# Machine Learning for Signal Processing
# (ENGR-E 511)
# Homework 3

**Due date: Mar. 6, 2022, 23:59 PM (US Eastern)**

## Instructions

- Submission format: Jupyter Notebook + HTML)
  - Your notebook should be a comprehensive report, not just a code snippet. Mark-ups are mandatory to answer the homework questions. You need to use LaTeX equations in the markup if you're asked.
  - Google Colab is the best place to begin with if this is the first time using iPython notebook.
  - Download your notebook as an .html version and submit it as well, so that the AIs can check out the plots and audio. Here is how to convert to html in Google Colab.
  - Meaning you need to embed an audio player in there if you're asked to submit an audio file
- Avoid using toolboxes.

## P1: DCT and PCA [5 points]

1. I like walking in the B-Line trail (although it doesn't mean that I have time to walk there frequently). `IMG_1878.JPG` is the photo I took there. Load it and divide the 3D array ($1024 \times 768 \times 3$) into the three channels. Let's call them $\boldsymbol{X}^R$, $\boldsymbol{X}^G$, and $\boldsymbol{X}^B$.
2. Randomly choose a block of 8 consecutive (entire) rows from $\boldsymbol{X}^R$, e.g. $\boldsymbol{X}^R_{(113:120,1:768)}$ (See the red box in Figure 1). This will be a matrix of $8 \times 768$. Collect another 2 such blocks, each of which starts from a randomly chosen first row position. Move on to the green channel and extract another three $8 \times 768$ blocks. Blue channel, too. You collected 9 blocks from all three channels. Now, concatenate all of them horizontally. This will be a matrix of $8 \times 6912$ pixels. Let's call it $\boldsymbol{R}$.
3. Subtract the mean vector of size $8 \times 1$ from all 6912 vectors.
4. Calculate the covariance matrix, which will be an $8 \times 8$ matrix.
5. Do eigendecomposition on the covariance matrix (feel free to use a toolbox) and extract 8 eigenvectors, each of which is with 8 dimensions. Yes, you did PCA. Imagine that you convert the original $8 \times 6912$ matrix into the other space using the learned eigenvectors. For example, if your eigenvector matrix is $\boldsymbol{W}$, than $\boldsymbol{W}^\top \boldsymbol{R}$ will do it. Plot your $\boldsymbol{W}^\top$ and compare it to the DCT matrix shown in M02-S21. Similar? Submit your plot and code.

Figure 1: B-Line trail

6. We just saw that PCA might be able to replace DCT. But, it seems to depend on the quality of PCA. One way to improve the quality is to increase the size of your data set, so that you can start from a good sample covariance matrix. To do so, go back to the $\boldsymbol{R}$ matrix generation procedure. But, this time, increase the total number of blocks to 90 (30 blocks per channel). Note that each block is with $8 \times 768$ pixels once again. See if the eigenvectors are better looking (submit the plot).

## P2: Instantaneous Source Separation [6 points]

1. From `x_ica_1.wav` to `x_ica_4.wav` are four recordings we observed at an audio scene. In this audio scene, there are three speakers saying something at the same time plus a motorcycle passing by. You may want to listen to those recordings to check out who says what, but I made it very careful so that you guys cannot understand what they are saying. In other words, I multiplied a $4 \times 4$ mixing matrix $\boldsymbol{A}$ to the four sources to create the four channel mixture:

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} s_1(t) \\ s_2(t) \\ s_3(t) \\ s_4(t) \end{bmatrix} \tag{1}$$

2. But, as you've learned how to do source separation using ICA, you should be able to separate them out into four sources: three clean speech signals and the motorcycle noise. Listen to your separated sources and transcribe what they are saying. Submit your separated .wav files along with your transcription.

