Machine Learning for Signal Processing
Homework 1

Bhiksha Raj - Abelino Jimenez, Anurag Kumar, Yixuan Zhang, Davy Uwizera

Section 1 and section 2 are questions of homework1. Section 3 is matlab instruction which can help you with resolving problem in section 2. Section 4 is a submission guide.

1 Linear Algebra

1.1 Rotational Matrices

1. A rotation in 3-D space(whose coordinates we will call X, Y, Z) is characterized by three angles. We will characterize them as a rotation around the x-axis, a rotation around the y-axis, a rotation around the z-axis.
   Derive the rotation matrix R_1 that transforms a vector $[x, y, z]^T$ to a new vector $[\hat{x}, \hat{y}, \hat{z}]^T$ by rotating it counterclockwize by angle θ around the x-axis, then an angle δ around the y-axis, and finally an angle ϕ around the z-axis.
   Derive the rotation matrix R_2 that transforms a vector $[x, y, z]^T$ to a new vector $[\hat{x}, \hat{y}, \hat{z}]^T$ by rotating it counterclockwize by an angle δ around the y-axis, then an angle θ around the x-axis, and finally an angle ϕ around the z-axis.

2. Confirm that $R_1 R_1^T = R_2 R_2^T = I$

1.2 Lengths of vectors

Suppose we have three matrixes, A, B and C. A is a square 5×5 matrix, B is a 4×5 matrix which can transform 5-dimensional vectors into 4-dimensional ones, C is a 6×5 matrix which can transform 5-dimensional vectors into 6-dimensional ones.

$$A = \begin{bmatrix} 3 & 3 & 13 & 8 & 9 \\ 1 & 6 & 10 & 2 & 4 \\ 9 & 7 & 6 & 9 & 5 \\ 14 & 15 & 3 & 4 & 10 \\ 11 & 3 & 7 & 6 & 4 \end{bmatrix} \quad B = \begin{bmatrix} 13 & 9 & 13 & 7 & 7 \\ 15 & 2 & 4 & 5 & 2 \\ 11 & 14 & 9 & 3 & 9 \\ 6 & 14 & 1 & 3 & 8 \end{bmatrix} \quad C = \begin{bmatrix} 11 & 5 & 11 & 10 & 23 \\ 11 & 8 & 15 & 12 & 21 \\ 10 & 10 & 8 & 7 & 8 \\ 1 & 7 & 5 & 2 & 4 \\ 2 & 13 & 2 & 4 & 13 \\ 1 & 24 & 3 & 9 & 11 \end{bmatrix}$$

1. A 5×1 vector v of length 1 is transformed by A as $u_1 = Av$, by B as $u_2 = Bv$ and by C as $u_3 = Cv$. What is the longest length that u_1 can be? What about u_2, u_3?

2. What is the shortest possible length of u_1? What about u_2, u_3?

3. The nullspace of a matrix A is the set of vector such that after applying the matrix A we obtain 0. Formally
   $$\mathcal{N}(A) = \{x \,:\, Ax = 0\}$$
   Find out dimensionality of nullspace of A, B, C, and A^TA, B^TB, C^TC.

1.3 Inverse

Let A be a $n \times n$ matrix, such that $A_{ij} = 1\ \forall i, j$.

1. If $n = 2017$, determine the value of $\alpha \in \mathbb{R}$ such that
   $$\left(I - \frac{1}{2021}A\right)^{-1} = I + \alpha A$$
   where I is the identity matrix of order n.

2. If $e \in \mathbb{R}^n$ such that $e_i = 1 \forall i$. Determine the norm of
   $$\left(I - \frac{1}{n}A\right)e$$
   Is $I - \frac{1}{n}A$ invertible?

2 Music Transcription Problem

2.1 Projection

Recording of "Polyushka Poly" is played on the harmonica, the file name of the recording is `polyushka.wav`, which can be found in folder `hw1materials` . It has been downloaded from YouTube with permission from the artist.

A set of notes from a harmonica can be found in folder `hw1materials/note15`. You are required to transcribe the music. For transcription you must determine how each of the notes is played to compose the music.

