Audio Source Separation Using Non-Negative Matrix Factorization (NMF)

Zahra Benslimane · Follow

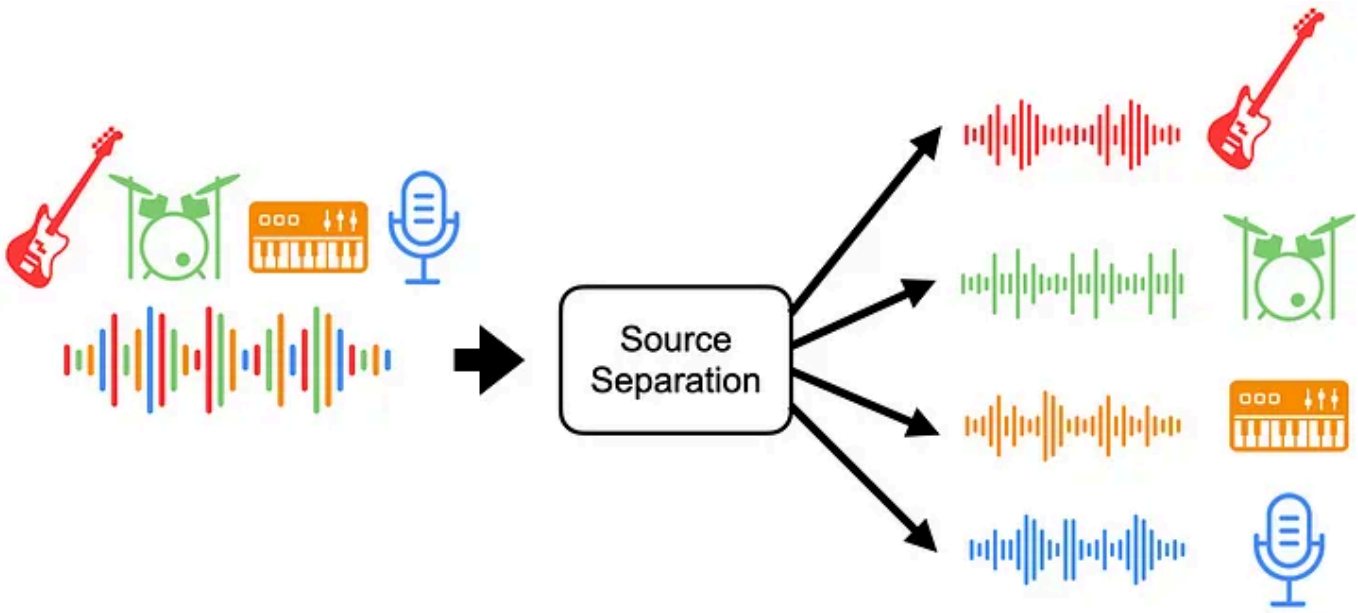
10 min read · Feb 1, 2023

01 Introduction :

This can be used in several applications, such as **music separation**, **speech separation** and **speech enhancement.**

In music separation, for example, we can separate the different instruments in a song, allowing the individual tracks to be edited or remixed. In speech separation, we separate different speakers in a recording, making it possible

to transcribe the speech of a single speaker. Lastly, speech enhancement can separate the vocals and the background noise or song.



music separation (source)

In this article, we will introduce a simple solving approach, and how it is used for audio source separation and we will implement a **python program** that allows us to **extract the played notes from a short piano sequence.**

02. Solving approach :

This problem, also known as the **Cocktail Party Problem**, is mostly tackled nowadays by deep learning approaches.

But, here, a more simpler and classical approach will be considered.

One of the most popular algorithms in signal processing used for solving this task is based on **Non-Negative Matrix Multiplication (NMF)**, a method that

was popularized by **Daniel D. Lee &H. Sebastian Seung** papers.

The NFM algorithm seeks to **reconstruct/ factorize a given data matrix V into two non-negative unknown matrices W and H**, such that the **product of the two matrices approximates the original matrix.**

$$V \approx WH = \hat{V}$$

What makes the NFM unique from other types of rank reduction algorithms, such as the PCA, is the additional **constraint** that **W and H must be strictly non-negative.**

$$W \geq 0$$
$$H \geq 0$$

In the context of sound source separation:

- The original **data matrix V** is typically the time-frequency representation of the audio signal, also known as a **spectrogram.**
- W represents a dictionary of **elementary recurring features** of the data.
- H contains the activations of the **different features.**

W is referred to as the **dictionary matrix** and H as the **activation matrix.**

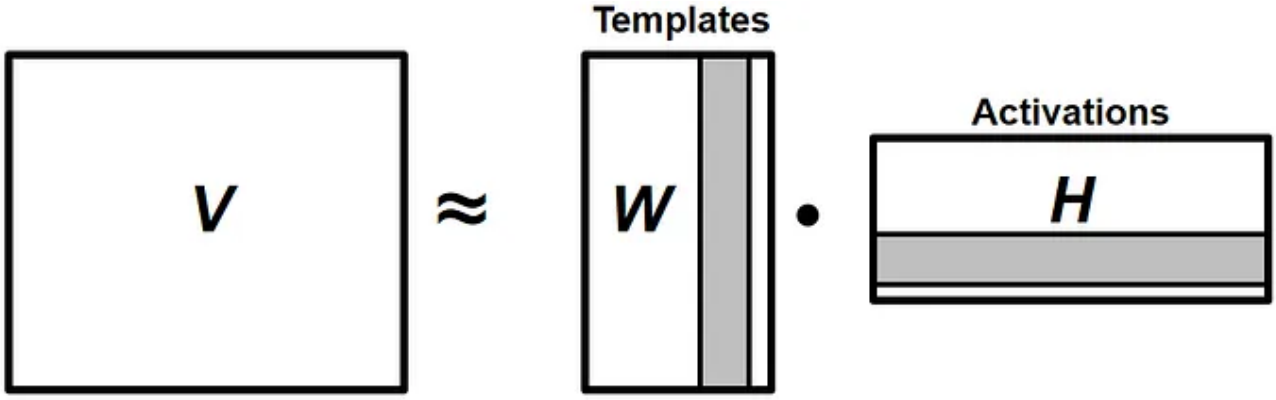


Figure 8.20b from [Müller, FMP, Springer 2015]