3. At every iteration of the ICA algorithm, use these as your update rules:

$$\Delta \boldsymbol{W} \leftarrow \left(N\boldsymbol{I} - g(\boldsymbol{Y})f(\boldsymbol{Y})'\right)\boldsymbol{W} \tag{2}$$
$$\boldsymbol{W} \leftarrow \boldsymbol{W} + \rho\Delta\boldsymbol{W} \tag{3}$$
$$\boldsymbol{Y} \leftarrow \boldsymbol{W}\boldsymbol{Z} \tag{4}$$

where

$$\boldsymbol{W} : \text{The ICA unmixing matrix you're estimating} \tag{5}$$
$$\boldsymbol{Y} : \text{The } 4 \times N \text{ source matrix you're estimating} \tag{6}$$
$$\boldsymbol{Z} : \text{Whitened version of your input (using PCA)} \tag{7}$$
$$g(x) : \tanh(x), \text{(works element-wise)} \tag{8}$$
$$f(x) : x^3, \text{(works element-wise)} \tag{9}$$
$$\rho : \text{learning rate} \tag{10}$$
$$N : \text{number of samples} \tag{11}$$

4. Don't forget to whiten your data before applying ICA!

5. Implementation notes: Depending on the choice of the learning rate the convergence of the ICA algorithm varies. But I always see the convergence in from 5 sec to 90 sec in my desktop computer.

## P3: Ideal Masks [4 points]

1. `piano.wav` and `ocean.wav` are two sources you're interested in. Load them separately and apply STFT with 1024 point frames and 50% overlap. Use Hann windows. Let's call these two spectrograms $\boldsymbol{S}$ and $\boldsymbol{N}$, respectively. Discard the complex conjugate part, so eventually they will be an $513 \times 158$ matrix[1]. Later on in this problem when you recover the time domain signal out of this, you can easily recover the discarded half from the existing half so that you can do inverse-DFT on the column vector of full 1024 points. Hint: Why 513, not 512? Create a very short random signal with 16 samples, and do a DFT transform to convert it into a spectrum of 16 complex values. Check out their complex coefficients to see why you need $N/2 + 1$, not $N/2$.
   I will allow you to use other implementations, such as `librosa.stft`, but I strongly encourage you to reuse your code from Homework 2.

2. Now you build a mixture spectrogram by simply adding the two source spectrograms: $\boldsymbol{X} = \boldsymbol{S} + \boldsymbol{N}$. Note that all the numbers here are complex values.

3. Since you know the sources, the source separation job is trivial. One way is to calculate the ideal masks $\boldsymbol{M} = \frac{\boldsymbol{S}}{\boldsymbol{S}+\boldsymbol{N}}$ (once again, note that they are all complex valued and the division is element-wise). By the definition of the mixture spectrogram, $\boldsymbol{S} = \boldsymbol{M} \odot \boldsymbol{X}$, where $\odot$ stands for a Hadamard product. But we won't use this one today.

[1] The exact number of columns may be different depending on your STFT setup. If it's in the same ball park, it's okay.

4. Sometimes we can only estimate a nonnegative real-valued masking matrix $\bar{\boldsymbol{M}}$ especially if we don't have an access to the phase of the sources. For example, $\bar{\boldsymbol{M}} = \frac{|\boldsymbol{S}|^2}{|\boldsymbol{S}|^2+|\boldsymbol{N}|^2}$. Go ahead and calculate $\bar{\boldsymbol{M}}$ from your sources, and multiply it to your mixture spectrogram, i.e. $\boldsymbol{S} \approx \bar{\boldsymbol{M}} \odot \boldsymbol{X}$. Convert your estimated piano spectrogram back to the time domain. Submit the .wav file of your recovered piano source.

5. Listen to the recovered source. Is it too different from the original? One way to objectively measure the quality of the recovered signal is to compare it to the original signal by using a metric called Signal-to-Noise Ratio (SNR):

$$SNR = 10 \log_{10} \left( \frac{\sum_t \{s(t)\}^2}{\sum_t \{s(t) - \hat{s}(t)\}^2} \right), \tag{12}$$

   where $s(t)$ is the $t$-th sample of the original source and $\hat{s}(t)$ is that of the recovered one. Evaluate the SNR between `piano.wav` and your reconstruction for it. Note: their lengths could be slightly different. Just ignore the small difference in the end.

6. Yet another masking scheme is something called Ideal Binary Masks (IBM). This time, we use a binary (0 or 1) masking matrix $\boldsymbol{B}$, which is definded by

$$B_{ft} = \begin{cases} 1 & \text{if } |S|_{ft} > |N|_{ft} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

7. Create your IBM from the sources, and apply it to your mixture spectrogram, $\boldsymbol{S} \approx \boldsymbol{B} \odot \boldsymbol{X}$. Do the inverse STFT. How does it sound? What's its SNR value?

8. Don't forget to create audio players for the sound examples in your iPython (i.e., jupyter or Google Colab) notebook. Check if they play sound in the .html version.