You can use the matlab instructions given in ***section 3*** to convert each note into a spectral vector, and the entire music to a spectrogram matrix.

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

2. Recompose the music by "playing" each note according to the transcription you just found in last question. Set all negative elements in W to zero. Compute $\hat{M} = \mathbf{N}W$, and invert the result to produce music. Return the recomposed music(although we will not score you on music).

2.2 Optimization and non-negative decomposition

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

Conventional projection of $\mathbf{M}$ onto the notes $\mathbf{N}$ computes the approximation

$$\hat{\mathbf{M}} = \mathbf{NW}$$

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

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

$$\mathbf{M_j} = \sum_i \mathbf{N}_i \mathbf{W_{i,j}}$$

where $\mathbf{N_i}$, the i^{th} column of $\mathbf{N}$ and represents the i^{th} note and $\mathbf{W_{i,j}}$ 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 $\mathbf{W}_{i,j}$ values to be negative. In other words, this model requires you to subtract some notes - since $\mathbf{W}_{i,j}\mathbf{N}_i$ will have negative entries if $\mathbf{W}_{i,j}$ is negative, this is equivalent to subtracting note the weighted note $|\mathbf{W}_{i,j}|\mathbf{N}_i$ 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, $\hat{\mathbf{M}}$ may have negative entries. In other words, our projection of $\mathbf{M}$ onto the notes in $\mathbf{N}$ can result in negative spectral magnitudes in some frequencies at certain times. Again, this is meaningless physically – spectral magnitudes cannot, by definition, be negative.

We will do this by computing the approximation $\hat{\mathbf{M}} = \mathbf{NW}$ with the constraint that the entities of $\mathbf{W}$ must 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 $||\mathbf{M} - \mathbf{NW}||_F^2$ subject to the constraint that all entries in $\mathbf{W}$ are non-negative.

1. **Computing a Derivative**

   We define the following error function:

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

   where D is the number of dimensions (rows) in $\mathbf{M}$, and T is the number of vectors (frames) in $\mathbf{M}$.

   Derive the formula for $\frac{dE}{d\mathbf{W}}$.

2. **A Non-Negative Projection**

   We define the following gradient descent rule to estimate $\mathbf{W}$. It is an iterative estimate. Let $\mathbf{W}^0$ be the initial estimate of $\mathbf{W}$ and $\mathbf{W}^n$ the estimate after n iterations.
   We use the following project gradient update rule

$$\hat{\mathbf{W}}^{n+1} = \mathbf{W}^n - \eta \frac{dE}{d\mathbf{W}}|_{\mathbf{W}^n}$$

$$\mathbf{W}^{n+1} = \max(\hat{\mathbf{W}}^{n+1}, 0)$$

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

   In effect, our *feasible set* for values of $\mathbf{W}$ are $\mathbf{W} \succeq 0$, where the symbol $\succeq$ indicates that *every* element of $\mathbf{W}$ 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 $\mathbf{W}$ to a matrix of all 0s. Run the algorithm for η values $(100, 1000, 10000, 100000)$. Run 1000 iterations in each case. Plot E as a function of iteration number n. Return this plot and the final matrix $\mathbf{W}$. Also show a plot of best error E as a function of η.

3. **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 $\hat{\mathbf{M}} = \mathbf{NW}$. 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).

2.3 Linear Transformation

Here we have two audios of a simple rhythm of song "silent night" played by piano (audio A) and classical guitar (audio B), and one audio of a simple rhythm of song "little star" played by piano (audio C).

All audios are given in `hw1materials/Audio`.

Based on information you have, how would you get the classical guitar version (audio D) of song "little star"? Give the final magnitude spectrogram of audio D and convert it to audio (.wav file).

Hint: Learn a linear transformation that converts the magnitude of the spectrogram for a frame in audio A to the magnitude of spectrogram for the corresponding frame in audio B. Then apply it to audio C. You may find all given audios have two channels, in this case, just work with the first channel of audio. Don't forget that the magnitude of a complex number is always a positive value. Use 1024 as sample window, instead of 2048.

