Blind Source Separation with Negentropy-based Independent Component Analysis

Gabriele Carrino[a], Nicolò Pisanu[b]

[a]Polytechnic University of Milan, gabriele.carrino@mail.polimi.it
[b]Polytechnic University of Milan, M. Sc. Music and Acoustic Engineering, nicolo.pisanu@mail.polimi.it

Abstract

The aim of this project is to implement and validate the Fast ICA algorithm based on Negentropy, as described by Alaa Tharwat [1], Tu Shijie, and Chen Hang [2].

We reproduced the core functionalities of the algorithm and evaluated its performance using both synthetic data and simulated real-world scenarios. These scenarios utilized pre-recorded sound sources mixed in virtual environments created with the Pyroomacoustics library [3].

This experimental setup allowed us to separate and reconstruct the original sound sources using the FastICA algorithm and assess its effectiveness under various mixing conditions. We conducted the evaluation through direct comparisons and the application of Blind Source Separation (BSS) metrics [4].

1. Introduction

Independent Component Analysis (ICA) is a signal processing technique used to separate hidden source signals from their mixed observations, which are often linear combinations of the original, statistically independent source signals. ICA aims to recover these source signals by identifying a linear transformation that maximizes the statistical independence of the separated components. This paper explores the implementation and validation of such an ICA algorithm.

The first section provides a theoretical background on ICA, detailing the steps required to perform the algorithm. Core concepts, including blind source separation and the underlying assumptions of the ICA algorithm, will also be discussed.

Following the theoretical overview, the second section focuses on the development of a validation framework for the implemented ICA algorithm. This section describes the creation of a simulated dataset using the Pyroomacoustics library, which enables the generation of controlled environments to evaluate the algorithm's performance under various real-world mixing conditions. Then the validation process will be outlined, assessing the algorithm's ability to recover the original source signals from the mixed data.

Finally, we will compare results from the real-world simulated scenarios with those from an ideal synthetic one.

2. Theoretical background

2.1. Blind Source Separation

Blind source separation (BSS) refers to the concept of recovering hidden source signals from mixed observations, without any prior knowledge of the mixing process or the original source signals themselves.

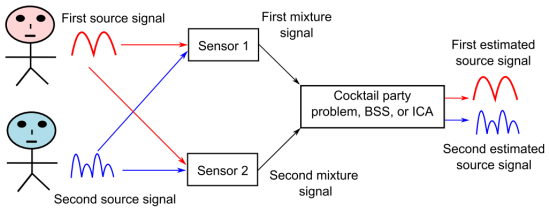

Figure 1: the cocktail party problem

One of the most widely used examples of Blind Source Separation is the cocktail party problem, where the goal is to extract the voices of individual party members even though people are speaking at the same time.

2.2. Mathematical formalism

The notation used in this paper follows the one in the work by Alaa Tharwat [1].

We define a *mixture of signals* as:

$$X = AS$$

where: A is the mixing matrix, S are the original signals and X are the mixtures.

Our aim, by applying ICA, it to find the *unmixing matrix*:

$$Y = W^T X$$

where: W is the unmixing matrix, X are the mixtures and Y are the recovered signals.

Finding the unmixing matrix lets us perform *Blind Source Separation* i.e., by projecting the mixed sources X onto W we will be able to reconstruct the original signals.

2.3. Assumptions behind the algorithm

The main idea behind ICA is the inversion of the *central limit theorem* (CLT), i.e., we are trying to find the most *non-Gaussian* and *independent* signals that give rise to the distribution we see in the dataset.

For this reason, we need to assume that the source signals are *independent* and *non-Gaussian* (or at most one Gaussian component). If this were not the case, the CLT would not hold, and we would not be able to reverse it.

2.4. Neg-entropy maximization

To be able to do what we described above we need to find a suitable metric for *independence* and *non-gaussianity* to maximize, a possible one for the non-gaussianity is *neg-entropy*.

