SOURCE SEPARATION FOR AUDIO-APPLICATIONS

Sebastian Ament[*] and Aarti Bagul [†]

Abstract. The goal of this project is to successfully implement and test an algorithm for audio source separation. We are going to use the Itakura-Saito divergence and an additional smoothness penalty function to construct a nonnegative matrix factorization (NMF) for a frequency spectrogram of a sound file. Further, we are going to use an online algorithm to reduce the memory and time complexity of calculating the NMF. This will allow us to efficiently test the algorithm on sound files with nearly arbitrary length.

1. Introduction:. Source separation involves estimating the signal produced by an individual source from a mixed signal produced from multiple sources. A classic example of this is the cocktail party problem, where there are a number of people talking simultaneously and a listener is trying to separate their individual voices and follow the discussion. It can also be used to separate the signals produced by individual musical instruments in an audio file. Another interesting application is medical imaging of the brain with magnetoencephalography (MEG)

2. Method:. Nonnegative matrix factorization using the Itakura-Saito divergence has been proven effective for audio source separation. The idea is the following. Given an input sound file, one can calculate frequency spectrograms for small time intervals throughout the length of the file. This leads to a matrix representation $V \in \mathbb{R}_+^{F \times N}$ of the input file, where F is the number of frequency bins and N is the number of spectrograms at different times.

Then, we want to find a factorization of the form

$$V \approx WH$$

where $W \in \mathbb{R}_+^{F \times K}$ and $H \in \mathbb{R}_+^{K \times N}$ are also both positive matrices. K is typically chosen such that $FK + KN << FN$, as to do dimensionality reduction on the data. W is usually called "dictionary" matrix while H is called "activation" matrix. The motivation behind this naming convention is that we are trying to create a "dictionary" of signals $\{\vec{w}_1, ..., \vec{w}_K\}$ such that for each time $t \in \{1, 2, ..., N\}$, we can find activation coefficients $h_{1t}, ..., h_{Kt}$ satisfying

$$\vec{v}_t = \sum_{k=1}^{K} h_{kt} \vec{w}_k$$

Now, as each $\vec{w}$ represents a spectrogram itself, its entries should be nonnegative. Further, the h's should to be nonnegative, because having the negative of a dictionary signal constitute a spectrogram is not consistent with our representation model of the signal. Additionally, allowing negative entries in either or both H and W does not work well in practice. The decomposition $V \approx WH$ is an expression of this idea for all frequency spectrograms simultaneously.

3. Mathematical Problem Formulation:. Above, we introduced the concept of a nonnegative matrix factorization. In order to make the NMF computationally tractable, it is

[*]New York University, New York, NY. E-mail: `sebastian.ament@nyu.edu`.
[†]New York University, New York, NY. E-mail: `aarti.bagul@edu`.

usually posed as an optimization problem. Most generally, we want to find $W \in \mathbb{R}_+^{F\times K}$ and $H \in \mathbb{R}_+^{K\times N}$ which minimize

$$D(V|WH) = \sum_{f=1}^{F}\sum_{n=1}^{N} d(V_{nf}|\,[WH]_{fn}) \tag{3.1}$$

for some penalty function d. Note that this minimization is subject to nonnegativity constraints of W and H. For source separation of audio files, the Iakura-Saito divergence $d_{IS}(y,x)$, given by

$$d_{IS}(y,x) := \sum_i \left(\frac{y_i}{x_i} - \log\left(\frac{y_i}{x_i}\right) - 1\right)$$

has proven effective. Most importantly, d_{IS} is scale-invariant, that is, $d_{IS}(\alpha x|\alpha y) = d_{IS}(x|y)$. Given an audio file with large dynamic ranges and a penalty function without scale-invariance, the optimization would neglect quiet passages for the sake of optimizing W and H to fit the loudest passages best. With scale-invariance however, we do not have this bias.

Furthermore, as we are applying the NMF to audio signals, it is reasonable to assume that the activation coefficients $\vec{h}_t$ should change continuously in time. To enforce smoothness of $\vec{h}_t$, we define

$$P_2(H) = \sum_{k=1}^{K}\sum_{t=2}^{N} d_{IS}(h_{k(t-1)}|h_{kt})$$

as a second penalty function. Using the IS divergence, P_2 punishes big changes in h_{kt}. Combining both penalty functions linearly, we get

$$C_2(W,H) = D_{IS}(V|WH) + \lambda P_2(H)$$

where $\lambda \in \mathbb{R}_+$ can be adjusted to give more or less weight to the smoothness of $\vec{h}_t$.