Given V of dimension K x N. W and H, are of dimensions K x *S and S* x N, respectively, where **S is the common dimension**, that can be thought of as the **number of sources we wish to extract** and can be referred to as the **rank of the factorization.**

To obtain a good reconstruction of the signal's spectrogram, the estimation of W and H are obtained by **optimizing a cost function** using the iterative updates of a **gradient descent.**

This means we will try to find the W and H that **minimizes the distance between the given data matrix V and the estimated one.**

$$\arg\min_{W,H\geq 0} = D(V|WH) = D(V|\hat{V}) = \sum_{i}^{K}\sum_{j}^{N} d(V_{i,j}||[WH]_{i,j})$$

03. Cost functions

We can use a variety of cost functions, but in this article, we will be seeing a family of statistical measures, called **β divergences** and defined as:

$$d_\beta(x|y) \overset{\text{def}}{=} \begin{cases} \frac{1}{\beta(\beta-1)}\left(x^\beta + (\beta-1)y^\beta - \beta x y^{\beta-1}\right), & \beta \in \mathbb{R}\backslash\{0,1\} \\ x\log\frac{x}{y} - x + y = d_{KL}(x|y), & \beta = 1 \\ \frac{x}{y} - \log\frac{x}{y} - 1 = d_{IS}(x|y), & \beta = 0 \end{cases}$$

β-divergence(source): The limit cases for β = 0 and β = 1 correspond to the Itakura-Saito (IS) and generalized Kullback-Leibler (KL) divergences, respectively.

The **quadratic cost** can also be found for β = 2

$$d_Q(x|y) = \frac{1}{2}(x-y)^2$$

Quardratic cost : β-divergence for β = 2 (source)

04. Gradient Descent Algorithm :

> *In mathematics,* ***gradient descent*** *is a first-order iterative optimization* ***algorithm*** *for finding a local minimum of a differentiable function. [source]*

For this task,

We denote θ a parameter corresponding to either W or H.

The update rule of the **gradient descent algorithm** looks like this:

$$\theta^{(i+1)} = \theta^{(i)} - \eta\nabla_\theta \beta Div$$

general gradient descent equation

for which θ(i+1) ≥ 0 in all iterations.

$$where : \theta^{(i+1)} \geq 0$$

Each gradient of the introduced cost functions can be written as a **difference between two non-negative functions.**

$$\nabla f(\theta) = [\nabla f(\theta)]^{+} - [\nabla f(\theta)]^{-}$$

Lee and Seung proposed in this paper to use adaptive learning rates to avoid subtraction that can produce negative elements.

The learning rate was set to :

$$\eta = \frac{\theta^{(i)}}{\nabla_H^{+} \beta_{Div}}$$

learning rate

When we replace the last two equations in the general gradient descent equation, we arrive at the given update rule :

$$\theta^{(i+1)} \leftarrow \theta^{(i)} \bigotimes \frac{[\nabla f(\theta)]^{-}}{[\nabla f(\theta)]^{+}}$$

Multiplicative update rule

This approach is taken such that the updates are multiplicative, where the correction value is equal to the ratio of the negative and positive parts of the derivative of the criterion.

The non-negativity of the multiplying update values ensures the non-negativity of the parameters W and H along the optimization iterations.

In the next section, we will compute the gradient of the cost function with respect to W and H, it can be very math intensive, so if not interested, feel free to skip it and move on to the next section (*06. Deriving Multiplicative Update Rules*) where we get the final update rules for the optimization algorithm.

05. Computing the gradients

Let's take the **generalized beta divergence** definition seen above:

$$\beta Div(V|WH) = \frac{1}{\beta(\beta-1)}(V^\beta + (\beta-1)[WH]^\beta - \beta V[WH]^{\beta-1}))$$
$$\beta Div(V_{i,j}|[WH]_{i,j}) = \frac{1}{\beta(\beta-1)}\sum_{ij}(V_{i,j}^\beta + (\beta-1)[WH]_{i,j}^\beta - \beta V_{i,j}[WH]_{i,j}^{\beta-1})$$

The gradient of this cost function can be computed **in terms of each parameter W or H at a time**, denoted as θ, as such :

$$\frac{\partial \beta Div(V|WH)}{\partial \theta_{p,q}} = \frac{1}{\beta(\beta-1)}\sum_{ij}((\beta-1)[WH]^{\beta-1}i,j\frac{\partial [WH]i,j}{\partial \theta_{p,q}} - \beta(\beta-1)V_{i,j}[WH]^{\beta-2}i,j\frac{\partial [WH]ij}{\partial \theta_{p,q}})$$
$$= \sum_{ij}([WH]_{i,j}^{\beta-1}\frac{\partial [WH]_{i,j}}{\partial \theta_{p,q}} - \beta V_{i,j}[WH]_{i,j}^{\beta-2}\frac{\partial [WH]_{i,j}}{\partial \theta_{p,q}})$$

We know by definition that each data point in the estimated product matrix of index [i,j] is the sum of the term-by-term multiplication of the ith row of W and the jth column of H.

$$[WH]_{i,j} = \sum_k W_{i,k} H_{k,j}$$

The partial derivative of [WH] over H for a given W :

As W is considered constant, we have :

$$\frac{\partial [WH] i,j}{\partial Hp,q} = \sum_k W_{i,k} \frac{\partial H_{k,j}}{\partial H_{p,q}}$$

This partial derivative is only non-zero for **k = p** and **j = q**, which can be translated to **2 Dirac delta function in those points**. The sum over k can be removed since all terms will be equal to zero except when **k = p**.

$$\frac{\partial [WH] i,j}{\partial Hp,q} = \sum_k W_{i,k} \delta_{j,q} \delta_{k,p} = W_{i,p} \delta_{j,q}$$

