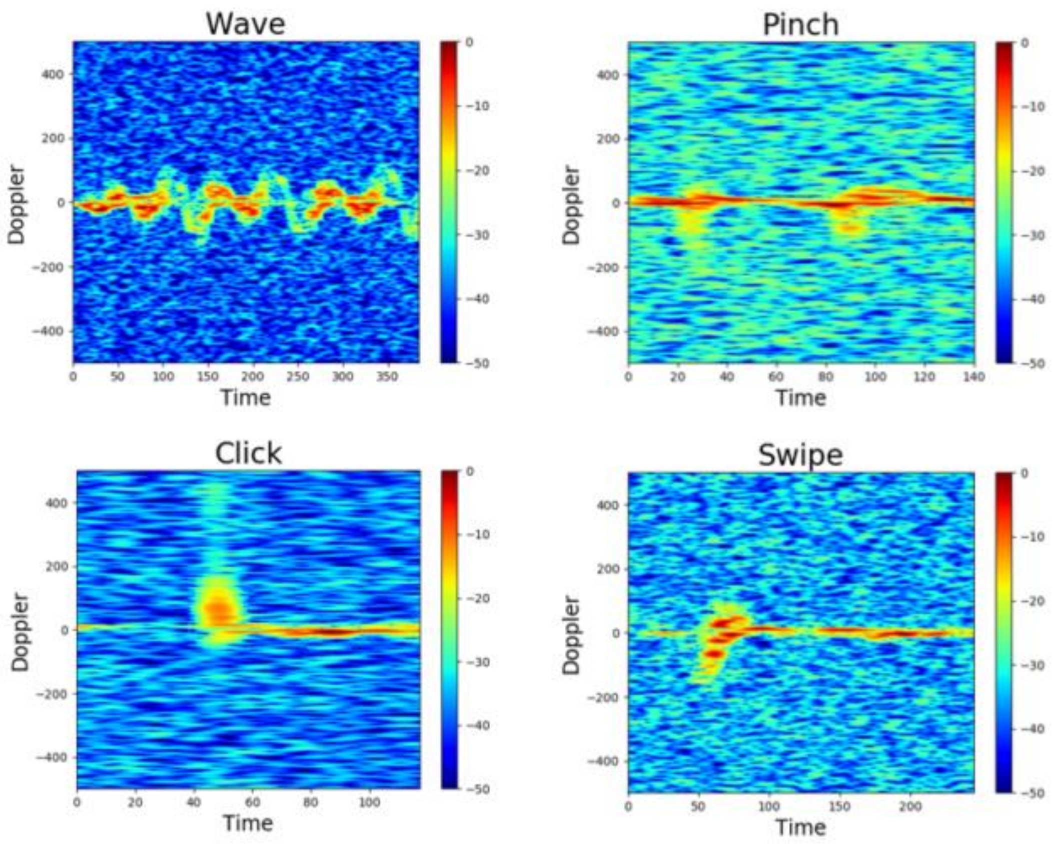

Figure 1. the spectrogram of the four gestures

In this CodeLab, you will use basic Python libraries Numpy, Matplotlib, and Pandas to get familiar with Python basics and data processing. Provide your answers in the Jupyter Notebook either as a comment or insert a markdown cell. Please print your Jupyter Notebook in PDF (or HTML) format and upload it to Brightspace. You will get points for each part separately, and you will get the grades based on your overall performance. Table 1 illustrates the grading scheme of CodeLab 1.

Table 1: Grading scheme of CodeLab1

Part	**Points**	**Grade**	**Points**
A	2	0	≤5
B	2	6	6-7
C	2	8	8-9
D	2	10	≥10
E	2		
Bonus	1		
Total	11		

Part A: Numpy Basics (2 points)

Basics (lists, arrays, dictionary)

Processed Dopnet data of 4 people is given as:

Person A	37.221	-36.3014	-35.4000	-40.9004
Person B	-34.8698	-34.3005	-33.7987	-39.0423
Person C	-40.8702	-42.6151	-35.8469	-35.2198
Person D	-34.5080	-40.1056	-42.3942	-45.5277

1) Create a list named "*P_A*" containing Person A's data, and print the created list.

2) Using Person B's data, create a list named "*P_B_abs*" that contains absolute values of the Person B data and print the list. (Hint1: Use the *abs()* function from the NumPy library to return the magnitude of the entries in the list.)

3) Make 2 separate lists, "*P_C_up*" and "*P_D_down*" where lists contain the rounded-up Person C and rounded-down Person D data. Print both lists. (Hint: You can use *ceil()* and *floor()* functions from NumPy)

4) For this part, create an array called "*CD*" that contains the newly created lists of Person C and Person D in a way that the first entry represents the list of Person C and the second entry Person D. To do so, you need to use the *array()* function of Numpy. Print the new array, its shape, and the summation of the two elements (as a list) in the array "*CD*" (CD[0]+CD[1]).

