CMRM Homework Assignment No. 2 (HWA2)

December 9, 2024

1 Description

The main topic of this homework is timbre transfer. The goal is to implement an algorithm that is able to transfer the timbre of a particular source to a target audio signal. In order to solve such a task, you will have to rely just on signal processing methods, in particular on Nonnegative Matrix Factorization (NMF). Use the Jupyter Notebook named `Homework2.ipynb` to implement the code, and explain in a report, step by step, what you have implemented.

1.1 Question 1

Once the zip file named "audio" has been extracted, you can take a look at the files at your disposals. You will have to process four targets using three different sources, which are split among two subfolders:

- `audio/sources` - It contains the three sources from which you are required to learn the timbre, namely `Bees_Buzzing.wav`, `Wind_Blowing.wav`, and `Chainsaw_Sawing.wav`;
- `audio/targets` - It contains the four targets to which you are required to apply the learned timbre, namely `Jingle_Bells_Boogie.wav`, `Have_Yourself.wav`, `Blue_Christmas.wav`, and `White_Christmas.wav`;

For Questions 1.1, 1.2, 1.3, you will have to consider just `Bees_Buzzing.wav` as source and `Jingle_Bells_Boogie.wav` as target. This is to let you understand the task and develop the pipeline. In Question 1.5, you will have to process all the given files.

As a first step, define the path of the different folders and create the folder `audio/results` where you will save the processed tracks. Then, define:

- a list named `sources` containing the names of the different sources;
- a list named `targets` containing the names of the different targets;
- a variable named `source` containing the name of the source we want to process (i.e., `Bees_Buzzing.wav`) exploiting the list `sources`;
- a variable named `target` containing the name of the target we want to process (i.e., `Jingle_Bells_Boogie.wav`) exploiting the list `targets`.

You can now load the two waveforms using the usual `librosa.load` function with a sampling frequency `Fs=22050` Hz. Then, define the window length `N_length` and the hop size `H_size`. In addition, consider the following steps:

- compute the STFT of `source` and assign it to the variable `X_source`.
- extract the magnitude `Y_source` and plot it.
- compute the STFT of `target` and assign it to the variable `X_target`.

- extract the magnitude `Y_target` and plot it.
- compute the time resolution `time_res` and the frequency resolution `freq_res`.

You will find guidelines in `Homework2.ipynb`. Please, add labels to axes. Remember to provide comments in the report.

1.2 Question 2

Now that you have loaded the audio files, you are ready to accomplish timbre transfer by means of NMF. Nowadays, timbre transfer is accomplished making use of very large and sophisticated neural networks, but standard signal processing techniques can be used to obtain a "quick and dirty" solution, and NMF is among them. Given a magnitude spectrum $V \in \mathbb{R}_{\geq 0}^{K\times N}$ (thus, a nonnegative matrix) and a $P \in \mathbb{N}$, classical NMF derives two nonnegative matrices $W \in \mathbb{R}_{\geq 0}^{K\times P}$ and $H \in \mathbb{R}_{\geq 0}^{P\times N}$ such that a distance $D(V, WH)$ is minimized, meaning that $V \approx WH$ holds true. In this context, the columns of W are typically called *templates* and the rows of H are typically called *activations.* To compute a factorization, one typically initializes W and H with random values and updates them iteratively using multiplicative rules. On the other hand, one can constrain the templates, the activations, or both to obtain a semantically more meaningful factorization. In particular, in this notebook you are going to constrain and fix W by exploiting the magnitude STFT of the source track. As a consequence, W will contain information about the frequencies involved in the source track, while H will be learned by means of the NMF iterative procedure in order to match the magnitude STFT of the target audio track, giving as a result an STFT with a magnitude that resembles both target and source.

In order to solve this question, you have first to install `scikit-learn` by running `pip install sklearn` and `tqdm` by running `pip install tqdm`. Then, perform the following steps:

- initialize the activation matrix `H0` with random values using the function `np.random.rand` providing as input the length (of the time dimension) of `X_source` and the length (of the time dimension) of `X_target`.
- initialize the template matrix `W0` by normalizing the `Y_source` matrix as

$$W_0 = Y_{\text{source}} \oslash \left(\sum_{\imath} Y_{\text{source},\imath\jmath} + \varepsilon \right) , \tag{1}$$

  where $\oslash$ indicates the Hadamard division and $\varepsilon > 0$ is a small value to avoid division by zero. Pay attention to the second term: it is a way of writing a summation over the dimension $\imath$ (i.e., the first dimension), which will give a vector as a result.

