Machine Learning for Signals (CS 545)
Homework 1

Due date: Sep. 22, 23:59 PM (Central)

Instructions

- Please don't submit your homework, although we encourage you to finish it by the due date.
- The solution will be distributed on the due date.
- Please still do your homework, as the quizzes and exams will be based on the homework assignments.
- Unless mentioned otherwise, you are expected to implement all the functions instead of using existing toolboxes, e.g., from `sklearn`.
- Avoid using toolboxes.

P1: Picky Eater

1. Prof. K's daughter was a picky eater when she was one year old—she only ate blueberries, strawberries, and dried yogurt drops. So, it was a little awkward for him to pack her lunch when she first started to attend the daycare because her lunchbox contained only those three kinds of food.
2. Prof. K has always been a good father. He was worried about his daughter's health, so when he picked her up at the end of the day, he would ask the daycare teachers how many blueberries, strawberries, and yogurt drops she ate for lunch.
3. For the first week at the daycare, the toddler ate a considerable amount of food, which Prof. K recorded in a spreadsheet:

Days	M	T	W	T	F
Blueberries	20	25	18	15	28
Strawberries	5	6	3	2	7
Yogurt drops	10	12	8	15	5

Table 1: A picky eater's lunch record

4. You know, her eating behavior follows the multinomial distribution with three parameters: $p_{\text{blueberry}}$, $p_{\text{strawberry}}$, and p_{yogurt}. Find them out using the maximum likelihood technique (hint: don't worry about the different numbers for different days. You can just think of the table as an observation for the entire week by summing up the numbers. You don't have to derive the MLE equation, either, because I already did it for you in L03.).

5. Prof. K and his wife were quite surprised, because the one-year-old usually doesn't eat that many blueberries at home. There is something called the "daycare effect"—a kid tends to be encouraged to eat more if there are a bunch of other kids around eating at the same time. Even if the daycare effect is true, they suspect that their kid might not have eaten all of the blueberries. For example, the toddler could have thrown the food on the floor like she does at home, which might have caused overcounting.

Figure 1: Food on the floor

6. Meanwhile, ever since she started eating those three kinds of food, the parents have recorded the amount she had. It turned out that the baby had eaten 1,080 blueberries, 450 strawberries, and 900 yogurt drops, in the past month before she entered the daycare. This is their *a priori* knowledge.

7. Use the *a priori* knowledge to better estimate the three parameters. Feel free to come up with a derivation for MAP estimation. You'll need to do the optimization on the log of the posterior probability (L01-S13) using the Lagrange multipliers (L03-S21). But I don't require that. I actually don't encourage this way at least for this problem.

8. The thing is, from your intuition, you can easily come up with a MAP estimation for this problem. Try to figure this out without math. The goal of this question is to give you insight about pseudo-counts.

P2: Central Limit Theorem [3 points]

1. You're taking Prof. K's MLSP course. It turned out that you really like his lecture, because he tells a lot of jokes. So, you decided to record it without his permission.

2. When you play the recording at home, you realize that the recording was contaminated by some annoying classmates around your smartphone. They were too talkative in the middle of the class, making Prof. K's deep and beautiful voice inaudible.

3. You know you'll learn about source separation soon in class, but you don't know how to do it yet. Instead, you learned that there is another one who secretly recorded the lecture that

day. You wanted to know if the other recording is any better than yours. For example, you know you'd prefer the one with fewer interfering sources.

4. `x1.wav` and `x2.wav` are two recordings made in the MLSP class by two devices. Even though both of them are capturing the same part of the lecture, the numbers of interfering sources are different from each other. For your information, I'm also providing `s.wav`, which is Prof. K's ground-truth speech. Listen to `s.wav`, `x1.wav` and `x2.wav`. Obviously, `s.wav` must be the best among the three, but can you tell which one is cleaner between `x1.wav` and `x2.wav` (i.e. with fewer interfering sources)?

5. You can use the Central Limit Theorem (CLT) to figure out which one is with more sources. First, we assume that the samples from a given source are from a random variable with unknown probabilistic distribution. We assume that it's not a Gaussian distribution. For example, draw a histogram from the samples of `s.wav`. Does it look like a Gaussian? To me it's too spiky, so maybe not. Although you cannot draw histograms of the other interfering sources (because you don't know them), let's assume they are from a non-Gaussian distribution, too.

6. According to CLT, the sum of random variables gets closer to a Gaussian distribution. If there are more random variables (i.e. sources) added up, the mixture of them will be more Gaussian-like. Therefore, you can measure the Gaussian-likeness of the sample distributions of `x1.wav` and `x2.wav` to see which one is with more sources.

7. You will use a non-Gausianity metric, called "Kurtosis," for this. Long story short, kurtosis is defined as follows if the distribution of the random variable x is normalized (i.e. zero mean and unit variance):
$$\mathcal{K}(x) = E(x^4) - 3 \tag{1}$$

8. In Python, it's convenient to import `soundfile`[1], which has a load function as follows:

```
import soundfile as sf
x1, rate = sf.read('x1.wav')
```

9. Standardize both signals by subtracting their sample means and by dividing by their sample standard deviation.

10. Calculate the kurtosis. Which one is less Gaussian-like according to the kurtosis values? Draw histograms of the two signals. Can you eyeball the graphs to see which one is less Gaussian-like?