We define *neg-entropy* as the kullback-leibler divergence of a random variable Y from a gaussian distribution with the same mean and variance:

$$J(Y) = D(Y||Y_{gauss}) = H(Y_{gauss}) - H(Y)$$

2.5. Mutual information

In information theory, mutual information quantifies the shared information between two random variables, X and Y. It measures the *reduction in uncertainty* about X after observing Y (and vice versa). Higher mutual information indicates a stronger dependence between the variable, while lower / zero values indicate independence.

Formally we define mutual information as:

$$I(X, Y) = H(X) - H(X|Y)$$

where $H(X|Y)$ is the conditional entropy of X given Y.

2.6. Mutual information minimization

An intersting fact is that by *maximizing neg-entropy* we aren't only searching for the *most non-gaussian* signals, but also for the *most independent* among them as we are effectivly *minimizing mutual information*.

This can be proven by the fact that *mutual information* can be rewritten by neg-entropy in this way:

$$I(y_1, ..., y_n) = C - \sum_{i=1} J(y_i)$$

where C is a constant term.

From the equation above it is clear that maximizing neg-entropy is related to minimizing mutual information as they differ only by a sign and a constant C.

2.7. The need for approximations

Calculating the entropy for finite data can be problematic, for this reason the algorithm illustrated below maximizes an approximation of neg-entropy given by:

$$J(y) = k_i(E[G_i(y)] - E[G_i(v_{gauss})])^2$$

Where k_i is a constant, v_{gauss} is a gaussian variable with zero mean and unit variance and G_i can be a quadratic function such as:

$$G_1(y) = log(cosh(y))$$

$$G_2(y) = e^{-y^2/2}$$

We denote the first and second derivatives of G by g and g' respectively.

3. FastICA algorithm

FastICA is a specific implementation of the ICA algorithm that extracts independent components by maximizing the non-Gaussianity by maximizing the negentropy for the extracted signals using a fixed-point iteration scheme.

3.1. Preprocessing

Before being able to feed our data to the ICA algorithm we need to apply a *centering* and a *whitening* step.

Those steps are really important as they help in decorrelating the data and making the independent components easier to detect.

3.1.1. Centering

Centering refers to the process of subtacting the mean from our data:

$$D = X - \mu$$

3.1.2. Whitening

Whitening instead refers to two procedures:

1. *Decorrelation*: We perform decorrelation by applying *principal components analysis*, a technique to perform dimensionality reduction by projecting the centered data in the space of maximum variance, i.e the eigenvectors of the covariance matrix:
$$T = U^T D$$
2. *Normalization*: After performing decorrelation we need to scale the data to be of unit variance (not having unit variance injects a bias toward the components with higher variance):
$$T_{norm} = \frac{T}{T_{std}}$$
Oss: the notation above refers to the pointwise division between the matrix T and the vector of the standard deviations T_{std}.

3.2. Undercomplete ICA

Undercomplete ICA refers to the case when we have more mixture than sources. To tackle this case, when we perform *decorrelation*, we keep only the first components up to the *number of sources*. This helps in *lowering the dimensionality* of the problem and makes the algorithm work more efficiently.

3.3. Multiple component extraction

The classical ICA algorithm estimates only one component. To estimate more then one we need run the algorithm multiple times and add a step in the middle to be sure to extract components that we have not already extracted.

We can do this by performing the *Gram-Schmidt orthogonalization* for every previously obtained component:

$$w_p = w_p - \sum_{j=1}^{p-1} (w_p^T w_j) w_j$$

3.4. Convergence and hyperparameters

FastICA has a cubic or at least quadratic convergence speed and hence it is much faster than Gradient-based algorithms that have linear convergence. Additionally, FastICA has no learning rate or other adjustable parameters which makes it easy to use.

3.5. Update rule

The update rule for the FastICA algorithm can be derived by maximizing the neg-entropy $J(Y)$ for the extracted signals using a fixed-point iteration scheme as seen in [1].

The final formula is composed of two steps:

$$w_p = \frac{1}{M} X g(w_p^T X)^T - E[g'(w_p^T X)] w_p$$

$$w_p = \frac{w_p}{\|w_p\|}$$

where the second is needed to scale back w_p to unit norm.

