Homework 1
Linear Algebra

CMU 11-755/18-797: Machine Learning for Signal Processing (Fall 2019)

OUT: September 5th, 2018
DUE: **September 19th, 11:59 PM**

START HERE: Instructions

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 3.4"). Second, write your solution independently: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only.
- **Submitting your work:** Assignments should be submitted as PDFs using Canvas unless explicitly stated otherwise. Please submit all results as `report_{YourAndrewID}.pdf` in you submission. **Each derivation/proof should be completed on a separate page**. Submissions can be handwritten, but should be labeled and clearly legible. Else, submissions can be written in LaTeX. Upon submission, label each question using the template provided by Canvas. Please refer to Piazza for detailed instruction for joining Canvas and submitting your homework.
- **Programming**: All programming portions of the assignments should be submitted to Canvas as well. Please `zip` or `tar` all the code and output files together, and submit the compressed file together with the pdf report. We will not be using this for autograding, meaning you may use any language which you like to submit.

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, and 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 counterclockwise 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 counterclockwise 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 matrices, 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.

$$A = \begin{bmatrix} 4 & -8 & 1 & -6 & -7 \\ -6 & -10 & -4 & 2 & 8 \\ 1 & -2 & -2 & -7 & 5 \\ 1 & 6 & -8 & -3 & 1 \\ 6 & 5 & 4 & -7 & -3 \end{bmatrix} \quad B = \begin{bmatrix} 9 & -5 & 2 & 10 & 4 \\ -7 & -1 & -3 & -10 & 10 \\ -4 & 3 & -5 & -5 & 4 \\ -6 & 2 & -1 & -5 & -2 \end{bmatrix} \quad C = \begin{bmatrix} -6 & -4 & -5 & -9 & -4 \\ -3 & -7 & -5 & -8 & -10 \\ -8 & -8 & 5 & 7 & -9 \\ -2 & -1 & 4 & -4 & -6 \\ 7 & -10 & -6 & 9 & -7 \\ -2 & 8 & -6 & 6 & 6 \end{bmatrix}$$

1. A 5×1 vector $\mathbf{v}$ of length 1 ($\|\mathbf{v}\| = 1$) is transformed by A as $\mathbf{u}_1 = A\mathbf{v}$, by B as $\mathbf{u}_2 = B\mathbf{v}$ and by C as $\mathbf{u}_3 = C\mathbf{v}$. What is the longest length that $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$ can be?
2. What is the shortest possible length of $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$?

1.3 Inverse through Singular Value Decomposition (SVD)

Consider the matrix A as follows:

$$A = \begin{bmatrix} 1 & 5 & 7 \\ 2 & -1 & 3 \\ 0 & 1 & -5 \end{bmatrix}$$

1. Compute the SVD of the matrix A, i.e., find the values σ_1, σ_2 and σ_3 (with $\sigma_1 \geq \sigma_2 \geq \sigma_3$) and the vectors $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{u}_3$, $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ such that $\|\mathbf{u}_i\| = \|\mathbf{v}_i\| = 1$ for all i, and

$$A = \sum_{i=1}^{3} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$$

2. Compute the matrix B defined by the following equation,

$$B = \sum_{i=1}^{3} \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^\top$$

3. Confirm and demonstrate that $AB = I$.

2 Music Transcription

2.1 Projection

The song “Polyushka Poly” is played on the harmonica, the file name of the audio recording is `polyushka.wav`, which can be found in the 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 need to compute the spectrogram of the music file using your language/toolbox of choice, such as python (Librosa) or Matlab. First, read and load the audio file at 16000 Hz sample rate.

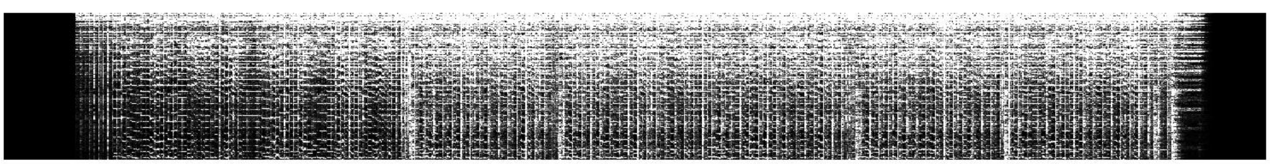
If you are using Python, you can use Librosa to load the wav file as follows (we also recommend using the `numpy` package for matrix operations below if you use python):

```
import librosa
audio, sr = librosa.load(filename, sr = 16000)
```

Next, we can compute the complex Short-Time Fourier Transform (STFT) of the signal s and its magnitude spectrogram. Use 2048 sample windows, which correspond to 64 ms analysis windows; overlap/hop length of 256 samples to 64 frames by second of signal. Different toolboxes should provide similar spectrograms. If you are using the provided Matlab script, you can use the following command (eps = 2.2204e-16):

```
spectrogram = librosa.stft(audio, n_fft=2048, hop_length=256, center=False, win_length=2048)
M = abs(spectrogram)
phase = spectrogram/(M + 2.2204e-16)
```

In this case, $\mathbf{M}$ represents the music file and should be a matrix of 1025×8869, where the rows correspond to the frequencies and the columns to time. A visualization (Librosa - specshow) of this matrix (spectrogram) should look like the following figure.