The gradient of the cost function with respect to H for a given W :

if we substitute this result in the previous gradient equation :

$$\begin{aligned}
\frac{\partial \beta Div(V|WH)}{\partial H_{p,q}} &= \sum_{ij} ([WH]_{i,j}^{\beta-1} \frac{\partial [WH]_{i,j}}{\partial H_{p,q}}) - \sum_{ij} (V_{i,j} [WH]_{i,j}^{\beta-2} \frac{\partial [WH]_{ij}}{\partial H_{p,q}}) \\
&= \sum_{ij} ([WH]_{i,j}^{\beta-1} W_{i,p} \delta_{j,q}) - \sum_{ij} (V_{i,j} [WH]_{i,j}^{\beta-2} W_{i,p} \delta_{j,q}) \\
&= \sum_{i} ([WH]_{i,q}^{\beta-1} W_{i,p}) - \sum_{i} (V_{i,q} [WH]_{i,q}^{\beta-2} W_{i,p})
\end{aligned}$$

We can then go back to a more simplified matrix form :

$$\begin{aligned}\frac{\partial \beta Div(V|WH)}{\partial H_{p,q}} &= \sum_i (W_{p,i}^T [WH]_{i,q}^{\beta-1}) - \sum_i (W_{p,i}^T [(WH)^{\beta-2}V]_{i,q}) \\ &= W^T (WH)^{\beta-1} - W^T [(WH)^{\beta-2}V]\end{aligned}$$

finally, we get a gradient that can be written as a difference between two non-negative functions.

The gradient of the cost function with respect to W for a given H:

The same process can be used to compute the solution over W to get :

$$\frac{\beta Div(V|WH)}{\partial W_{p,q}} = (WH)^{\beta-1} H^T - [(WH)^{\beta-2}V] H^T)$$

06. Deriving Multiplicative Update Rules:

We know that :

$$H^{(i+1)} \leftarrow H^{(i)} \bigotimes \frac{\nabla_H^- \beta_{Div}}{\nabla_H^+ \beta_{Div}}$$

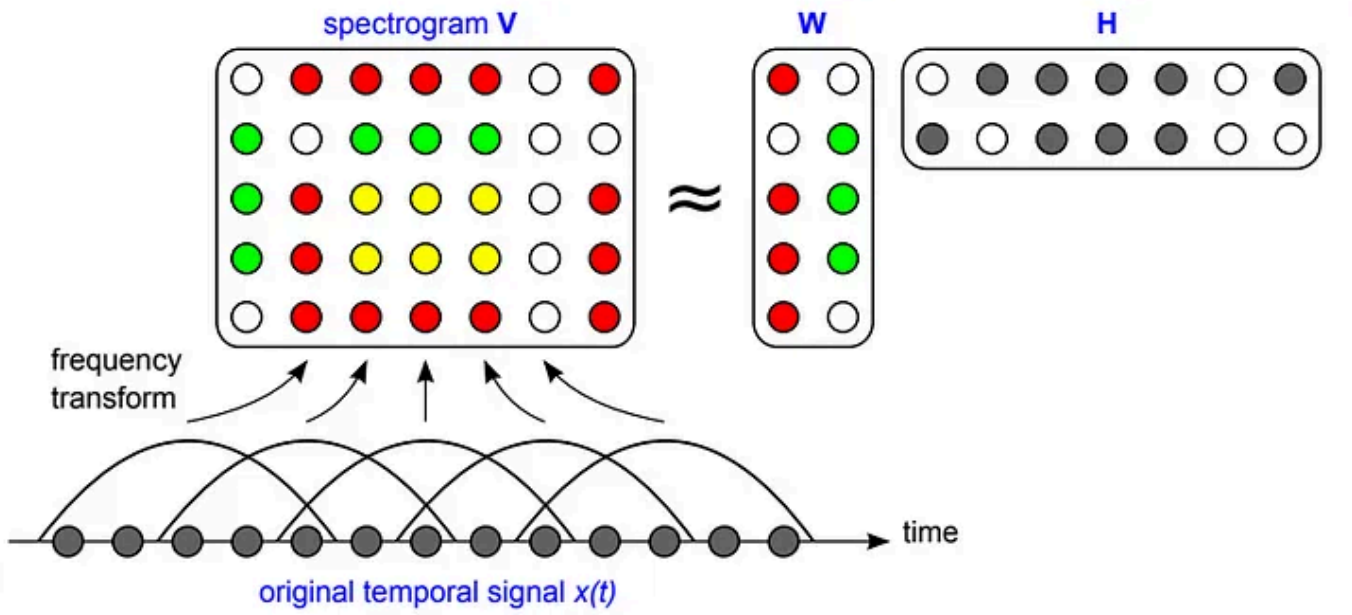
$$W^{(i+1)} \leftarrow W^{(i)} \bigotimes \frac{\nabla_W^- \beta_{Div}}{\nabla_W^+ \beta_{Div}}$$

Using the above-mentioned formulas and the derivatives produced in the last section, we finally get the update rules of both matrices W and H.

$$H^{(i+1)} \leftarrow H^{(i)} \bigotimes \frac{W^T [(WH)^{\beta-2}V]}{W^T (WH)^{\beta-1}}$$

$$W^{(i+1)} \leftarrow W^{(i)} \bigotimes \frac{[(WH)^{\beta-2}V]H^T}{(WH)^{\beta-1}H^T}$$

Hands-on example :



NMF-based audio spectral analysis (source)