4. Algorithms. The algorithms used are:
Online Algorithm for IS-NMF

```
Input Training set, $W^{(0)}$, $A^{(0)}$, $B^{(0)}$, $\rho$, $\beta$, $\eta$, $\epsilon$
repeat
    t ← t+1
    draw point $v_t$ from training set.
    $h_t \leftarrow \arg\min_h d_{IS}(\epsilon + v_t, \epsilon + Wh)$
    $a^{(t)} \leftarrow (\frac{\epsilon+v_t}{(\epsilon+Wh_t)^2} h_t^T).W^2$
    $b^{(t)} \leftarrow \frac{1}{(\epsilon+Wh_t)^2} h_t^T$
    if $t \equiv 0[\beta]$
        $A^{(t)} \leftarrow A^{(t-\beta)} + \rho \sum_{s=t-\beta+1}^{t} a^{(s)}$
        $B^{(t)} \leftarrow B^{(t-\beta)} + \rho \sum_{s=t-\beta+1}^{t} b^{(s)}$
        $W^{(t)} \leftarrow \sqrt{\frac{A^{(t)}}{B^{(t)}}}$
        for k = 1...K
            $s \leftarrow \sum_f W_{fk}$,
            $W_{fk} \leftarrow W_{fk}/s$
            $A_{fk} \leftarrow A_{fk}/s$,
            $B_{fk} \leftarrow B_{fk}$ x $s$
        end for
    end if
until $\left\|W^{(t)} - W^{(t-1)}\right\|_F < \eta$
```

Smooth IS-NMF Algorithm

```
Input: nonnegative matrix $\mathbf{V}$
Output: nonnegative matrices $\mathbf{W}$ and $\mathbf{H}$
Initialize $\mathbf{W}$ and $\mathbf{H}$ with nonnegative values
Compute $\hat{\mathbf{V}} = \mathbf{WH}$
for $i = 1 : n_{iter}$ do
    // Update $\mathbf{H}$
    $\mathbf{G}^- = \mathbf{W}^T(\mathbf{V}.\hat{\mathbf{V}}^{-2}); \mathbf{G}^+ = \mathbf{W}^T(\hat{\mathbf{V}}^{-1})$
    Update $\mathbf{h}_1$
    for $n = 2{:}\ N$ - 1 do
    $\mathbf{h}_n^{(i)} = \sqrt{[g_n^- .(\mathbf{h}_n^{(i-1)})^2 + \lambda\mathbf{h}_{n-1}^{(i)}]/[g_n^+ + \lambda/\mathbf{h}_{n+1}^{(i-1)}]}$
    end for
    Update $\mathbf{h}_N$
    Compute $\hat{\mathbf{V}} = \mathbf{WH}$
    // Update $\mathbf{W}$
    $\mathbf{W} \leftarrow \mathbf{W}[(\hat{\mathbf{V}}^{-2}.\mathbf{V})\mathbf{H}^T]/[\hat{\mathbf{V}}^{-1}\mathbf{H}^T]$
    Compute $\hat{\mathbf{V}} = \mathbf{WH}$
    Normalize $\mathbf{W}$ and $\mathbf{H}$ together
end for
```

5. Dataset and Software:. We are planning to generate a dataset by generating multiple recordings of our voices and using Sound eXchange to manipulate these audio files. We are planning to use the pyFASST module (Python implementation of the Flexible Audio Source Spearation Toolbox (FASST)) for the source separation.

6. Baseline:. First, we made a recording of a female speaker counting from one to ten in English, and a recording of a male speaker counting from one to ten in German. We then mixed both recordings together to get one 14 second mono .wav file, with a sampling rate of 44,100. We then down-sampled the recording by a factor of 2. This step was not crucial for the performance of our baseline programs, as the input file was small already. However, it may be of higher importance for longer recordings which we are going to deal with later. Next, we created a power spectrogram of the down-sampled recording. We chose a window size of 100 samples (4.5 ms) for the short time Fourier transform and computed a spectrogram with 129 frequency bins and 5291 time frames. We attached a picture of the spectrogram at the end of this document.

