COL 786 - Advanced Functional Brain Imaging - Assignment 4
Develop a pre-processing Tool for fMRI Data

Due Date: 9:00pm, Wednesday, 21rd April, 2021
Max Marks : 100

Notes:

- The objective of this assignment is to develop a command-line based pre-processing tool **preproc** in python for fMRI data. Given an image in **NIFTI** format, the tool should be able to perform the following functions:
  1. slice time correction using linear interpolation
  2. band pass temporal filtering
  3. spatial smoothing
- **Usage:** `python preproc.py -i <input> -o <output> [-tc t <slice_time_acquisition_file> ] [ -tf high low ] [ -sm fwhm ]`
- The command line options in the square brackets (e.g., [-sm fwhm]) are all **optional**; the tool should perform functions corresponding to the specified arguments in the order in which they apprear:
  INPUT:
  1. **-i input** Input image file name
  2. **-o output** Output filename prefix. Eg. file name = abc.txt, then **output** = abc
  3. **-tc t slice_time_acquisition_file**
     target time (**t**) in milliseconds for slice time correction (the output file shall contain slice time corrected image corrected to the target time)
     slice_time_acquisition_file, (the number of lines in this file is <Z>, the number of slices in the image; every line consists of a single value, with the i^{th} line denoting the time (in milliseconds) when the i^{th} slice was acquired)
  4. **-tf high low**
     the cut offs in seconds (say **high** and **low**) for temporal filtering (remember to convert the input in seconds to frequency)
  5. **-sm fwhm** (in mm) for spatial smoothing

  OUTPUT:
  1. It should create a file **output.nii.gz** which contains the slice-time-corrected images, is temporally filtered and spatially smoothed.
  2. It should also create a file **output.txt** which contains the timeseries of the specified voxel.

  Note that the order of functions (slice time correction, temporal filtering and spatial smoothing) given in usage has to be respected when invoking the tool. This is required since these operations performed in different orders may lead to differences in the output.

Example Usage:

1. For performing slice time correction only:
   python preproc.py <input> <output> -tc t <slice time acquisition file>
2. Similarly, in order to perform slice time correction and spatial smoothing, the usage would be:
   python preproc.py <input> <output> -tc t <slice time acquisition file> -sm fwhm
3. To print the timeseries, perform temporal filtering and spatial smoothing, it would be:
   python preproc.py <input> <output> -ts xcoord ycoord zcoord -tf high low -sm fwhm

- You will need to **determine the TR** from the header of the input file (be careful about the units).
- There will be **no data provided** for this assignment. You may use image.nii.gz files from the previous assignments. Please note your program should work for any image file, so do not hard code the file names in your script.
- The datatype of the input will either be **int16** or **float32**. **The output should have the same datatype as the input.**
- The use of any standard libraries (such as for convolution, high pass/low pass filtering) is not allowed. The only allowed libraries are **fft** and **ifft** (used for Fast Fourier Transforms).
- The assignment is to be done individually.
- Details of each function are given at the end.
- You are required to submit **preproc.py** . The submission is to be done on **moodle**. **The assignment may be auto-graded, so make sure that the specifications are followed.**
- The assignment will take time. Start early. Do not postpone till the last few days.
- For doubts send an email to aman.bhardwaj@cse.iitd.ac.in

1 Slice time correction [30 marks]

With the advent of multi band MRI machines, it is now possible to acquire multiple slices in parallel. This has lead to a significant decrease in the TR time. For instance, we can now acquire data at the rate of **TR** = 645**ms.** Your tool should be able to perform slice time correction using linear interpolation. Your script will be given as input, a Slice Time acquisition file. The number of lines in the Slice Time acquisition file should be equal to <Z> (number of slices in the image). Each line in the file contains a single value; the value in the i^{th} line denotes the time (in milliseconds) at which the i^{th} slice was acquired. Note that this time should vary from 0 to TR and that any other value would be an error. **Your tool should perform this check and throw an error if need be**. A *sample Slice Time acquisition file* with 72 slices and TR=645 ms is available at: *acquisition file*.
Your script should bring all the slices to target time (specified as a command line argument) using linear interpolation. For the sake of simplicity, keep the last volume constant.

The tool should:

1. create a file **output.nii.gz** which contains slice time corrected image corrected to the target time.
2. print SLICE TIME CORRECTION SUCCESS or SLICE TIME CORRECTION FAILURE in **output.txt** based on whether or not the input was correct

Thus, this function utilizes the following command line arguments:

1. image file name (**input**)
2. Target time (**t**)
3. Slice time acquisition file
4. output file name (**output**)

2 Temporal filtering [35 marks]

Your tool should be able to perform band pass temporal filtering. The high and low cutoffs (in seconds) would be provided as input arguments (note that these cutoffs need to be converted to frequency). The only libraries allowed for this are **fft** and **ifft** i.e,Fast Fourier transforms. The tool should either create a new temporally filtered file output.nii.gz (if not already created after slice time correction) or should use the slice time corrected file as input and perform temporal filtering on it. Thus, this function utilizes the following command line arguments:

1. image file name (**input**): only if slice time correction was not done
2. cutoff for the high pass filter (**high**)
3. cutoff for the low pass filter (**low**)
4. output file name (**output**)

3 Spatial smoothing [35 marks]

Your tool should be able to perform spatial smoothing. The Full Width Half Maximum (in mm) will be given as input argument. The tool should either create a new spatially smoothed file output.nii.gz (if not already created after slice time correction or temporal filtering) or should use the slice time corrected/temporally filtered file as input and perform spatial smoothing on it. It is advisable to perform spatial smoothing in the three dimensions separately (using 1D kernels) instead of using a 3D kernel as the former is a more efficient implementation. Thus, this function utilizes the following command line arguments:

1. image file name (**input**): only if neither slice time correction nor temporal filtering was done
2. fwhm (in mm) (**fwhm**)
3. output file name (**output**)

Common mistakes that may result in failure of testcases:

1. Output file names different from the names as instructed in each question. Example *output1.text, output_slicetime.nii.gz, OUTPUT.txt, out.nii.gz.* Please keep the exact names as instructed.
2. Un-necessary console print statements. "Eg. print("Starting slice time correction..."), print("processing complete.")". You may use these statement during development. But make sure you comment them out before the final submission.
3. Don't hard code the input file name arguments. Please read them from the `system args` as instructed in Example usage.
4. Output file datatype not same as the input.
5. For Q1 print the exact error, either *SLICE TIME CORRECTION SUCCESS / SLICE TIME CORRECTION FAILURE*