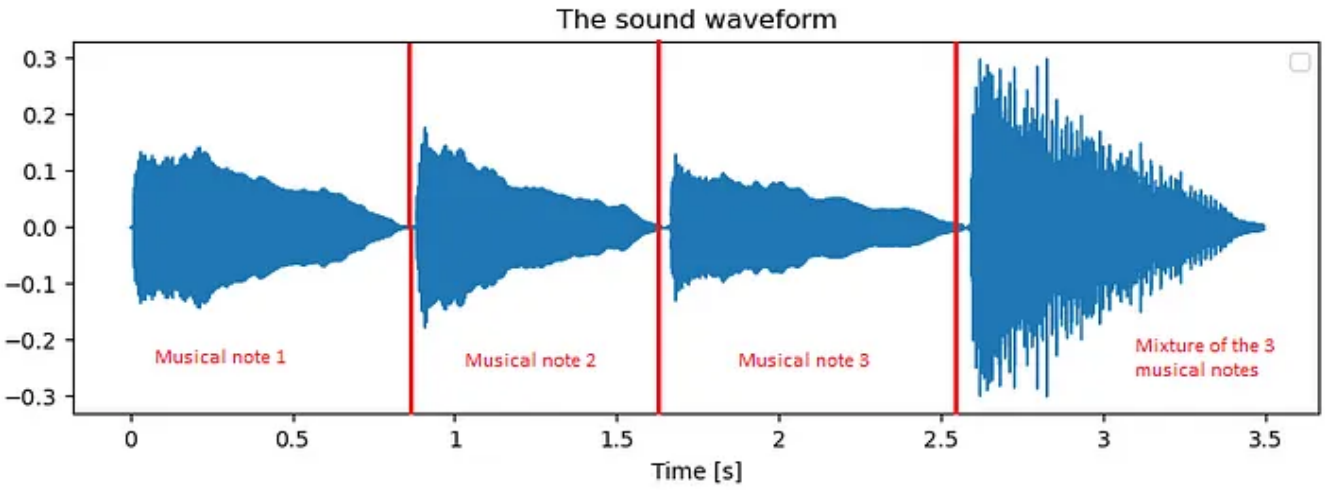
Now that we got the expression of the update rules, we can jump on the **different steps for extracting the audio sources by using NMF** applied to the spectrogram of a **short piano sequence.**

Unfortunately, I couldn't embed audio sounds into this medium article, I would be providing the GitHub that goes with this article, so you could run the complete code.

We will discuss the important steps to complete the task.

1. Load and display the audio sound signal

The audio consists of 3 consecutive different piano notes followed by the 3 notes played together.



output of the code above (Image by the author)

2. Compute Short Term Fourier Transform of the signal

An audio sound is a signal in which its frequency components vary over time. To have a good representation of such signals, we use the **Short-Term Fourier Transform (STFT)**, where its complex-valued coefficients provide the frequency and phase content of local sections of a signal as it evolves over time.

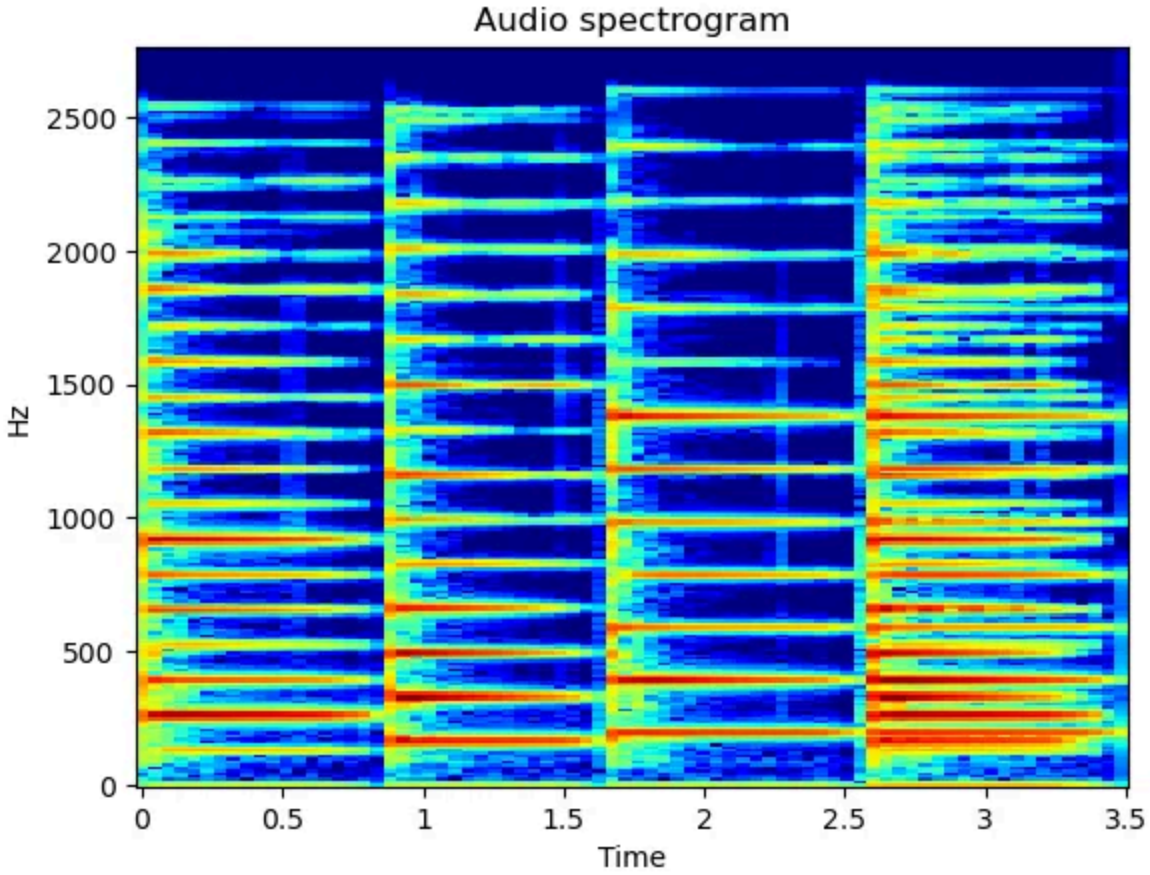
A **spectrogram** is a visual representation of the **changing spectra as a function of time**. Each column of a spectrogram represents a **time** frame of the audio, and each row represents a certain **frequency**.

