Homework 1

Deadline Monday, October 11, 23h59

Submission Instructions

Please follow these instructions closely. We will deduct points for not sticking to the template. Solutions should be uploaded to Blackboard. HW1_template.zip contains all the data and necessary templates required to solve all the problems.

Problem 1: Projections

The problem_1 directory has a data/ folder, and one sub-folder problem_1/results, to include the results. The main directory also has a template file called problem_1/run_problem_1.m. Write your code into this script. There are instructions within the script on how to save your outputs. The scripts must be written so that we can simply run run_problem_1.m from MATLAB within the directory, without any additional commands or adding any additional file to the directories. If we cannot run the program, we cannot score you.

Note: All your results should go in the problem_1/results/ folder. Instructions for each part:

(a) Part 1: Analysis by individual note

1. Your solution will give you one score line or transcription per note. Place all the scores in a single matrix and save this as a single file named problem1a.dat
2. For this part, the synthesized music must be saved as problem1b_synthesis.wav within the results directory.

(b) Part 2: Joint analysis using all notes

1. Your solution will give you single score matrix. Save this as a single file named problem1a.dat
2. For this part, the synthesized music must be saved as problem1b_synthesis.wav within the results directory.

Problem 2:

We shall use the same data (Polyushka.wav) music used in the previous question. There is a ”problem_2/run_problem_2.m” file that should output only the following:

(a) Plot of error as a function of iteration number for $\eta = 0.0001, 0.001, 0.01, 0.05$.

(b) Final weight matrix after 250 iterations

(c) (Optional) Approximated music saved as polyushka_approx.wav

(d) All outputs should be saved in the problem_2/results folder. Save the plots as PNG named as error_eta_#.png where # should correspond to the eta value used and the final weight matrix as final_weight.dat

(e) The derivation for $\frac{\partial E}{\partial W}$ should be saved in problem_2_writeup.pdf file in the results folder.

Note: Ensure graphs have a title and are labelled properly. Points will be deducted if your code doesn't match the required output format mentioned above. Your code should not output anything else.

Problem 1: Linear Algebra

Please note all the files needed for this exercise can be found in homework template HW1_template/problem_1/.

The recording segment Polyushka.wav, played on harmonica, has been downloaded from YouTube with permission from the Artist.

Note.tar is a tar file, which contains a set of notes from a harmonica. You are required to transcribe the music. For transcription you must determine how each of the notes is played to compose the music.

Matlab_instructions file inside the homework package contains information how to convert each note into a spectral vector, and the entire music to a spectrogram matrix.

I. Analysis by individual note:

a) For each note individually, compute the contribution of that note to the entire music. Mathematically, if N_i is the vector representing the i^{th} note, and M the music matrix, find the row vector W_i such that $N_i W_i \approx M$. Return the transcription of each note.

b) Recompose the music by "playing" each note according to the transcription you just found. Mathematically, compute $\hat{M} = \sum_i N_i W_i$ and invert the result to produce music. To invert it to a music signal follow the instructions given in the matlab notes. Return the recomposed music. Comment about how the recomposed music compares to the original signal.

II. Joint analysis using all notes:

a) Compute the contribution of all notes to the entire music jointly. Mathematically, if $N = [N_1, N_2, ...]$ is the *notes matrix* where the individual columns are the notes, find the matrix W such that $NW \approx M$. The i^{th} row of W is the transcription of the i^{th} note.

b) Recompose the music by "playing" each note according to the transcription you just found in IIa. Mathematically, compute $\hat{M} = NW$, and invert the result to produce

music. Return the recomposed music. Comment about how the recomposed music compares to the original signal. Is the recomposed music identical to the music you constructed in Ib. Explain your finding.

Problem 2: optimization and non-negative decomposition

Please note all the files needed for this exercise can be found in homework template HW1_template/problem_2/.

Simple projection of music magnitude spectrograms (which are nonnegative) onto a set of notes will result in *negative* weights for some notes. To explain, let **M** be the (magnitude) spectrogram of the music. It is a matrix of size ***DxT*** , where ***D*** is the size of the Fourier transform and ***T*** is the number of spectral vectors in the signal. Let ***N*** be a matrix of notes. Each column of ***N*** is the magnitude spectral vector for one note. ***N*** has size ***DxK***, where ***K*** is the number of notes.