- define the variable `Xs` which is the normalized version of `X_source` as

$$X_{\text{s}} = X_{\text{source}} \oslash \left(\sum_{\imath} Y_{\text{source},\imath\jmath} + \varepsilon \right) . \tag{2}$$

- perform NMF on the target magnitude STFT `Y_target` by using the function `nmf` that you find inside the `nmf.py` file. Set `num_iter=50`, `init_W=W0`, `init_H=H0`, `fix_W=True`, `cont_polyphony=10`, `cont_length=7`, `cont_grid=5`, and `cont_sparsen=(1, 7)`. The results will be the matrices `W` and `H`.
- compute the magnitude STFT approximation by means of `W` and `H`. Then, print the 2-norm of the error (difference) between true and approximated magnitude STFTs. Comment on the result: is it a low value or a high value? Why do we obtain such a value?

Now, it is time to add some plots in order to analyze the results. In particular:

- plot the activation matrix before and after NMF, i.e., both `H0` and `H`. Please, add a colorbar and labels on both axes (be careful and reason about the shape of the matrix). What is the information provided by this matrix? What can you tell from the plots? During the course, we have introduced different post-processing steps. What action can you perform on `H` in order to make the content more visible? Perform it and plot the result.

- define the function `visualize_nmf` which takes in the magnitude STFT (in our case of `Y_target`), the activation matrix, the template matrix, the time resolution and the compression factor `gamma`; you find a prototype in the notebook. The function has to:
  1. plot a compressed version of matrix `H`;
  2. plot a compressed version of matrix `W`. Constrain the y-axis between 0 and 2000;
  3. plot a compressed version of the approximation obtained using `W` and `H`. Constrain the y-axis between 0 and 2000;
  4. plot a compressed version of the magnitude STFT of `Y_target`. Constrain the y-axis between 0 and 2000.

  Please, add labels to the axes (pay attention to the correct naming), colorbars, and titles to plots. You may decide to apply different compression factors as a function of gamma to the different plots (e.g., twice gamma, half gamma, etc.).

Which parameters do you think influence the most the factorization? What can you tell by analyzing the plots concerning approximation and original magnitude STFT? Do they look similar? If not, can you explain why they are different? (*Hint: it may help answering also the question "Why do we fix the template matrix and learn only the activation matrix?"*)

1.3 Question 3

Once learned the activation matrix `H`, we can use it to perform timber transfer. In particular, we will apply it to the complex-domain STFT such that also a phase information is present. Then, in order to compute the time-domain waveform, we will consider two alternative ways: i) the first concerns the naive Inverse STFT (ISTFT); ii) the second concerns the use of a more sophisticated algorithm known under the name of Griffin-Lim algorithm. We must add that, in fact, the reconstruction is not straightforward as we do not know the correct phase to be used for the processed audio. The Griffin-Lim algorithm has been proposed exactly to tackle situations of the sort. It is an iterative technique used in signal processing to reconstruct a time-domain signal from its magnitude, when the phase information is missing. It is commonly applied in applications like speech synthesis. The algorithm tries to minimize the reconstruction error by iteratively improving the phase while keeping the magnitude fixed. Although it does not guarantee perfect reconstruction, it is extremely useful in practice for many use cases. Please, address carefully the following points:

- apply the learned activation matrix to the complex-domain STFT of the target in order to enforce the source timbre. In practice, the complex-domain STFT of the processed audio `Y_tt` is obtained considering
$$Y_{\mathrm{tt}} = X_{\mathrm{s}} H \,. \tag{3}$$
- re-synthesize the audio by means of the Inverse STFT employing the function `librosa.istft` and the arguments used for computing the STFT in Question 1.2. Store the result in the variable `y_istft`.
- re-synthesize the audio by means of the Griffin-Lim algorithm employing the function `librosa.griffinlim` and the arguments used for computing the STFT in Question 1.2. Store the result in the variable `y_gl`.
- save the audio waveform as a `.wav` file. You can use the function `sf.write`. Please, name the obtained file according to "`<target_name>_<source_name>.wav`".
- plot the `y_istft`. Compute the phase out of the FFT and plot the first 10 seconds. Add labels to the axes.
- plot the `y_gl`. Compute the phase out of the FFT and plot the first 10 seconds. Add labels to the axes. Is the phase equal to the previous one? What can you tell by comparing the two plots?
- listen to the results by employing the `ipd.Audio` function (you can also use to play the source and the target).