3.6. Complete algorithm

- Input: C number of desired components, X preprocessed data.
- Output: S extracted independent components, W unmixing matrix.

```
for p = 1 to C:
  w_p = random init

  while w_p changes:
    w_p = (1/M) X g(w_p^T X)^T - E[g'(w_p^T X)] w_p
    w_p = w_p - sum_{j=1}^{p-1} (w_p^T w_j) w_j
    w_p = w_p / ||w_p||

  W_:p = w_p

return:
  S = W^T X,   W
```

3.7. Ambiguities of ICA

ICA has some inherent ambiguities:

- **Order of Independent Components:** The order in which the components are extracted is arbitrary. This is because the weight vector w_p is initialized randomly and iteratively updated to find one independent component. Thus, ICA extracts source signals without a specific order.
- **Sign of Independent Components:** The sign of the extracted components does not affect their independence, as such the matrix W in n-dimensional space has 2-n local minima , i.e., two local maxima for each independent component, corresponding to s_i and -s_i.

4. Dataset generation

The dataset was generated by extracting sound clips from Freesound [5] and refining them using Audacity [6] to adjust amplitude levels, fading and durations as needed. Additionally, to replicate realistic acoustic environments, virtual rooms were constructed using Pyroomacoustics [3]. Within these simulated rooms, the sound clips were assigned to virtual sources for playback and diffusion.

Microphones placed within the environments captured the diffused sound. Various factors such as the positioning of the sources, the layout and directional sensitivity of the microphones, and the unique characteristics of the rooms contributed to the signals recorded by each microphone.

During the room simulation process, the dataset obtained yielded a multi-channel signal, where each channel represents a distinct audio signal captured by a microphone. This results in a matrix of M signals (mixtures), where each row corresponds to the signal captured by a specific microphone. The aim of the subsequent processing with the fastICA algorithm is to extract the original N source signals from these matrices.

4.1. Real-world acoustic

Before presenting the experimental validation of the algorithm, it is useful to review some key concepts in acoustic physics to understand the subsequent discussions.

When recording sound with microphones at varying distances from a source, distinct differences arise in the recorded signals. Suppose we have a sound source and two microphones: microphone M_1 placed at a distance of d_1 from the source, and microphone M_2 placed at a distance of d_2 from the source, with $d_1 < d_2$. Below are the key differences in the signals recorded by the two microphones:

- **Intensity:** The signal recorded by M_1 will be stronger in intensity compared to M_2 due to the inverse square law, which states that the intensity of sound diminishes with the square of the distance from the source. Mathematically,

the intensity I at a distance d from a sound source is given by:

$$I = \frac{P}{4\pi d^2}$$

where P is the acoustic power of the sound source. If I_1 is the intensity at distance d_1 and I_2 is the intensity at distance d_2, then:

$$I_2 = I_1 \left(\frac{d_1}{d_2}\right)^2$$

[1]

- **Amplitude:** Since intensity is proportional to the square of the amplitude, the amplitude of the signal recorded by M_2, denoted as A_2, will be scaled by the ratio of the distances compared to that recorded by M_1, denoted as A_1:

$$A_2 = A_1 \left(\frac{d_1}{d_2}\right)$$

[2]

- **Time Delay:** Given the speed of sound in air is approximately $v \approx 343$ meters/second, the sound takes time $t = \frac{d}{v}$ to travel a distance d. Thus, the difference in arrival time Δt between the signals recorded by M_1 and M_2 is:

$$\Delta t = \frac{d_2 - d_1}{v}$$

[3] At a sampling rate of f_s samples per second, this corresponds to:

$$\Delta n = \Delta t \cdot f_s = \frac{(d_2 - d_1) \cdot f_s}{v}$$

[4]

- **Phase Difference:** This time delay introduces a phase difference $\Delta\phi$ between the two signals at a frequency f, given by:

$$\Delta\phi = 2\pi f \Delta t = 2\pi f \frac{d_2 - d_1}{v}$$

[5]

- **Spectral Differences:** The higher frequencies may be more attenuated in the signal recorded by M_2 compared to M_1 due to the increased distance, as higher frequencies tend to be absorbed more by the air. This can be modeled by a frequency-dependent attenuation factor $H(f, d)$:

$$H(f, d) \approx e^{-\alpha(f)d}$$

where $\alpha(f)$ is the absorption coefficient at frequency f. [6]