The spectrogram corresponds to the squared magnitude of the STFT.

$$spectrogram(t, \omega) = |STFT(t, \omega)|^2$$

Definition of STFT (Image by the author)

We will be using this spectrogram as our data matrix V in the NMF, as it satisfies the non-negativity criterion.



Spectrogram of the audio signal (Image by the author)

3. Apply the Non-Negative Matrix Factorisation :

The NMF algorithm is quite simple to implement once we derived the update rules for β divergences type of cost functions. The algorithm consists of 3 steps :

1. Randomly initialize W and H matrices with non negative values.
2. Update W and H, by considering the other fixed.
3. Repeat 2 until the convergence threshold is hit.

Define a function that computes the β Divergence

Help function to compute the cost function for a given β

Define the main NMF function

Running the NMF algorithm on the data matrix.

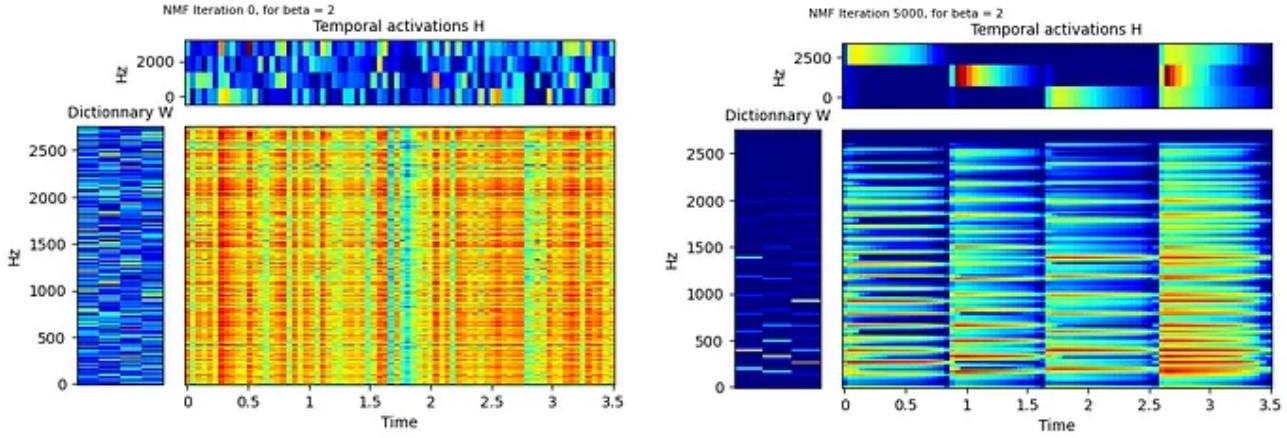
Visualizations

In the next figure, we can see a visual plot of the W, H, and their product WH which represent an approximation of the data matrix V.

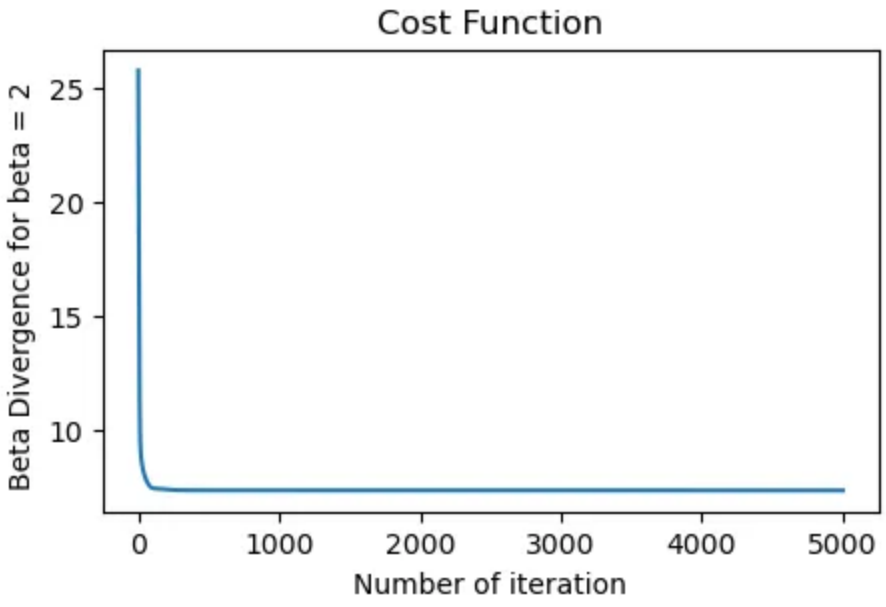
$$V \approx WH = \hat{V}$$

Left : Random initialization of the W and H matrices.

Right : The last iteration, updating stopped after 5000 iterations. The convergence threshold wasn't hit, because it was set too low, but we can see that the cost function has drastically dropped.

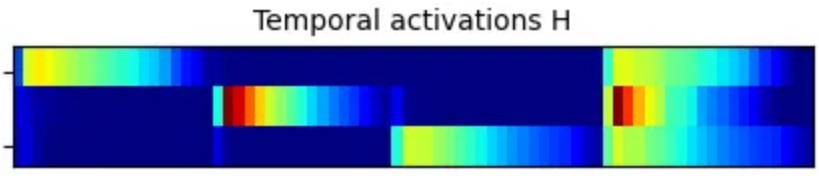


(Image by the author)



Loss function in terms of the number of iterations (Image by the author)

Now, if we look closely at the activation matrix H, we can see that the algorithm has successfully been able to distinguish the three consecutive notes as different audio sources (it has separated it into 3 sources because we have set S, the common dimension of W and H, to 3).

