Assignment #4: Fourier Transform and PCA (total 10 points), due by 11:59 pm Monday, 25 November 2024

Exercise 1: Fourier analysis. The basis for touch-tone dialling on a phone is the Dual Tone Multi-Frequency (DTMF) system. Basically, the telephone dialling pad acts as a 4×3 matrix as shown below.

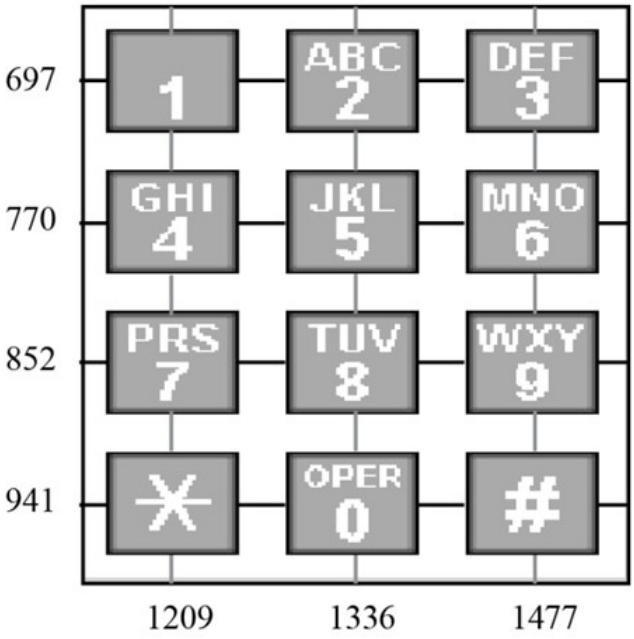

Figure 1: Telephone keypad.

A frequency (in Hz) is associated with each row and each column. These base frequencies are {697, 770, 852, 941} along each row and {1209, 1336, 1477} along each column. The tone generated by the button at a position (i, j) is obtained by superposing the two fundamental tones with frequencies f_{ri} at row i and f_{cj} at column j. For example, the signal corresponding to the button '5' can be created by superposing two sine functions as

$$y_5(t) = \frac{\sin(2\pi\, f_{r2}\, t) + \sin(2\pi\, f_{c2}\, t)}{2} \tag{1}$$

where t is the time (in seconds), f_{r2} represents the frequency at the second row (reading from top to bottom on the figure), and f_{c2} represents the frequency at the second column (reading from left to right on the figure).

(a) Script a function named `gentone` that can generate the tone associated with any key of the keypad from 0 to 9. Run the script and provide us with an example of a tone that we can execute and reproduce the result. Plot the signal as a function of time. **(1 point)**

(b) Apply Fast Fourier Transform (FFT) on the signal generated in (a) and check if you obtain the correct frequencies. Provide the code that reads the signal and performs the FFT. Plot the power spectrum evidencing the frequencies associated with the input signal. **(1 point)**

(c) Generate a 2-digit time signal in which a person presses any digit on the keypad and, after a fraction of a second, the person presses a different digit on the keypad. Add a relatively small random noise to the signal. Provide the script and plot the signal as a function of time. **(1 point)**

(d) Apply a short-time Fourier transform (STFT) on the signal generated in (c). Provide a time versus frequency representation plot and check if the frequencies (and time information) correspond to what the input signal carries. **(1 point)**

(e) The file `phone.dat` contains the signal recorded when an 11-digit phone number is dialled. Analyze this signal and identify the phone number. Provide the code with the Fourier analysis output that reveals the phone number. The file does not contain time information but typical sampling rates in telephone audio data is 8000 Hz. Note that for this problem, the ideal sampling rate is not exactly 8000 Hz, you will need to adjust that slightly based on the knowledge of the frequencies given on the keypad and aliasing. **(1 point)**

*****

Exercise 2: Principal components analysis (PCA). In the textbook *Advanced Data Analysis from an Elementary Point of View* by Cosma Rohilla Shalizi available at `https://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/`, PCA is covered in Chapter 15 in which Section 15.2 introduces an example of PCA to analyze a dataset storing cars' features. The dataset file is provided in the assignment D2L. For this exercise, only perform PCA computations on the features associated with numerical entries as Cosma presents on page 355.