- **Environmental Differences:** The microphone M_2 will capture more reflected and reverberant sound compared to M_1. This is because the sound travels further and interacts more with the environment (walls, objects, etc.) before reaching M_2, resulting in a more complex and diffuse signal. The recorded signal can be modeled as a combination of the direct sound and multiple reflections:

$$x(t) = x_{\text{direct}}(t) + \sum_i x_{\text{reflected},i}(t - \tau_i)$$

where τ_i represents the delay of each reflected path.

Thus, the signal $x_1(t)$ recorded by M_1 can be expressed as a combination of the source signals $s_1(t)$ and $s_2(t)$ with certain coefficients, and similarly for $x_2(t)$ recorded by M_2:

$$x_1(t) = a_{11} \cdot s_1(t) + a_{12} \cdot s_2(t)$$

$$x_2(t) = a_{21} \cdot s_1(t) + a_{22} \cdot s_2(t)$$

These differences in amplitude, intensity, time delay, phase, spectral content, and environmental effects are crucial for techniques such as ICA to effectively separate the individual source signals from the recorded mixtures.

4.2. Experimental scenarios

The experiment unfolds in three distinct scenarios. The first scenario is entirely synthetic, featuring a mixing matrix crafted by us, while the other two are constructed using Pyroomacoustics. A specific number of sources N and microphones M, are strategically positioned within the space with unique acoustic characteristics to create realistic and plausible scenes and real-world situations. Additionally, a variety of sounds are employed to ensure a diverse auditory landscape.

The experiment incorporates different types of microphones and rooms specifically created to have distinct impulse responses. Moreover, the ratio of microphones to sources in each scenario is carefully considered to reflect different practical setups. This comprehensive approach aims to provide a thorough evaluation and to understand the algorithm's performance across heterogeneous cases and dataset.

- **Scenario 1:** A synthetic cocktail party scenario, where $N = 3$ voice signals are combined using a crafted mixing matrix to produce $M = 3$ mixtures. In this ideal case, only the amplitudes of the sources are modified, as the signals do not propagate in the environment, and other characteristics of the signals remain unchanged.

- **Scenario 2:** A classic cocktail party scenario with $N = 3$ individuals talking in a room equipped with $M = 3$ hypercardioid microphones (represented in Figure 2). The microphone array was centered in the room to record the conversations, capturing voices in different proportions and angles. The small room had highly absorptive walls, with a small amount of reflection considered in the simulation.

[1] For example, if $d_1 = 1$ meter and $d_2 = 2$ meters, then $I_2 = I_1 \left(\frac{1}{2}\right)^2 = \frac{I_1}{4}$. All examples below will use these distances.

[2] $A_2 = A_1 \left(\frac{1}{2}\right) = \frac{A_1}{2}$.

[3] $\Delta t = \frac{2-1}{343} \approx 0.00291$ seconds.

[4] If $f_s = 44100$ Hz, then $\Delta n = \frac{(2-1)\cdot 44100}{343} \approx 128$ samples.

[5] If $f = 1000$ Hz, then $\Delta\phi = 2\pi \cdot 1000 \cdot \frac{2-1}{343} \approx 18.3$ radians.

[6] For a given absorption coefficient $\alpha(f)$ at $f = 1000$ Hz, the attenuation factor will be different for each distance, with $H(f, 2)$ being less than $H(f, 1)$.

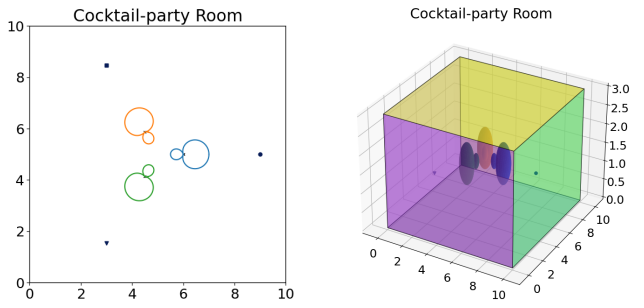

Figure 2: Room with $N = 3$ sources and $M = 3$ microphones.

