MSE-213 Autumn 2024 – Python Practice Week 4

Run in the conda mse213 environment, use a Jupyter notebook. If you have trouble with that, you can use the online environment noto.epfl.ch

If you don't know how to proceed, stackoverflow.org, google.com, chatgpt.com can be very helpful. For a specific Numpy function, numpy.org has the documentation, or just type "np.function?" into Jupyter (where function is the name of your function of course).

Task 1: Python lists VS Numpy arrays

Aim: Learn the difference between python lists and numpy arrays. Practice creating lists of given lengths and studying the data.

- Create an empty python list
- Using nested for loops, create a matrix of size N*N with N=100, 1000 and 3000 filled with numbers following a normal distribution. Use the `np.random.normal(mean, std)` function to get your numbers, and the `append(x)` function of python lists to add elements to your lists.
    - A matrix is simply a « list of lists »
    - Use `timeit.default_timer()` to time the creation of these lists :

```
import timeit
start_time = timeit.default_timer()
# Your code here

# ---
end_time = timeit.default_timer()
print("Time taken : ", end_time-start_time)
```

- Do the same thing, using numpy's `random.normal(min, max, size)` function, making use of the size parameter.
- Compare the time taken to create these lists.
- Compare the time taken to achieve this with python loops when N = 100, 1000, 3000. Does it seem linear?
- Optional: Take other values of N and plot the time taken to complete the creation. How long do you think it would take to create with python loops for N = 10000? 100000?

- Using the numpy matrix, get the mean and standard deviation of each row.
    - Either make use of `np.mean's` axis parameter, or loop over the lines of your matrix using for loops.
- Find the maximum/minimum values of your matrix, as well as their row and column.
    - Use the `np.array.max()/np.array.min()` functions to get the values, and `np.array.argmax()/np.array.argmin()` to get their indice.

Task 2 : The Gaussian Distribution

Aim : Practice creating and visualising the gaussian distribution.

- Create a numpy array of N random numbers with a gaussian distribution.
- Plot a histogram of these values. Use bin sizes of 0.1, with one bin centered on 0.
    - Add a vertical line indicating the expectation value from the gaussian function. Use `plt.axvline(x)` to get a vertical line on your graph.
- Extract and plot the frequencies of your random gaussian distribution
    - Use the numpy `np.histogram(data, bins)` to get the values of the frequencies as a list.

- For each bin, compute the expected normalised gaussian value at the center of the bin and plot the result.
- Compute the difference between your frequencies from the random gaussian distribution, and the expectation value from the gaussian. Plot the result and discuss.