To represent notes, you also need to compute the spectrogram of each note file. However, unlike the music file, we need to represent the matrix just as a one column vector. Hence, we can choose only one vector, or compute the mean of the matrix across time, etc. In this example, we select the middle column:

```
# n is the spectrogram of the note
import math
middle = n[:, int(math.ceil(n.shape[1]/2))]
```

To focus on the most relevant frequencies, we can clean up and normalize the note as follows:

```
middle[middle < (max(middle)/100)] = 0
```

Finally, you need to normalize this vector as follows,

```
middle = middle/np.linalg.norm(middle) # import numpy as np
```

1. Compute the joint contribution of all notes to the entire music.
   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 $\mathbf{NW} \approx \mathbf{M}$, or that produce a small error $||\mathbf{M} - \mathbf{NW}||_F^2$. The i_{th} row of $\mathbf{W}$ is the transcription of the i_{th} note. Submit the matrix $\mathbf{W}$ as `problem1.csv` together with your code.
2. Recompose the music by "playing" each note according to the transcription you found in last question. Set all negative elements in W to zero and compute $\hat{M} = \mathbf{N}W$. Report the value of $||\mathbf{M} - \hat{\mathbf{M}}||_F^2 = \sum_{i,j}(\mathbf{M}_{i,j} - \hat{\mathbf{M}}_{i,j})^2$ and submit the recomposed music named as `resythensized_proj.wav` file.

   To recover the signal from the reconstructed spectrogram $\hat{\mathbf{M}}$ we need to use the phase matrix we computed earlier from the original signal. Combine both and compute the Inverse-STFT to obtain a vector and then write them into a wav file. To compute the STFT and then write the wav file you can use the following python command:

   ```
   signal_hat = librosa.istft(M_hat*phase, hop_length=256, center=False, win_length=2048)
   librosa.output.write_wav("resynthensized_proj.wav", signal_hat, sr=16000)
   ```

2.2 Optimization and non-negative decomposition

A projection of the music magnitude spectrogram (which are non-negative) onto a set of notes will result in negative weights for some notes. To explain this, let $\mathbf{M}$ be the (magnitude) spectrogram of the music, which 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 of size $D \times K$, where K is the number of notes and each column D is the magnitude spectral vector of one note.

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

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

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

$\hat{\mathbf{M}}$ is the projection of $\mathbf{M}$ onto $\mathbf{N}$. Moreover, $\mathbf{W}$ is given by $\mathbf{W} = pinv(\mathbf{N})\mathbf{M}$ and $\mathbf{W}$ 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 is the spectrum in the j^{th} frame of the music, which are approximated by the notes in $\mathbf{N}$ as follows:

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

where $\mathbf{N_i}$, the i^{th} column of $\mathbf{N}$ represents the i^{th} note and $\mathbf{W_{i,j}}$ is the (contribution) 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. 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 due to the values in $\mathbf{W}$. 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.

Hence, we will compute the approximation $\hat{\mathbf{M}} = \mathbf{N}\mathbf{W}$ 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 $||\mathbf{M} - \mathbf{N}\mathbf{W}||_F^2$, subject to the constraint that all entries in $\mathbf{W}$ are non-negative.

1. **Computing a Derivative**

   We define the following error function:

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

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

   Derive and write down 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 for all ηs in a figure. Show this plot with some analysis in the separate page, and submit the best final matrix $\mathbf{W}$ (which resulted in the lowest error) named as `problem2W.csv` with the code.

3. **Recreating the music**

   For the best η (which resulted in the lowest error) recreate the music using this transcription as $\hat{\mathbf{M}} = \mathbf{N}\mathbf{W}$. Resynthesize the music from $\hat{M}$. What does it sound like? Submit the resynthesized music named as `resynthesized_nnproj.wav` with the code.

3 Style transfer using a 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`.

From these files, you can obtain the spectrogram $\mathbf{M}_A$, $\mathbf{M}_B$ and $\mathbf{M}_C$. Your objective is to find the spectrogram of the classical guitar version of the song "little star" ($\mathbf{M}_D$).

You may find all given audios have two channels. To compute the spectrogram from the given files just work with the first channel and use the same instructions of the previous problem, but considering 1024 as sample window, instead of 2048.

In this problem, we assume that style can be transferred using a linear transformation. Formally, we need to find the matrix $\mathbf{T}$ such that

$$\mathbf{T}\mathbf{M}_A \approx \mathbf{M}_B$$

1. Write a code to determine matrix $\mathbf{T}$ and report the value of $\|\mathbf{T}\mathbf{M}_A - \mathbf{M}_B\|_F^2$.
   Submit the matrix $\mathbf{T}$ as `problem3t.csv` and your code
2. Our model considers that $\mathbf{T}$ can transfer style from piano music to classical guitar music. Applying $\mathbf{T}$ on audio C should give us a estimation of "little star" played by guitar, getting an estimation of $\mathbf{M}_D$. Using this matrix and phase matrix of C, synthesize an audio signal.
   Submit your code, your estimation of the matrix $\mathbf{M}_D$ as `problem3md.csv` and the sythensized audio named as `problem3.wav`