The arrangement and directional sensitivity of the microphones provided a detailed and varied audio mixture of the individual conversations.

- **Scenario 3:** A jazz-jam concert scenario, where $N = 5$ instruments were positioned on a stage at the bottom of the room, and $M = 10$ microphones were used. Each instrument had its own dedicated cardioid microphone, specifically chosen based on the characteristics of the instrument. Additionally, 5 other omni-directional microphones were strategically placed on the public side of the room. The trapezoidal shape of the room created very complex reflections. All these microphones captured the sounds of the instruments with several time-delay and reflection of the ambient environment, resulting in very diverse audio mixtures.

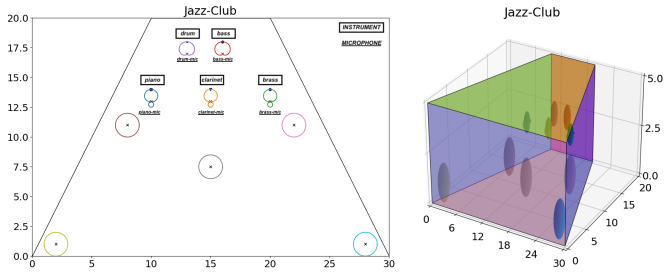

Figure 3: Room with $N = 5$ sources and $M = 6$ microphones.

In all of these cases, the number of mixtures resembles the number of microphones and the number or requested sources to fastICA corresponds to the actual number of original sources.

5. Validation

After the application of fastICA in all the scenarios, the validation of the algorithm focuses on comparing the original source signals with the reconstructed ones. One main challenge we encountered in this part is that the fastICA algorithm provides the independent signals as requested, but, due to inherent ambiguities, assigns them serial numbers in a random order and with arbitrary signs. This complicates the comparison between the original and reconstructed signals.

5.1. Matching and flipping

In order to ensure that each extracted source was associated with its corresponding original source, the Source-to-Interference Ratio (SIR) metric was utilized. The SIR will be explained in the following paragraph, as it is also used in the validation of the results.

After performing a normalization to ensure all signal amplitudes were within the range $[-1, 1]$, we computed the SIR for each possible pair of original and reconstructed sources. This resulted in N possible combinations for each original source. The pairs with the highest respective SIR were then associated. In this way, the problem of assignment of the serial number was largely resolved.

The second issue to address is the phase inversion of the reconstructed signal. This issue has numerical and analytical implications rather than perceptual ones, as the human ear cannot detect phase inversion. Addressing this issue is crucial for achieving accurate numerical results and better visual representations in plots.

To find an automated solution, we examined the positions of the extreme values (maxima and minima) of the signals after permutation matching. If the extreme value of the original source changed its sign in the reconstruction, this indicated a phase inversion, necessitating a reversal of the reconstructed signal. A simpler method to compute this numerically is to use the dot product, where a positive result indicates no inversion. This problem becomes challenging when the reconstruction quality is poor, making it ambiguous to determine the correct phase.

Once we resolved the issues related to the ICA outputs, we proceeded to compute the metrics for the various proposed scenarios.

5.2. BSS Metrics

The measure of performance is based on the principle that a given estimated source can be decomposed as:

$$\hat{s}(t) = s(t) + e_{\text{interf}}(t) + e_{\text{noise}}(t) + e_{\text{artif}}(t)$$

where:

- $s(t)$ is the original source.
- $e_{\text{interf}}(t)$ is the noise due to interference with other sources.
- $e_{\text{noise}}(t)$ is a perturbation term due to noise (typically Gaussian).
- $e_{\text{artif}}(t)$ is the noise due to artifacts created by the separation process.

The following metrics are used to quantify the performance of the separation process:

- **SDR (Source-to-Distortion Ratio)**: measures the overall quality of the separated source by comparing the original source signal to the total distortion present in the estimated source.

$$\text{SDR} = 10 \log_{10} \frac{\sum s(t)^2}{\sum (e_{\text{interf}}(t) + e_{\text{artif}}(t))^2}$$

  A higher SDR indicates better overall separation quality, with less distortion in the estimated source.