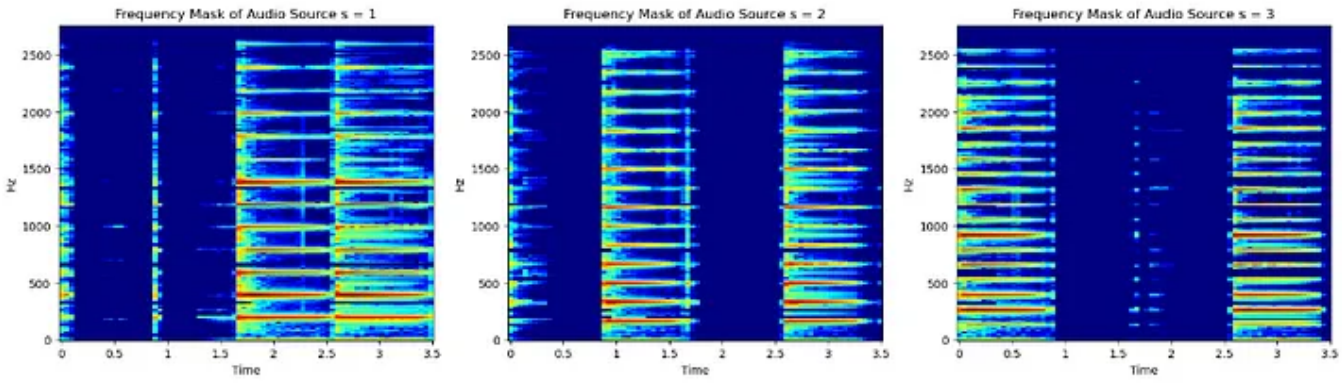


(Image by the author)

4. Filtering the different audio sources

Each extracted sound source k, can be calculated by multipling the **k-th column of W** and **k-th line of H.**

$$S_k = W_k H_k$$

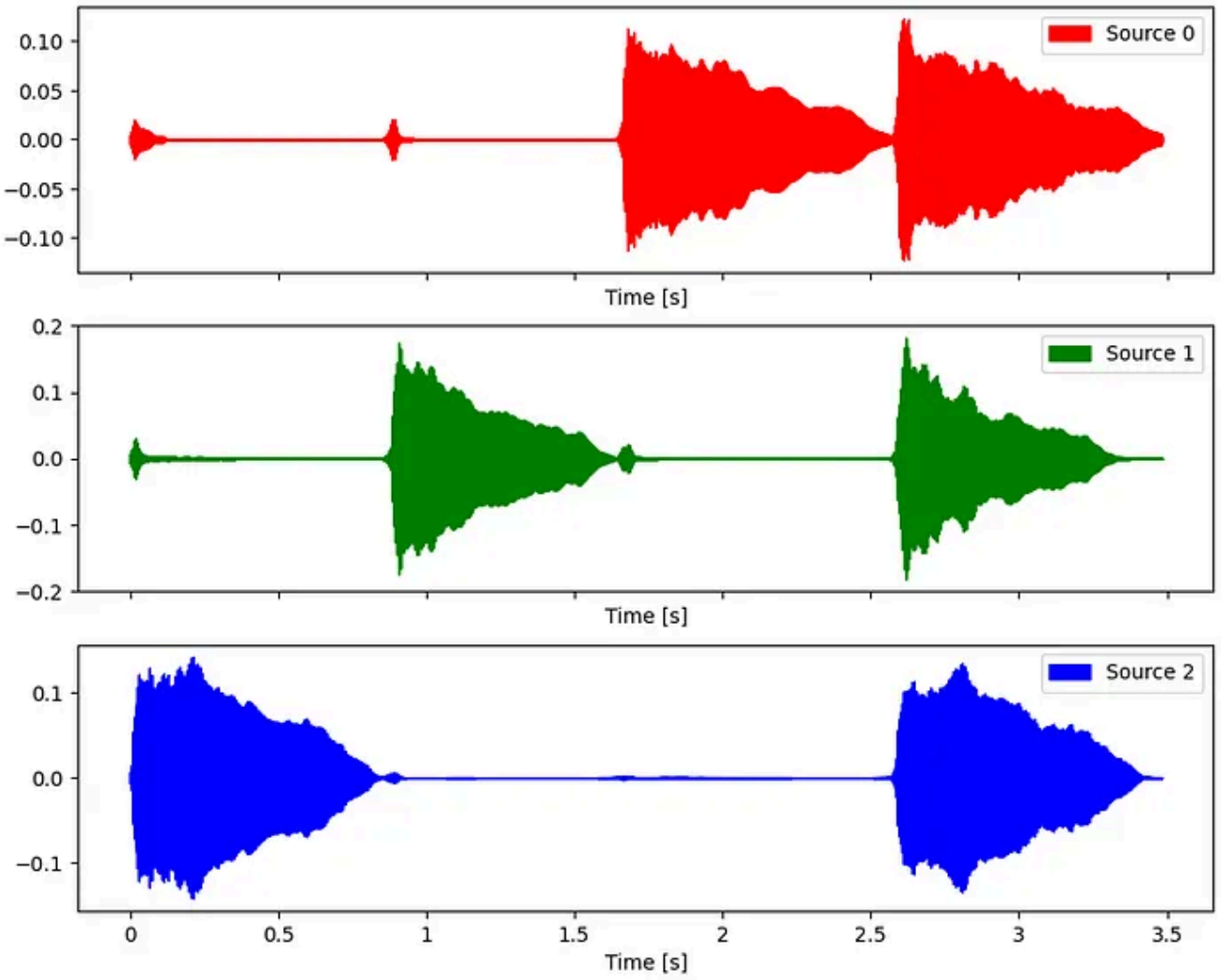


spectrograms representing the extracted sources (Image by the author)

5. Reconstructing source's audio signals

The phase of the sources is recovered by simply adding the phase of the original mixture audio, which was computed by the STFT, to the filtered sources.