Conventional projection of ***M*** onto the notes ***N*** computes the approximation

$$\hat{M} = NW$$

such that $\left\|M - \hat{M}\right\|_F^2 = \sum_{i,j} (M_{i.j} - \hat{M}_{i,j})^2$ is minimized. Here $\left\|M - \hat{M}\right\|_F$ is known as the Frobenius norm of $M - \hat{M}$. $M_{i.j}$ is the $(i,j)^{th}$ entry of M and $\hat{M}_{i,j}$ is similarly the $(i,j)^{th}$ entry of $\hat{M}$. Please note the definition of the Frobenius norm; we will use it later.

$\hat{M}$ is the projection of M onto N. W, of course, is given by $W = pinv(N)M$. W can be viewed as the transcription of M in terms of the notes in N. So, the j^{th} column of M, which we represent as M_j and is the spectrum in the j^{th} frame of the music, is approximated by the notes in N as

$$M_j = \sum_i N_i W_{i,j}$$

where N_i, the i^{th} column of N and represents the i^{th} note and $W_{i,j}$ is the weight assigned to the i^{th} note in composing the j^{th} frame of the music.

The problem is that in this computation, we will frequently find $W_{i,j}$ values to be *negative*. In other words, this model requires you to *subtract* some notes — since $W_{i,j}N_i$ will have negative entries if $W_{i,j}$ is negative, this is equivalent to subtracting note the weighted note $\left|W_{i,j}\right|N_i$ in the

j^{th} frame. Clearly, this is an unreasonable operation intuitively; when we actually play music, we never *unplay* a note (which is what playing a negative note would be).

Also, $\hat{M}$ may have negative entries. In other words, our projection of M onto the notes in N can result in *negative* spectral magnitudes in some frequencies at certain times. Again, this is meaningless physically – spectral magnitudes cannot, by definition, be negative.

In this homework problem we will try to fix this anomaly.

We will do this by computing the approximation $\hat{M} = NW$ with the constraint that the entries of must W always be greater than or equal to 0, i.e. they must be non-negative. To do so we will use a simple gradient descent algorithm, which minimizes the error $\|M - NW\|_F^2$ subject to the constraint that all entries in W are non-negative.

I. Computing a derivative

We define the following error function

$$E = \frac{1}{DT}\|M - NW\|_F^2$$

where D is the number of dimensions (rows) in M, and T is the number of vectors (frames) in M.

Derive the formula for $\frac{\partial E}{\partial W}$.

II. A Non-Negative Projection

We define the following gradient descent rule to estimate W. It is an iterative estimate. Let W^0 be the initial estimate of W and W^n the estimate after n iterations. We use the following *projected gradient* update rule

$$\hat{W}^{n+1} = W^n - \eta \frac{\partial E}{\partial W} | W^n$$
$$W^{n+1} = \max(\hat{W}^{n+1}, 0)$$

where $\frac{\partial E}{\partial W}|W^n$ is the derivative of E with respect to W computed at $W = W^n$, and $\max(\hat{W}^{n+1}, 0)$ is a *component-wise* flooring operation that sets all negative entries in $\hat{W}^{n+1}$ to 0.

In effect, our *feasible set* for values of W are $W \geq 0$, where the symbol $\geq$ indicates that *every* element of W must be greater than or equal to 0. The algorithm performs a conventional gradient descent update, and projects any solutions that fall outside the feasible set back onto the feasible set, through the *max* operation.

Implement the above algorithm. Initialize W to a matrix of all 1s. Run the algorithm for η values (0.0001, 0.001, 0.01, 0.1) . Run 250 iterations in each case. Plot E as a function of iteration number n. Return this plot and the final matrix W. Also show a plot of best error E as a function of η.

III. Recreating the music (No points for this one).

For the best η (which resulted in the lowest error) recreate the music using this transcription as $\hat{M} = NW$. Resynthesize the music from $\hat{M}$. What does it sound like? You may return the resynthesized music to impress us (although we won't score you on it).

Problem 3: Source Separation using ICA

In this problem, you have to implement your own version of ICA and apply it to source separation. You are given 2 audio recordings, sample1.wav and sample2.wav. These can be found in the directory HW1_template/problem_3.