Provide comments in the report on the performed steps. Can you spot (and hear) some differences between the two reconstructions? Can you tell which is the best? Does the target show the source timbre? What are the parameters that in your opinion influence the most the timbre transfer?

1.4 Question 4

You have developed the whole pipeline for accomplishing timbre transfer. In this question, you will have to pack everything together inside one single function, such that you can easily apply the pipeline to all the other sources and targets. Create the function named `timber_transfer`, which takes in the target waveform `t`, the source waveform `s`, the sampling frequency `f_s`, the STFT hop size `hop_size`, the STFT window length `win_length`, the re-synthesis method `resynth`, and a boolean variable `plot`. It returns `y`, i.e., the audio waveform of the target with the source timbre. The function has to follow the steps performed in the previous points. To sum up:

- compute the magnitude STFT of both the target and the source.
- initialize the activation matrix.
- initialize the template matrix.
- normalize the STFT of the source.
- perform nonnegative matrix factorization on the target magnitude STFT.
- compute the complex-domain STFT of the processed track by applying the learned activation matrix to the normalized STFT of the source.
- if `plot==True`, use the function `visualize_nmf` to plot the matrices involved in the NMF.
- if `resynth==istft`, use the ISTFT to reconstruct the time-domain signal, while if `resynth==gl`, use Griffin-Lim instead.
- save the audio file using `sf.write` and the title convention introduced in the previous question.
- listen to the results by using the `ipd.Audio` function.

1.5 Question 5

Now that you have implemented the function `timbre_transfer`, you are ready to process all the other sources and targets. Set up the whole pipeline in order to process all the sources defined by the list `sources` and the targets defined by the list `targets`. For each target and for each source, load the audio wav files, apply the function `timbre_transfer`, and listen to the results. Then, address the following points:

- Is the algorithm able to transfer the timbre of the source to the target for each source and target? What are, in your opinion, the best results? Why some target-source pairs provide worse results with respect to the others?
- Experiment with the values for `H_size` and `N_length`. Find the best pair of values for each target-source pair. Comment on the chosen values: why do we need to modify said parameters? Is it mostly due to the target or the source? (*Hint: you may argument on the time and frequency resolutions.)*
- Experiment with the NMF parameters now. Which is, in your opinion, the most relevant for the final result?

Are there other processing steps that you can think of to improve the results (you are not required to actually implement them, but you can if you wish)? Please, provide comments giving both harmonic and signal processing clues. Can you think of a way to check whether the timbre of two songs are similar? Try your idea on one of your processed tracks and provide comments in the report.

2 General Rules

There are two Homework Assignments (HWAs) for Computer Music Representations and Models (CMRM), each worth 20% of the final grade, and this is HWA2. HWAs can be done individually, or in groups of two students. Please, keep in mind that students of the same group will be given the same grade, irrespective of their contribution, therefore, please make sure that all members equally contribute to the assignment.

The maximum grade attainable is 30/30, but if the HWA2 is turned in by Sunday Jan 12 2025 (before midnight) you will get a bonus of 3 points. If you turn it in after that date but before the first "appello" (i.e., Jan 22 2025), you will get the full grade without bonus. If you turn it in after the first "appello", you will get a penalty of 5 points.

3 Files to be Delivered

You are required to deliver the following files:

1. a report, containing all answers to the questions and comments to the code. Include your surname/surnames in the title of the report (e.g., `Rossi.docx` or `Bianchi_Rossi.docx`). You can use whatever editor, even LaTeX, and, in general, you can provide a pdf file (rather than a doc file);
2. the filled `Homework2.ipynb` file. This is already divided into different sections and cells according to the questions that you are required to solve. In order to ease the solution, the notebook is provided with some guidelines in the form of comments. Rename the notebook with your surname/surnames (e.g., `Rossi.ipynb` or `Bianchi_Rossi.ipynb`). Please, add comments to the code, and plot or print all intermediate results. It is suggested to add titles, axis labels, and/or legends to the plots.
3. a folder containing the audio files generated via the processing explained above.

Zip the report and the notebook. Name the zip file using your surname/surnames (e.g., `Rossi.zip` or `Bianchi_Rossi.zip`). The zip must be turned in by Sunday Jan 12 2025 (before midnight). **Only one student for group must load the zip file on WeBeep**.