CMRM Homework Assignment No. 2 (HWA2)

December 12, 2023

1 Description

The main topic of this homework is music source separation applied to chorales. The goal is to implement an algorithm that is able to separate the canonical four voices used in chorales. 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 "data" has been extracted, you can take a look at the files at your disposals. You will have to process four chorales, whose information is split among three subfolders. In fact, inside the folder "chorales" you will find three subfolders:

- `chorales/midi` - It contains for each chorale the midi files (.mid) related to the four sources `soprano`, `alto`, `tenor`, and `bass`;
- `chorales/mix` - It contains the wav files of the four chorales (mixtures). These are the tracks to process for extracting the four sources. In particular, `chorale_1_mix` and `chorale_2_mix` are chorales BWV 360 and BWV 248-45 written by J. S. Bach, respectively, while `chorale_3_mix` and `chorale_4_mix` are two traditional Christmas songs, namely "Stille Nacht" and "Joy to the World."
- `chorales/wav` - It contains the target wav files for each source of the four chorales.

For Question 1.1, 1.2, 1.3, you will have to consider just chorale number 1. This is to let you understand the task and develop the pipeline. In Question 1.5, you will have to process all the given chorales.

As a first step, define the list `source_names = ['soprano', 'alto', 'tenor', 'bass']` containing the four canonical sources. Load the file `chorale_1_mix.wav` and plot its waveform. Then, address the following points:

- implement the function named `extract_annotations`, which takes in the midi directory `midi_dir`, the number of the chorale to process `number`, the list `source_names`, and the boolean variable `verbose`. You will find a prototype in the notebook. The function extracts the note annotations contained in the midi files. There are many different libraries that allow you to import and process midi files. Please, install `pretty_midi` by running `pip install pretty_midi` inside your terminal. The function must return the following lists:
  1. `start`: containing all the note start times in seconds. You can look at the `pretty_midi.Note.start` contained in the list `pretty_midi.instruments`, i.e., an attribute of the object created when using `pretty_midi.PrettyMidi`;
  2. `duration`: containing all the note durations in seconds. You can obtain this, e.g., by using `pretty_midi.Note.get_durations()`;
  3. `pitch`: containing all the note pitches;

4. `velocity`: containing all the note velocities;
5. `label`: containing the name of the source associated to all the notes;
6. `annotations:` containing all the lists defined till now in the form `[[start, duration, pitch, velocity, label]]`. Thus, one possible entry of such a list could look like `[0.0, 0.8, 64, 70, 'soprano']`.

Moreover, if `verbose=True`, plot the piano roll for all the four sources. You may use the `get_piano_roll()` method provided by `pretty_midi`.

- use the `extract_annotations` function with `verbose=True` on chorale 1.
- create a `pandas.DataFrame` starting from the `annotations` list. In order to create it, you must install `pandas` by running `pip install pandas`. Such a library is the to-go library for processing dataset metadata (thus, take the chance to learn how to use it). The DataFrame should have as column labels `start, duration, pitch, velocity, label`.
- print the DataFrame.

You will find guidelines in `Homework1.ipynb`. Please, add labels to axes. Remember to provide comments in the report.

1.2 Question 2

Now that you have loaded the audio file and have extracted annotations as far as note onsets and pitches are concerned, you are ready to accomplish separation by means of NMF. Source separation is a prolific field of research. Many different audio companies (such as Meta, Deezer, SONY, Apple, etc.) are putting effort into developing increasingly better algorithms. Nowadays, source separation is accomplished making use of very large and sophisticated neural networks, but many signal processing methodologies have been used in the past, 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 both, by imposing an explicit harmonic structure to W and temporal information to H. As a consequence, W will contain information about the pitches of the audio track, while H information where such pitches are present in the temporal dimension. Such pieces of information are contained within midi files: this is the reason why they are fundamental for accomplishing this specific type of NMF.

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

- compute the Short-Time Fourier Transform (STFT) of the audio file x, and apply logarithmic compression on the magnitude as follows:

$$V = \log\left(1 + |\,\mathrm{STFT}(x)|\right), \tag{1}$$

  where $\mathrm{STFT}(x)$ is the STFT of x.
- define the variable `freq_res` containing the frequency resolution and the variable `frame_res` containing the frame resolution.
- define two variables containing the two shapes of V, respectively.
- initialize the activation matrix which will be used as a starting point later on for NMF. You can use the function `initialize_activation` that you find inside the `nmf_utils.py` (please, import the whole file). Set `tol_note=[0.1, 0.5]` and `tol_onset=[0.2, 0.1]`. Name the output matrix as `H_init`.
- initialize the template matrix which will be used as a starting point later on for NMF. You can use the function `initialize_template` that you find inside the `nmf_utils.py`. Set `tol_pitch=0.05` and name the output matrix as `W_init`.

- instantiate the NMF model by using `sklearn.decomposition.NMF` with `n_components=H_init.shape[0]`, `solver='mu'`, `init='custom'`, `max_iter=1000`.
- perform NMF on the spectrogram V by using the `fit_transform` method. Set `H=H_init` and `W=W_init`. The result will be the template matrix. The activation matrix is, instead, equal to the attribute `components_` of the NMF model.
- Compute the spectrogram approximation. Then, print the 2-norm of the error (difference) between true and approximated spectrograms.

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

- plot the template matrix. Constrain the y-axis between 0 and 2000. Please, add a colorbar and labels on both axes (be careful and reason about the shape of the matrix). What does it resemble? Can you recognize a trend? What are the horizontal lines?
- plot the activation matrix. 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?
- create a figure with two subplots (1 row, 2 columns). In the first one, plot the approximated spectrogram, while in the second one plot the original spectrogram. Constrain the y-axis between 0 and 2000. Please, add a colorbar, labels on both axes, and titles.