These recording were generated mixing two different audios. Your objective is to reconstruct the original sounds using ICA. Do the following steps:

1. Implement your own version of ICA based on FOBI (Freeing Fourth Moments) method that we discussed in class. Write a function ICA which receives as input a *2 × N* matrix and outputs a *2 × N* matrix where its rows are the extracted independent components. **Submit your code.**

2. Read the file sample1.wav and extract the audio signal **s1**. This should be a vector with 132,203 components. Read the file sample2.wav obtaining the signal **s2**. Both **s1** and **s2** have the same size. Transpose and concatenate these signals generating a matrix ***M*** with 2 rows and 132,203 columns. Apply the function ICA on the matrix ***M*** and using the

Matlab function audiowrite, save the components generated as source1.wav and source2.wav respectively. Don't forget ICA does not consider scale factors, so you may need to boost or decrease the resulting signal. **Submit files source1.wav and source2.wav.**

3. If $\boldsymbol{H}$ is a $2 \times N$ where its rows correspond to the output of ICA, then we can consider that

$$\boldsymbol{M = AH}$$

In this case $\boldsymbol{A}$ is the mixing matrix which produces our observation $\boldsymbol{M}$. Compute the 2 × 2 matrix for this case. **Submit *A* as mixing matrix.csv.**

Problem 4 : Signal Separation using Non Negative Matrix Factorization

In this problem you will use NMF to separate a mixed signal into its component signals. Specifically, your goal is to separate music and speech sounds from a signal consisting of both.

The basic idea is that you will use NMF to first learn bases for speech and music. Then you will use the learned bases to separate out music and speech from a recording consisting of both.

Do the following processing steps.

1. In the directory HW1_template/problem_4 you will find speech.wav file. Read the speech signal and compute its spectrogram. Use the stft function followed by magnitude computation, as before;

```
[s, fs] = audioread('filename');
spectrogram = stft(s',2048,256,0,hann(2048))
M = abs(spectrogram)
```

2. Do the same for music.wav file.

If we call $\boldsymbol{M_m}$ and $\boldsymbol{M_s}$ the magnitude spectrum of the music and speech file respectively, then the dimensionality of both matrices should be 1025 × 977.

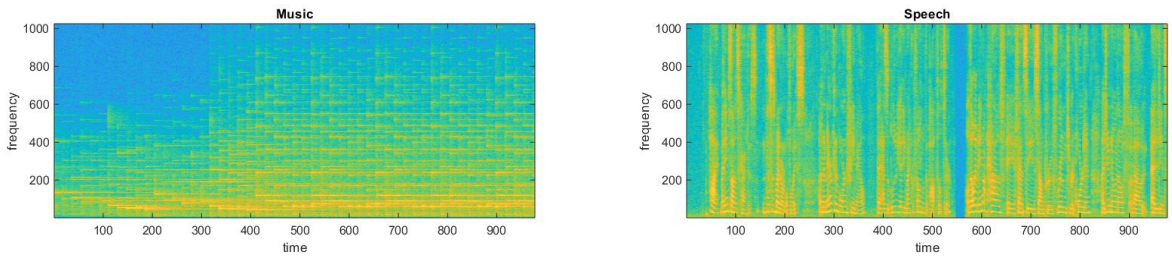

Figure 1: Magnitude spectrum (in log scale) of files `music.wav` and `speech.wav`

Now using the spectrograms for music M_m and speech M_s you are first going to learn bases, B_m and B_s for music and speech respectively. Therefore, we given the non-negative matrices M_m and Ms, we need to find the non-negative matrices B_m, W_m, B_s and W_s such that

$$M_m \approx B_m W_m$$
$$M_s \approx B_s W_s$$

NMF Estimation: Learning Bases

In this homework, given a non-negative matrix $M \in R^{p \times q}$, we will consider the algorithm to estimate the non-negative matrices $B \in R^{p \times k}$ and $W \in R^{k \times q}$ that attempts to minimize the ***KL*** divergence between ***M*** and ***BW***. The KL divergence between matrix ***A*** and matrix ***B*** is given by

$$D(A \,||\, B) \quad = \quad \sum_{ij} \left(A_{ij} \log \left(\frac{A_{ij}}{B_{ij}} \right) - A_{ij} + B_{ij} \right)$$

Then, the algorithm that minimizes ***D(M||BW)*** is the following:

• Initialize ***B*** and ***W*** randomly.