To do the source separation, we used the nnmf method in Matlab with a multiplicative update algorithm, and with the number of iterations set to 10,000 and $K = 10$ (K as defined in 2.). We repeated this 100 times with random matrix initializations. We got a non-negative matrix factorization W,H with root mean squared error of 0.19.
We tried another approach using the FastICA module in scikit- learn.
We used the same audio file from before and used the python wave module to get information about the parameters of the audio signal. We used this to plot the audio signal and generate a spectrogram. We then used the FastICA module to separate the components of the signal and generate graphs of these components. The results were not promising and the graphs of the two components had substantial overlap. Then, we used PCA to first reduce the dimensionality and then applied ICA. This gave us better results than using just ICA.

7. Homework 4: Project. First, we would like to mention that, as we are going to work on blind source separation, this exercise might seem to be of limited direct use. However, an interesting idea for separation of instruments would be to precompute a centroids in a k-means clustering which represent indicative signals for different instruments, and use it to detect what instruments are present. For this exercise, we computed a "vocabulary" for our two voices using two training recordings. Subsequently, we used it to recognize our voices on test recordings. In the following, we describe the specific procedure.

Like in our baseline, we computed power spectrograms for our input files, where the frequency bins represent the features of the samples. We tested the optimal window size for the short-time Fourier transform using a grid search with dyadic increments. That is, our grid for the window size was 128, 256, 512, 1024, and 2048. However, it is critical to note that the length of the feature vector grows with the grid size, as a longer Fourier expansion has to be used to accurately portray the signal in each window. Additionally, we know that k-means does not perform well in very high dimensions. That is, the performance of k-means, especially in the worst case, is negatively affected by increasing the window size. On the other hand, decreasing the window size has a negative effect on the amount of detail that can be described by the resulting feature vector. As a consequence, we chose 256 as our optimal window size.

Regarding the optimal number of clusters for the k-means part of the approach, we made an interesting observation. As we only considered our two voices, the number of clusters to necessary to distinguish between our voices was relatively low. With only about 20 clusters, the difference between the histograms of our voices were already significant enough. However, this does not mean that the audio signal was represented well by these clusters. Indeed, the total distortion of the representation of the audio signals by only 20 clusters was expectedly large. Therefore, to generalize this approach, we will have to employ k-means with far larger number of clusters. For future investigations, we are considering using this approach to detect different instruments in a sound file. Once detected, we could use this information to initialize the matrix in the NMF algorithm using k-means representations of the present algorithms. We will report on that later on.

8. Homework 5: Project. We implemented the Hottopixx algorithm and tested it on the power spectrogram of two Jazz pieces, each about a minute long. The spectrogram had 129 frequency bins, and about $4 \cdot 10^4$ temporal bins. In order to speed up our experimental computations, we chose a uniform sample of about 10^3 columns of the spectrogram. We then ran several experiments with 10^3 epochs of the incremental gradient descent with primal stepsize $s_p = 10^{-1}$ and dual stepsize $s_d = 10^{-2}$, with varying input parameter r. Each run took approximately three minutes.

Interestingly, if we choose a small r, relative to $f = 129$, the algorithm does not converge to a solution to the NMF problem. The empirically smallest value of r that seems to work on the songs is 70. This suggests that the spectrogram data exhibits a high degree of linear independence. As a consequence, the data cannot be efficiently written as a linear combination of a small number of columns of the spectrogram. It is notable that the authors of the paper ran their benchmarks on synthetic datasets, which were created by a linear combination of no more than three random vectors (see paragraph 3 on page 8).

To visualize the results, we plotted rows of the activation matrix W and columns of the library matrix F in the accompanying notebook.

We also tried to run the author's implementation of hottopixx on our dataset, but failed to get the data in the right format to be accepted by the algorithm.

REFERENCES

[1] Augustin Lefevre, Francis Bach, Cedric Fevotte, *Online algorithms for Nonnegative Matrix Factorization with the Itakura-Saito divergence*,2011..

[2] Cedric Fevotte, *Majorization-Minimization Algorithm for Smooth Itakura-Saito Nonnegative Matrix Factorization*, CNRS LTCI; Telecom ParisTech Paris, France.

[3] Sound eXchange http://sox.sourceforge.net/

[4] pyFASST https://pypi.python.org/pypi/pyFASST