(a) Make a grid-style (pairwise) plot of the raw multivariate data. This visualization should be as explained in the routine `pairplot` at `https://seaborn.pydata.org/generated/seaborn.pairplot.html` for Python. Explain what this result depicts. **(1 point)**

(b) Make a covariance heatmap plot as illustrated at `https://seaborn.pydata.org/generated/seaborn.heatmap.html` for Python. Explain what this result depicts. **(1 point)**

(c) Apply PCA onto the data and check if you can reproduce the components' results of Cosma. Make a Scree plot to decide on how many principal components are needed to explain at least 80% of the data. Explain your analysis and results. **(1 point)**

(d) Make a PCA plot (first principal component along the horizontal axis versus second principal component along the vertical axis) with the transformed/projected data. How do you interpret this plot? **(1 point)**

NOTE: All codes for both exercises need to be uploaded to the Gradescope.ca platform. Their content as well as their outputs need to be explained using comment lines, markdowns, and/or docstrings. This item will evaluate the clarity, optimization, and readability of your codes. Codes with too many redundancies will have points deducted. Note that if a code designated for a particular exercise item above is not uploaded or is not enough commented/explained, points will be deducted not only from this item but also from the designated item that the code is supposed to cover. Codes uploaded without testing and with syntax errors will result in various deductions of points since we cannot verify the answers. **(1 point)**

*****

Submission Information

IMPORTANT: POINTS WILL BE DEDUCTED IF THESE SUBMISSIONS PROCEDURES ARE NOT FOLLOWED.

Name the code for exercise 1 as `code1_assignment_4.ipynb`, the code for exercise 2 as `code2_assignment_4.ipynb`. **The solution code for each exercise needs to be placed in different files as described. Do not combine codes of different exercises on the same Jupyter Notebook file.** Note that only Jupyter Notebook files are accepted for the coding part. You will upload your codes to the Gradescope platform. **It is important that you log in to Gradescope.ca and not Gradescope.com.** Your codes need to be documented, i.e., introduce comment lines to explain their main procedures. Pure lines of code without explanatory comments will have reduced marks. You can upload multiple files to the Gradescope platform and you can resubmit your work until the due date. Upload all files at once. We will test-run your submitted codes for syntax errors and check if they generate the requested outcomes. We will also check if the results/figures presented in your assignment are 'paper-like' quality and that the quantitative predictions are scientifically/mathematically founded.

ONLY `pdf` (for the written reports), `jpg/jpeg` (for figures), and `.ipynb`, (for codes) FILES ARE ACCEPTED. This means we are only accepting Python Notebooks.

IMPORTANT: In your codes include instructions on how to run it and, if applicable, include testing values for the initial conditions or settings that you attempted so we can reproduce the results. We will test other initial conditions and parameter variations to evaluate the robustness of your code. But we need a place to start. Include also information about the version of your coding platform. If this information is not included, points will be deducted.

IMPORTANT: Your codes need to output all required numbers of the exercises and they should be printed clearly. The printed values should be properly identified with text saying what is being printed. We can only assess and consider marks for numerical answers printed by your code (we

cannot give marks to what we cannot see). We cannot debug your code and/or search the answers inside your code for you. This is a graded component of the course, therefore, we can ONLY evaluate what is uploaded and see printed on the screen. We will test other parameter variations and settings in your code, but we will not alter its structure and logistics. Interpret your code as any other graded component as 'in paper', meaning what you submitted is your final answer and what is not provided (or printed) in terms of answers, we cannot consider for marks.

IMPORTANT: (Follow-up from the item above) If an exercise asks you to write a code to compute or calculate a value, we will need to see the calculated output value and not simply a number manually printed using the `print` command. For instance, if an exercise asks you to write a code to obtain the velocity of a particle, we need to see the output velocity of the algorithm you implemented and not simply `print(2.34, 'km/s')`.

*****