• Iteratively update ***B*** and ***W*** using the following rule:

$$B = B \otimes \frac{\left(\frac{M}{BW}\right) W^\top}{\mathbf{1}_{p \times q} W^\top}$$
$$W = W \otimes \frac{B^\top \left(\frac{M}{BW}\right)}{B^\top \mathbf{1}_{p \times q}}$$

where $\otimes$ denotes component-wise matrix multiplication, - denotes the component-wise matrix division, and $\mathbf{1}_{p \times q}$ is the $p \times q$ matrix which each of its component is 1.

1- Write the function NMF train which receives as inputs:

• ***M***: a p × q non-negative matrix.
• ***B_init***: a p × k non-negative matrix. This matrix is the initial value of matrix B.
• ***W_init***: a k × q non-negative matrix. This matrix is the initial value of matrix W .
• ***n_iter***: a positive integer value. It determines the number of iterations of the algorithm.

Based on the algorithm previously described, the outputs of this function must be:

- ***B***: a p × k non-negative matrix.
- ***W***: a k × q non-negative matrix.

Submit your code.

2- In the directory HW1_template/problem_4 you can find the files Bm init.csv and Wm init.csv. These files are matrices which must be used as initial condition to run your function NMF train over the magnitude spectrum of the music signal M_m. Consider values for n_iter in {0, 50, 100, 150, 200, 250}. Run your function using these parameters and plot $\boldsymbol{D(M_m||B_mW_m)}$ as function of n_iter. Attach this plot to your report and write the value of $\boldsymbol{D(M_m||B_mW_m)}$ for n_iter = 250.

3- Similarly, you can also find in the directory HW1_template/problem_4 the files Bs_init.csv and Ws_init.csv. As before, these files are matrices which must be used as initial condition to run your function NMF train over the magnitude spectrum of the speech signal M_s. Repeat the previous task your this signal considering the same setting. Attach your plot to the report and write the value of $\boldsymbol{D(M_s||B_sW_s)}$ for n_iter = 250.

Signal Separation

Now, using the bases you learned for speech and music, we will separate a signal which is a mix between a different speech signal and a music signal.

In the directory HW1_template/problem_4 you can find a file named mixed.wav. Compute its magnitude spectrum, that we will call Mmixed. The following figure visualize this matrix.

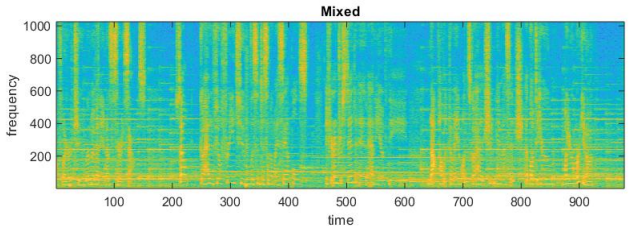

Figure 2: Magnitude spectrum (in log scale) of file `mixed.wav`

Our model considers that the bases we learned in the previous part are good enough to represent this audio file. More precisely, in this case, given $\boldsymbol{M_{mixed}}$, ***Bs*** and ***Bm***, we need to learn the weights ***W*** such that

$$M_{mixed} \approx [B_s\ B_m]$$

In this equation, $[B_s\ B_m]$ represents the concatenation of the bases you already learned; so if Mmixed is a $p \times q$ matrix, $[B_s\ B_m]$ is a $p \times 2k$ matrix. Moreover, the matrix W gives us the contribution of each element in this new set of bases. To learn W we can use the previous algorithm, but without updating the bases B (we already know B).

Write the function separate signals which receives as inputs:
• **M_mixed**: a $p \times q$ non-negative matrix.
• **B_speech**: a $p \times k$ non-negative matrix. This matrix is the bases you have to represent speech.

• **B_music**: a $p \times k$ non-negative matrix. This matrix is the bases you have to represent music.
• **n_iter**: a positive integer value. It determines the number of iterations of the algorithm.

The output of this function must be:
• **M_speech_rec**: a $p \times q$ non-negative matrix. It is the recovered speech magnitude spectrum.

• **M_music_rec**: a $p \times q$ non-negative matrix. It is the recovered music magnitude spectrum.

Submit your code

1. Using the bases you learned from the previous problem when you considered n_iter = 250, apply your function separate signals using the magnitude spectrum of the mixed signal and taking n_iter = 500. Submit your output as *M* speech rec.csv and M music rec.csv.
2. Now using the phase for the mixed signal along with reconstructed spectrograms, reconstruct time domain music and speech signals. You did this in previous exercise as well. You can use the stft function again to do this. Save the generated signals as speech rec.wav and music rec.wav. Submit these two files.