What can you tell by analyzing the two plots? Do they look similar? On which parameters can you act to improve the representation (you do not have to do it though), or, equivalently, which parameters do you think influence the most the factorization?

1.3 Question 3

The NMF model is ready to be applied for source separation. In this question, you will create some spectral masks to be applied to the activation matrix in order to extract a first example of sources. Then, with the aim of removing some audible artifacts, in Question 1.4 you will refine the separation using another masking technique. Please, address carefully the following points:

- define the function `split_annotations` which takes in the `annotations` and provide as output `ann_sop, ann_alt, ann_ten, ann_bas`, i.e., the annotation data for soprano, alto, tenor, and bass, respectively.
- apply the function to the annotations.
- obtain the spectral masks for each source by running again the function `initialize_activation` and considering same arguments of Question 1.2 but the annotations which must be now those computed in the first point. Hence, you will have to run it four times, one for each of the sources (i.e., considering `ann_sop, ann_alt, ann_ten, ann_bas`).
- apply the spectral masks to the activation matrix H (computed in Question 1.2) in order to obtain the activations of each of the sources. For example, for the soprano case:

$$H_{\text{sop}} = H H_{\text{init, sop}} , \tag{2}$$

  where $H_{\text{init, sop}}$ is the spectral mask of the soprano source obtain in the previous point.
- four each source, create a figure with three subplots (1 row, 3 columns). In the first subplot, plot the template matrix computed in Question 1.2; in the second subplot, plot the source activation matrix obtained in the previous point; finally, in the third subplot, plot the reconstructed spectrogram. For instance, for the soprano case, this is obtained by computing WH_{sop}. Please, add titles to each subplot, labels on both axes, and constrain the frequencies between 0 and 2000.

Provide comments in the report on the performed steps. How do the source activation matrices look like? Are they compliant with what you expected (*hint:* reason about the pitch set typically associated to each source)? What can you tell about the reconstruction of the different sources?

1.4 Question 4

NMF-based models typically yield only a rough approximation of the original magnitude spectrogram (or even a logarithmically compressed version thereof), where spectral nuances or the actual magnitudes may not be captured well. Therefore, the audio components reconstructed in this way may contain a number of audible artifacts. Some of these artifacts may be removed or attenuated by considering another masking technique. Instead of directly using the source activation matrices, we use them to compute soft masks. For the soprano case, we obtain the soft mask by considering the following equation

$$M_{\text{sop}} = (WH_{\text{sop}}) \oslash (WH + \varepsilon)\,, \tag{3}$$

where $\oslash$ indicates the Hadamard division and $\varepsilon > 0$ is a small value to avoid division by zero. The soft mask is then applied directly to the original $\text{STFT}(x)$ in order to obtain the STFT of the source (thus retaining the original mixture phase). Perform the following steps:

- compute the soft mask for each source and apply it to $\text{STFT}(x)$.
- compute the Inverse STFT in order to obtain the source audio waveform in time-domain. You can use the `librosa.istft` function.
- plot both the original and the reconstructed waveforms. Please, add labels on both axes.
- listen to the results by using the `ipd.display` and the `ipd.Audio` functions. Create an `ipd.display` for each source and one for the mixture itself.

Provide comments in the report. How do you rate the separation? Is the method working fine for all the sources? Can you spot some differences? If yes, can you guess why the method has different performance according to the processed source?

1.5 Question 5

You have developed the whole pipeline for accomplishing source separation of chorales. In this question, you will have to pack everything together inside one single function, such that you can easily apply the pipeline to other input files. Create the function named `separate`, which takes in the audio waveform `x`, the `annotations`, the length of the FFT `N_fft`, and the hop size of the FFT `H_fft`. It returns `x_sop, x_alt, x_ten, x_bas`, i.e., the source waveforms of soprano, alto, tenor, and bass, respectively.

Set up the whole pipeline in order to process all the chorales defined by the list `number = ['1', '2', '3', '4']`. For each chorale, load the audio wav file, apply the function `separate`, and use `ipd.display` to listen to the results. Then, address the following points:

- Consider the wav files in the folder `chorales/wav`. Choose a data metric (even multiple metrics, if you want) that quantifies the separation performance by comparing the reconstructed signal to the target signal (please, be aware that targets may have different shapes).
- What is the Signal-to-Distortion (SDR) ratio? Why is it usually considered for source separation?
- Compute the SDR for all the sources (please, be aware that targets may have different shapes). You can use the function provided by `mir_eval` (you have to install it). How should such a metric be interpreted? Can you rely only on it for rating the performance of a source separation algorithm?

Do you hear leaks? If yes, can you tell why they are present? Do you think the separation pipeline gives good results for all the chorales? If no, can you tell why? Please, provide comments giving both harmonic and signal processing clues.

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 Friday Jan 12 2024 (before midnight) you will get a bonus of 3 points. If you turn it in after that date but before the second “appello” (i.e., Feb 1 2024), you will get the full grade without bonus. If you turn it in after the second “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 `Rossi_Bianchi.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 `Homework1.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 `Rossi_Bianchi.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.

Zip the report and the notebook. Name the zip file using your surname/surnames (e.g., `Rossi.zip` or `Rossi_Bianchi.zip`). The zip must be turned in by Friday Jan 12 2024 (before midnight). **Only one student for group must load the zip file on WeBeep**.