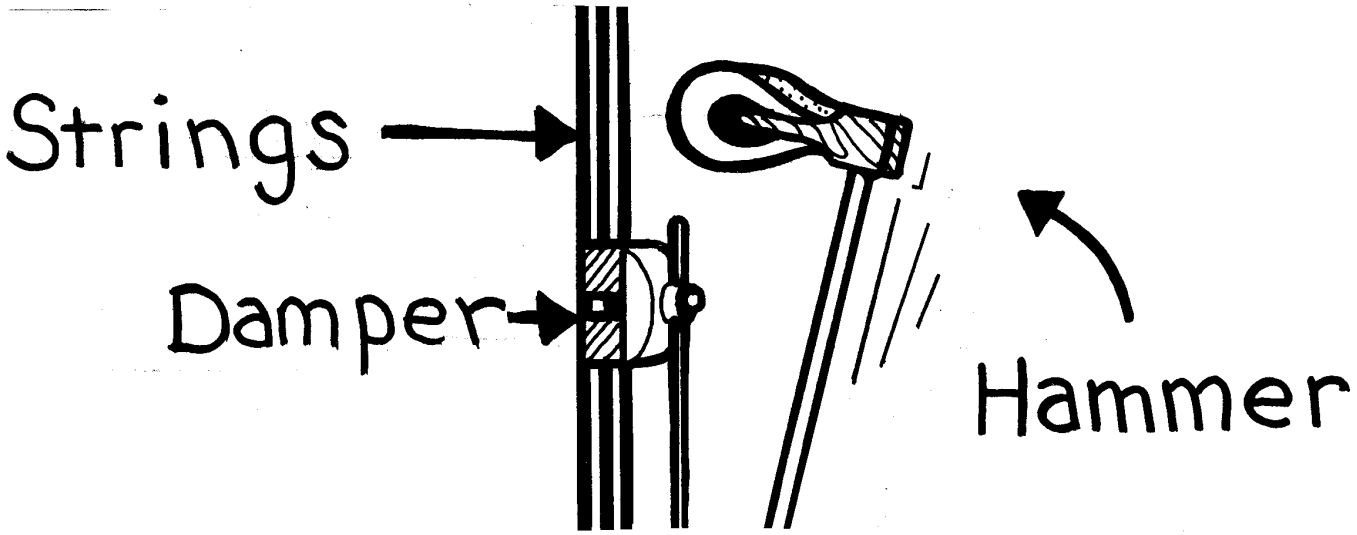
The three audio source signals are then reconstructed with the inverse STFT.



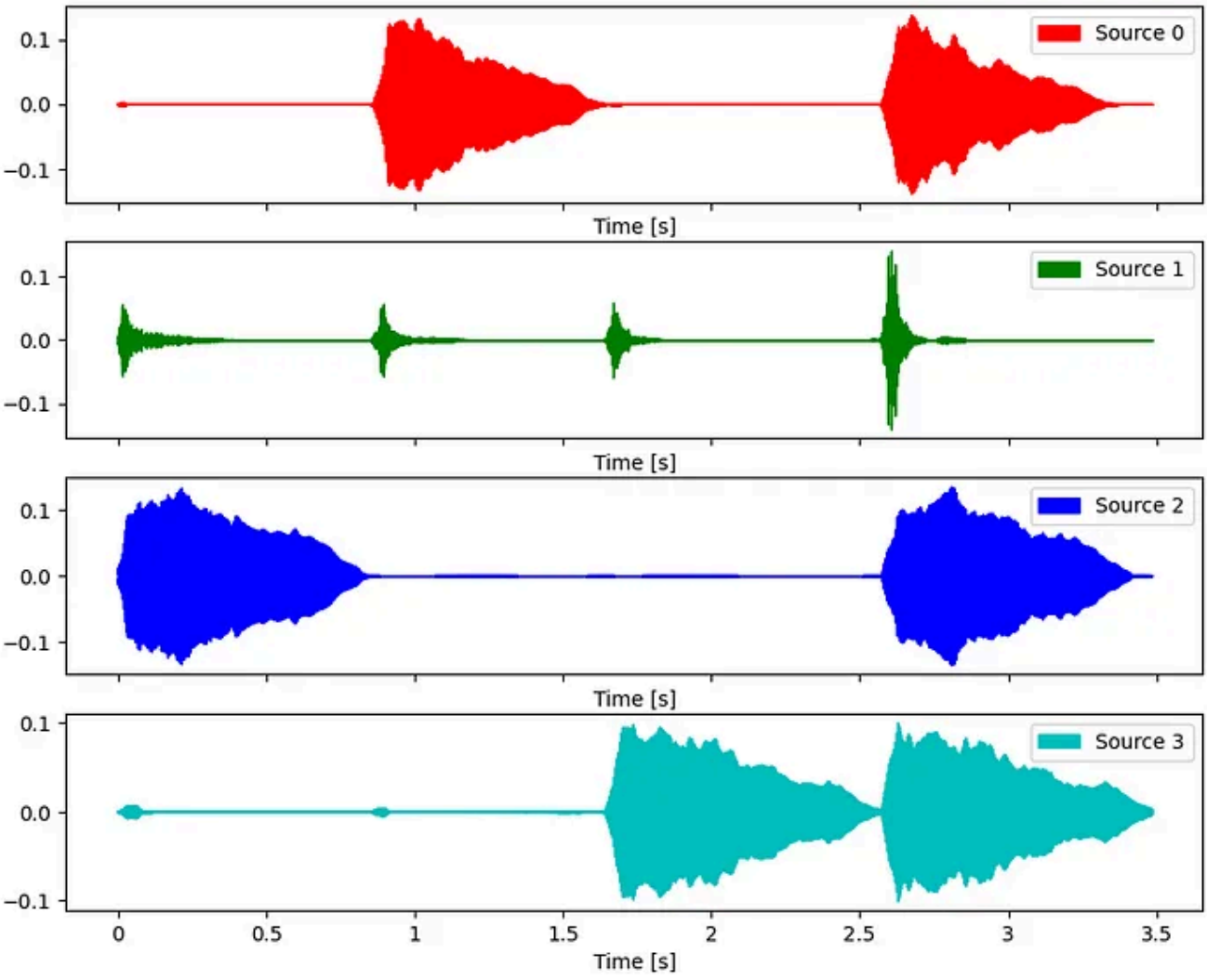
Extracted audio sources (Image by the author)

One of the challenges in applying NMF to sound source separation is determining the appropriate number of sound sources. NMF can be applied in several different ways to address this issue. One popular method is to use an iterative algorithm that starts with a small number of sound sources, and then increases the number of sound sources until the separation is satisfactory.

