Machine Learning for Signal Processing
(ENGR-E 511)
Homework 4

Due date: Mar. 27, 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: kNN Source Separation [5 points]

1. Prepare your $\boldsymbol{S}$ and $\boldsymbol{N}$ matrices by using the procedure in Homework 3 P3.1. In addition to them, let's create another matrix $\boldsymbol{G}$, the magnitude spectrogram of `trs.wav`+`trn.wav`. Also, let's create yet another matrix called Ideal Binary Masks (IBM), $\boldsymbol{B}$ defined in Homework 3 P3 (or repeatedly as follows):

$$\boldsymbol{B}_{f,t} = \begin{cases} 1 & \text{if } \boldsymbol{S}_{f,t} \geq \boldsymbol{N}_{f,t} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

   My hope is that $\boldsymbol{S} \approx \boldsymbol{B} \odot \boldsymbol{G}$. This sounds a bit too rough, but you know it works to some degree as shown in Homework 3 P3.

2. Load `x_nmf.wav` and convert it into a complex-valued spectrogram $\boldsymbol{X}$ and its magnitudes $\boldsymbol{Y}$, respectively. This time, we compare each column of $\boldsymbol{Y}$ with all columns in $\boldsymbol{G}$ to find out k nearest neighbors. Then, for t-th given test frame in $\boldsymbol{Y}$ we have a list of indices to the k nearest frames in $\boldsymbol{G}$, e.g. $\{i_1, i_2, \cdots, i_k\}$, where $i_1, i_2, \cdots, i_k \in \{1, \cdots, 990\}$, where 990 is the number of frames in $\boldsymbol{G}$ (yours can be slightly different depending on the STFT setup). For the comparison, I used Euclidean distance for no good reason, but you can choose a different (more proper) one if you want.

3. What we want to do with this kNN search is to predict the unknown IBM of the test signal, $\boldsymbol{D} \in \mathbb{B}^{513 \times 131}$ (again 131 is the number of frames in $\boldsymbol{Y}$). By using the nearest neighbor indices, I can collect the IBM vectors from $\boldsymbol{B}$, i.e. $\{\boldsymbol{B}_{:,i_1}, \boldsymbol{B}_{:,i_2}, \cdots, \boldsymbol{B}_{:,i_k}\}$. Since $\boldsymbol{Y}_{:,t}$ is very similar

to $\{\boldsymbol{G}_{:,i_1}, \boldsymbol{G}_{:,i_2}, \cdots, \boldsymbol{G}_{:,i_k}\}$, I hope that my $\boldsymbol{D}_{:,t}$ should be similar to $\{\boldsymbol{B}_{:,i_1}, \boldsymbol{B}_{:,i_2}, \cdots, \boldsymbol{B}_{:,i_k}\}$. How do we consolidate them though? Let's try median of them, because that'll give you one if one is the majority, and vice versa:

$$\boldsymbol{D}_{:,t} = \text{median}(\{\boldsymbol{B}_{:,i_1}, \boldsymbol{B}_{:,i_2}, \cdots, \boldsymbol{B}_{:,i_k}\}). \tag{2}$$

4. Repeat this for all your test frames (131 frames in my STFT) and predict the full $\boldsymbol{D}$ matrix. Recover the complex-valued spectrogram of your speech by masking the input: $\hat{\boldsymbol{S}}_{test} = \boldsymbol{D} \odot \boldsymbol{X}$. Convert it back to the time domain. Listen to it. Is the noise suppressed? Submit this wavefile as well as your code.

P2: Motor Imagery [5 points]

1. `eeg.mat` has the training and testing samples as well as their labels. Use them to replicate my BCI classification experiments in Module 5 (not the entire lecture, but from S3 to S8 and S37). But, instead of PCA dimension reduction, we're going to use locality sensitive hashing to extract binary features. Also, instead of naïve Bayes, we'll do kNN classification.

2. For the kNN classification, for every test example you have to find the K nearest neighbors from 112 training samples. You're lucky. This kNN search is not a big deal, because your training set is tiny. However, as your professor I have to help you guys be prepared for your future big data projects at your fancy new job after your graduation (don't forget me). So, I'm giving you this homework assignment that will help you come up with a speed-up technique for the kNN search. It's not that complicated.

3. Come up with a random projection matrix $\boldsymbol{A} \in \mathbb{R}^{L \times M}$, where M denotes the number of dimensions of your data samples (5 rows of your magnitude STFT, vectorized), 255[1], and L is the new dimension after this random projection. So, you're replacing PCA with this random projection. Note that L doesn't always have to be smaller than M. Although $\boldsymbol{A}$ is a matrix with random values, you should make sure that the row vectors (the projection vectors) should be unit vectors, i.e. $||\boldsymbol{A}_{i,:}||_2 = 1$. Make sure of it.

4. Do the random projection and take the sign of the result (element-wise), so that your random projection produces a bunch of +1's and -1's:

$$\boldsymbol{Y} = \text{sign}(\boldsymbol{A}\boldsymbol{X}_{1:M,:}) \tag{3}$$

5. Use this $\boldsymbol{A}$ matrix and the sign function wrapper to convert your 255 dimensional TRAINING samples into L-dimensional bipolar binary hash codes. Ditto for the TEST samples (use the same $\boldsymbol{A}$ matrix). If you're not comfortable with bipolar binaries, you can turn them into 0-1 binaries by replacing -1's with 0's. It's up to you.

6. Do your kNN classification using these binary version of your data samples. Note that you can compare the test bit string with the bit strings of the training data by using the Hamming distance, instead of the Eucleadian distance between the original real-valued vectors. You know your computer prefers binary values, right? This way you can speed up the comparison,

[1]Note that this number comes from the calculation [# channels] × [# frames] × [# subbands], while # frames can slightly vary depending on your STFT implementation. Mine resulted in $3 \times 17 \times 5 = 255$

and eventually the kNN search (although you might not be able to see the actual speed-up due to the too small dataset size and your script language that doesn't exploit binary variables). Report your classification results with different choice of K and L. I think a table of accuracies will be great. You don't have to submit all the intermediate plots. What I need to see is the accuracies and your source code.

7. Compare your results with mine in M5 S37. You're classifier is based on binary variables and bitwise operations, so you must have expected a big performance drop. How do you like it eventually? Report your classification accuracy. Submit your code and your discussion.

P3: Multidimensional Scaling [5 points]

1. `MDS_pdist.mat` holds a 996×996 square matrix, which holds squared pairwise Euclidean distances that I constructed from a set of data points. Of course I won't share the original data samples with you. Instead, you'll write a multidimensional scaling code for this so that you can recover the original map out of $\boldsymbol{L}$. If your MDS routine is successful, you should be able to see what the original map was. The original map was in the 2D space, so you may want to see two largest eigenvectors. Report the map you recovered in the form of a scatter plot.

2. Note that the MDS result can be rotated and shifted.