11. As a sanity check, compare them with the histogram of `s.wav` as well. Can you see the mixture signals' histograms are more Gaussian-like than the ground-truth source'? Calculate the kurtosis of `s.wav` as well for a comparison.

12. Note that this is just an anecdotal example. Estimating the number of audio sources from a mixture is a difficult problem.

[1]https://pypi.org/project/soundfile/

P3: Power Iteration

1. Write your own power iteration routine that calculates the eigenvector one by one (L02-S11).
2. Load `flute.npy` using `np.load`. It is a matrix representation of the two musical notes played by a flute. Plot this matrix by using a color map of your choice to show the intensity of all the horizontal bars (they're something called harmonics). Your plot will look like the one in L02-S07 (not identical though).
3. The input matrix, $\boldsymbol{X}$, has 142 column vectors, each of which has 128 frequency elements. Compute the sample covariance matrix out of them. It must be a 128×128 symmetric matrix.
4. Estimate two eigenvectors from this by using your power iteration routine (See L02-S12 to recall how to calculate multiple eigenvectors one-by-one). Plot your eigenvectors. These must look similar to the two column vectors in the middle in L02-S07.
5. Now you know the representative spectra for the two notes. How would you recover their temporal activation? They will be two row vectors for the two notes, respectively (the third matrix with two row vectors in L02-S07). You need to think how to calculate these activation vectors in equations. Plot the activation (row) vectors.
6. Another alternative approach would be to calculate the covariance matrix out of $\boldsymbol{X}^\top$, which will give you a 143×143 matrix. By doing so, you can say that you have 128 observed samples, each of which has 143 dimensions. Perform power iteration twice to get the two eigenvectors. They should correspond to the temporal activations you calculated in the previous question, but this time you got them in a different way. How would you get the representative spectra this time?
7. Out of the two approaches, i.e. doing eigendecomposition on the original data matrix and its transposed version, which do you prefer? You should be able to explain your preference.

   (Hint: you can justify your preference in terms of the "reconstruction error.")

P4: De-beeper

1. `x.wav` is a speech signal contaminated by a beep sound. As I haven't taught you guys how to properly do speech enhancement yet, you're not supposed to know a machine learning-based solution to this problem (don't worry I'll cover it soon). Instead, you did learn how to do STFT, so I want you to at least manually erase the beep sound from this signal to recover the clean speech source. If you have a perfect pitch, you might be able to know the pitch of the beep, but I assume that you don't. That's why you have to see the spectrogram to find out the beep frequency.
2. First off, create a DFT matrix $\boldsymbol{F}$ using the first equation in M02-S12. You'll of course create a $N \times N$ complex matrix, but if you see its real and imaginary versions separately, you'll see something like the ones in M02-S14 (the ones in the slide are 20×20). Determine your N, which is the frame size shown in M02-S17. For example, since the signal's sampling rate is 16kHz, if your N is 1600, your frequency resolution (the range a Fourier coefficient covers) will be 10Hz. If your N is large, you'll get finer frequency resolution and vice versa. Feel free to try out a few different choices.

3. Prepare your data matrix $\boldsymbol{X}$. You extract the first frame of N samples from the input signal, and apply a Hann window (or any other windows that can overlap-and-add to one). What that means is that from the definition of Hann window[2], you create a window of size N and element-wise multiply the window and your N audio samples. Place it as your first column vector of the data matrix $\boldsymbol{X}$. Move by $N/2$ samples. Extract another frame of N samples and apply the window. This goes to the second column vector of $\boldsymbol{X}$. Do it for your third frame (which should start from $(N+1)$'th sample, and so on. Since you moved just by the half of the frame size, your frames are overlapping each other by 50%.

4. Apply the DFT matrix to your data matrix, i.e. $\boldsymbol{F}\boldsymbol{X}$. This is your spectrogram with complex values. See how it looks like (by taking magnitudes and plotting). Locate two thin horizontal lines. They are from the beep sound. Note that due to the conjugacy your spectrogram is mirrored vertically. The bottom half is a mirrored version of the top half in terms of their magnitudes, although their imaginary parts are with a different sign (complex conjugate). The spectrograms you've seen in class are the top half of a spectrogram, because the bottom half has no useful information (except for the flipped phase). This is why you see two beeper lines in your spectrogram. Anyway, locate them, and make the entire row zero.

5. Apply the inverse DFT matrix, which you can also create by using the equation in L4 S12. Let's call this $\boldsymbol{F}^*$. Since it's the inverse transform, $\boldsymbol{F}^*\boldsymbol{F} \approx I$ (you can check it, although the off diagonals might be a very small number rather than zero). You apply this matrix to your spectrogram, which is free from the beep tones, to get back to the recovered version of your data matrix, $\hat{\boldsymbol{X}}$. In theory this should give you a real-valued matrix, but you'll still see some imaginary parts with a very small value. Ignore them by just taking the real part. Reverse the procedure in 1.3 to get the time domain signal. Basically it must be a procedure that transpose every column vector of $\hat{\boldsymbol{X}}$ and overlap-and-add the right half of t-th row vector with the left half of the $(t+1)$-th row vector and so on. Listen to the signal to check if the beep tones are gone.

[2] I allow you to use a pre-defined function to compute this window, but I encourage you to go ahead and implement one based on the simple equation: https://en.wikipedia.org/wiki/Window_function#Hann_and_Hamming_windows