In our case, if S, the number of sources to extract is set to S = 4. We are able to extract the the sound of the actionning the piano's hammer. (Source 2 , in green below)



(source)



Extracted piano sources (source 0, source 2 and source 3) and the hammer's actionning sound (Image by the author)

In conclusion, Non-negative matrix factorization (NMF) is a powerful sound source separation technique that can extract individual sound sources from a mixture of sounds. **NMF is well suited for audio signals** as it enforces non-negativity, which is a desirable property for audio signals.

To be able to **listen to the reconstructed signal**, check out my **GitHub repository**, where you can find the complete **code.**

––

Thanks for reading my first post until the end.

References :

- Single-channel audio source separation with NMF: divergences, constraints and algorithms
- Algorithms for Non-negative Matrix Factorization
- Algorithms for nonnegative matrix factorization with the β-divergence
- NONNEGATIVE MATRIX FACTORIZATIONS AS PROBABILISTIC INFERENCE IN COMPOSITE MODELS

- Advanced Audio Processing Course at Sorbonne University.

Technology Data Science Artificial Intelligence

Written by Zahra Benslimane

Follow

19 Followers

PhD student, working on Efficient Neural Networks

More from Zahra Benslimane

Zahra Benslimane

Behind The Scenes : AI in Manufacturing

Introduction

6 min read · Mar 24, 2024

50

See all from Zahra Benslimane

Recommended from Medium

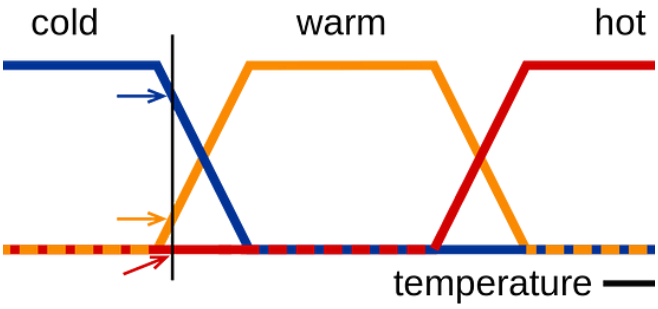
Sandaruwan He... in Data Science and Machine Le...

FP-Growth Algorithm in Data Mining

In data mining, particularly in the discovery of frequent itemsets and association rules, the...

6 min read · Jan 23, 2024

64



Oleh Dubetcky

Mastering Fuzzy Logic in Python

Mastering fuzzy logic in Python involves understanding the principles of fuzzy logic...

9 min read · Apr 3, 2024

3

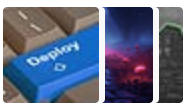
Lists

AI Regulation

6 stories · 472 saves

ChatGPT prompts

47 stories · 1632 saves



Predictive Modeling w/ Python

ChatGPT

21 stories · 663 saves

20 stories · 1251 saves

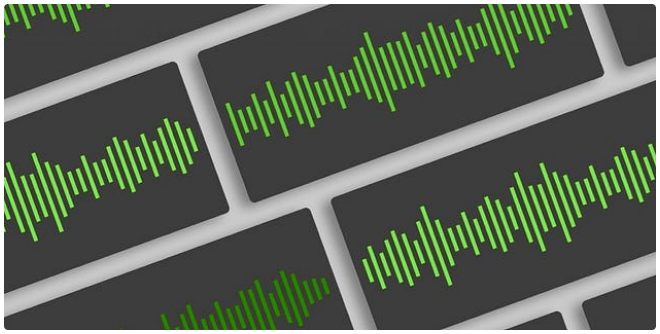
Mark Stent

Dynamic Time Warping

An introduction

10 min read · Apr 9, 2024

8



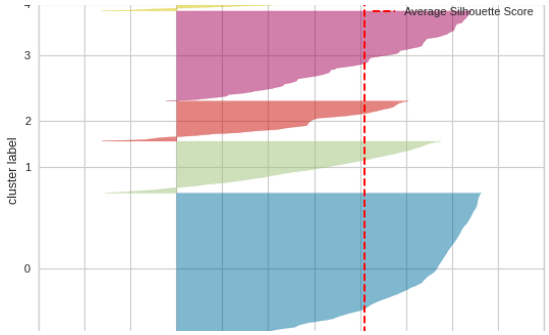
Serkan Celik in Huawei Developers

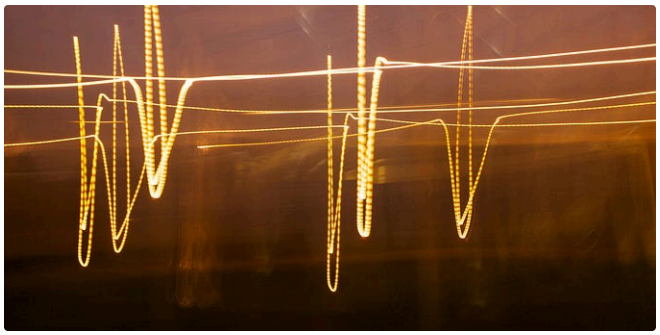
Basics of Audio Processing

Hi! In this article, we will talk about the basic components of MindAudio and the audio pre...

6 min read · Dec 8, 2023

68





Prasan N H

Exploring the World of Clustering: K-Means vs. K-Medoids

Clustering is a powerful technique in machine learning and data analysis, used to group...

4 min read · Jan 10, 2024

43 1

Harshivs

Wavelet Spectrogram: Leveraging Wavelet Transform for...

Unlock hidden details in sound! Wavelet spectrograms, traditional for fast frequency...

4 min read · Mar 19, 2024

151

See more recommendations