Use the instruction of **section 3** to compute spectrograms and reconstruct the signal.

3 Matlab Instructions

3.1 Computing the spectrogram

Get the matlab file `stft.m` in folder `hw1materials`.
`stft.m` computes the complex spectrogram of a signal.
You can read a wav file into matlab as follows.

```
[s,fs] = audioread('filename');
s = resample(s,16000,fs);
```

Above, we resample the signal to a standard sampling rate for convenience. Next, we can compute the complex short-time Fourier transform of the signal, and the magnitude spectrogram from it, as follows. Here we use 2048 sample windows, which correspond to 64ms analysis windows. Adjacent frames are shifted by 256 samples, so we get 64 frames/second of signal.

To compute the spectrogram of a recording, e.g. the music, perform the following.

```
spectrum = stft(s',2048,256,0,hann(2048));
music = abs(spectrum);
sphase = spectrum./(abs(spectrum)+eps);
```

This will result in a 1025-dimensional (rows) spectrogram, with 64 frames(columns) per second of signal.

Note that we are also storing the phase of the original signal. We will need it later for reconstructing the signal. We explain how to do this later. The **eps** in this formula ensures that the denominator does not go to zero.

You can compute the spectra for the notes as follows. The following script reads the directory and computes spectra for all notes in it.

```
notefolder = 'notes15/';
listname = dir([notesfolder '*.wav']);
notes = [ ];
for k = 1:length(listname)
    [s,fs] = audioread([notesfolder listname(k).name]);
    s = s(:,1);
    s = resample(s,16000,fs);
    spectrum = stft(s',2048,256,0,hann(2048));
    %Find the central frame
    middle = ceil(size(spectrum,2)/2);
    note = abs(spectrum(:,middle));
    %Clean up everything more than 40db below the peak
    note(find(note < max(note(:))/100)) = 0;
    note = note/norm(note);
    %normalize the note to unit length
    notes = [notes,note];
end
```

The "`notes`" matrix will have as many columns as there are notes (15 in our data). Each column represents a note. The notes will each be represented by a 1025 dimensional column vector.

3.2 Reconstructing a signal from a spectrogram

The recordings of the complete music can be read just as you read the notes. To convert it to a spectrogram, do the following. Let **reconstructedmagnitude** be the reconstructed magnitude spectrogram from which you want to compute a signal. In our homework we get many variants of this. To recover the signal we will use the **sphase** we computed earlier from the original signal.

```
reconstructedsignal = stft(reconstructedmagnitude.*sphase,2048,256,0,hann(2048));
```

4 Submission Guide

4.1 Linear Algebra

Please submit all results for this part in "Report_YourAndrewID.pdf" in you submission

4.2 Music Transcription Problem

2.1 Projection

The "2.1" subdirectory has a template file called Run_problem21.m. Write your code into this script. There are instructions within the script on how to save your outputs. We repeat the instructions below.

The scripts must be written so that we can simply run Run_problem21.m from matlab within the directory, without any additional commands. If we cannot run the program, we cannot score you.

1. Your solution will give you single "score" matrix $\mathbf{W}$. Save this as a single file named "problem211.dat"

2. For this part, the synthesized music must be saved as "problem212_synthesis.wav" within the "results" directory.

2.2 Optimization and non-negative decomposition

1. For this part, return the solution in "Report_YourAndrewID.pdf" in your submission.

2. For this part, store the final $\mathbf{W}$ for each η value in a text file called "problem222_eta_xxx.dat" where xxx is actual η value. E.g. for $\eta = 0.01$, xxx will be "0.01". Plot the plot of E(total error) vs. iterations for each η in a file called "problem222_eta_xxx_errorplot.png", where xxx is the η value. Also plot the η vs. E as a bar plot and save it in "problem222_eta_vs_E.png".

3. For this part, save the synthesized music in the "results" directory in a file called "polyushaka_syn.wav"

2.3 Linear Transformation

Your solution should include final magnitude spectrogram of audio D. Save this as a single file named "problem23.mat". Also save audio D as "problem23_audio.wav" within the "result" directory.