5) Define a dictionary "*DopNet*" with people as **keys** and their corresponding data as **values** (enter the values in the form of an array). Print the data of Person D from the "*DopNet*" dictionary. Print the data of Person D from the "*DopNet*" dictionary.

6) Create a vector (1-d array) of values starting from 1 to 24 with steps of 0.5. Use *arange()* function of NumPy. Print out the multiplication of all elements in the vector with *prod()* function.

7) Find and print the dot (*dot()*) and cross (*cross()*) products of given vectors "*a*" and "*b*". What is the difference between dot and cross product? a=[3,5,6] , b=[1,2,8]

8) *W1* list is given as "*W1*=[1,2,3]", assign a new list "*W2*" which is defined as "*W2*=*W1*". Replace the first element of *W2* with 0. Print both *W1* and *W2*. Write your observations. Create "*W3*" with the *copy()* function from "*W1*" and change its last element with 0. Print *W1*, *W2* and *W3* together.

Part B: Statistics (2 points)

1) Import the dataset *"PersonAclick.npy"* into array "A" using load() function of NumPy. Using the dataset's 6th (5th index) row, create an array called "*Click*". This array will be used to perform basic statistical operations.

- Find and print the largest element in "Click".
- Find and print the smallest element in "Click".
- Find and print the mean of "Click".
- Find and print the median of "Click".
- Find and print the standard deviation of "Click".

Use only NumPy functions to perform the tasks above.

2) First, generate the vector *X* containing 100,000 random variables (*random.normal()* function from Numpy) with a normal distribution of mean=10, and std=2. Then, plot the histogram of the vector *X* by using the *hist()* function. Set the setting to "*bins*=100 , *density*=True" to get the data distribution.

3) Plot the boxplot of X to show the quartiles of the data.

4) Plot the histogram of "*Click*" in part B1. Set "*density*=True" to get a density function. Try different bins. Define a title and labels for each axis (leave the axes labels as density (y) and units (x)). Write your observations. How does the number of bins affect the distribution?

5) Plot the boxplot of "*Click*". Define a title and labels for each axis (leave the axes labels as units (y) and null (x)). Compare histograms and boxplots of X and "Click".

6) Using the dataset (the one that was loaded), present a scatter plot where the x-axis is the first row and the y-axis is the second row. Define proper labels and the title.

Part C: Loops(for, while, do while) & If/Else statements (2 points)

1) Using the unscaled version of “*personAclick*”, ('.npy file'), assign the first 10 values from the file to an array called “*Reduction_A*” according to the following pattern in the figure and print.

(That is, apart from the first element, every first row(r+1) and second column (c+2). Red cells will be inserted into a new array.)

0 🟥	1	2	3	4	5	6	....
1		🟥					....
2				🟥			
3						🟥	
4							
5							
:							

2) Print the number of positive, negative and 0 elements in the first row of “*personAclick*”.

Part D: Data manipulation (Introduction to Pandas) (2 points)

1) Import the given four '.npy' files. Create four separate data frames corresponding to the imported files. Check the dimensions of these data frames using the '*dataframe.size()*' function. Print out the dimensions.

2) Reduce data frame size by creating new data frames by only taking every 8th column and then by taking every 12th row of the data frames created above. Hint: You can do this process in 2 separate steps: First, select every 8th column from the original set, then select every 12th from your reduced version.

3) Plot a spectrogram of these scaled-down data frames (4 separate plots), then compare it with the spectrogram created using the original data frame. Use the given spectrogram function “*specplot*”. Are the spectrograms produced by the scaled-down data frame able to produce the same observations as the original spectrogram? Why do we need this kind of scale-down?

Part E: Train-Test Split (2 points)

Create a training-test split function called “My_split”. The function shuffles the input data and selects random samples to generate training and test data sets determined by another user input “ratio” between [0.0-1.0].

Divide the data frame of “*personAclick*” into training and test sets with a ratio of 0.8 (80% Training, 20% Test) where each row represents a single data entry. What are the dimensions of the train and test data frames? Why do we need this process for machine learning applications?

Bonus : Stratified Train-Test Split (1 point)

Add a column (label vector) containing 0s and 1s to the data frame of "*personAclick*" of 800 elements with randomly placed 40-60 split of 0s and 1s (40% 0s, 60% 1s). Split the data frame into training and test sets with a ratio of 0.8, where the ratio of 0s to 1s remains the same. This split process is called a stratified train-test split. Show the ratio of 0s to 1s for both the training and test set.

References:

[1]"GitHub - UCLRadarGroup/DopNet", GitHub, 2022. [Online]. Available: https://github.com/UCLRadarGroup/DopNet. [Accessed: 26- May- 2022]