- **SIR (Source-to-Interference Ratio)**: assesses how well the separation process has isolated the target source from interfering sources. Measures the ratio of the energy of the true sources to the energy of the interference caused by other sources.

$$\text{SIR} = 10 \log_{10} \frac{\sum s(t)^2}{\sum e_{\text{interf}}(t)^2}$$

  A higher SIR means that the interference from other sources is minimized, indicating more effective separation of the target source.

- **SAR (Source-to-Artifact Ratio)**: evaluates the extent to which artifacts introduced by the separation algorithm affect the estimated source measuring the ratio of the energy of the true sources to the energy of any artifacts introduced by the separation process.

$$\text{SAR} = 10 \log_{10} \frac{\sum (s(t) + e_{\text{interf}}(t))^2}{\sum e_{\text{artif}}(t)^2}$$

  A higher SAR indicates fewer artifacts, implying a cleaner separation process.

- **SNR (Source-to-Noise Ratio)**: measures the amount of noise relative to the original source signal defined by the difference between the actual and the reconstructed source.

$$\text{SNR} = 10 \log_{10} \frac{\sum s(t)^2}{\sum (s(t) - \hat{s}(t))^2}$$

  A higher SNR indicates that the separated source has less noise, thus being closer to the true original source signal.

All the indices mentioned above are expressed in dB.

These metrics are commonly used to evaluate the performance of Blind Source Separation (BSS) algorithms. These metrics provide insights into how well the source separation algorithm is performing in terms of preserving the original sources while minimizing interference and artifacts.

All the metrics, except SNR, are computed using the `bss_eval` toolkit [4], which is widely used for evaluating the performance of BSS algorithms.

6. Results

Our study investigated three distinct scenarios: an ideal case and two real-world situations with simulated acoustic environments. While the third scenario presents a different narrative, the first two underline the same situation - the first being ideal and the second being real.

In the ideal cocktail-party scenario, where sources were mixed solely based on amplitude differences, the FastICA algorithm demonstrated remarkable performance, achieving high values across all BSS metrics. As depicted in Figure 4, all three sources achieved very high results and the waveforms of the original sources are perfectly overlapped with those of the reconstructed sources.

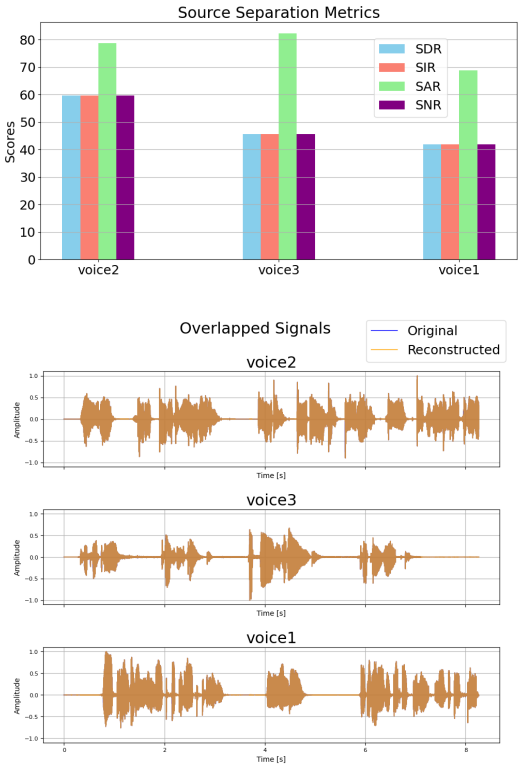

Figure 4: Results and plots of the ideal cocktail-party scenario.

The cocktail-party situation was then replicated using the same sound-clip voices in a simulated room with acoustic factors. In this environment, the sources reached the microphones with different time delays, resulting in mixtures where the signals have different onsets. This added complexity challenges FastICA's ability to separate the sources, leading to less than perfect results compared to the ideal scenario. However, all sources still achieved decent scores in Signal-to-Interference Ratio, indicating minimal interference between them (Figure 5).

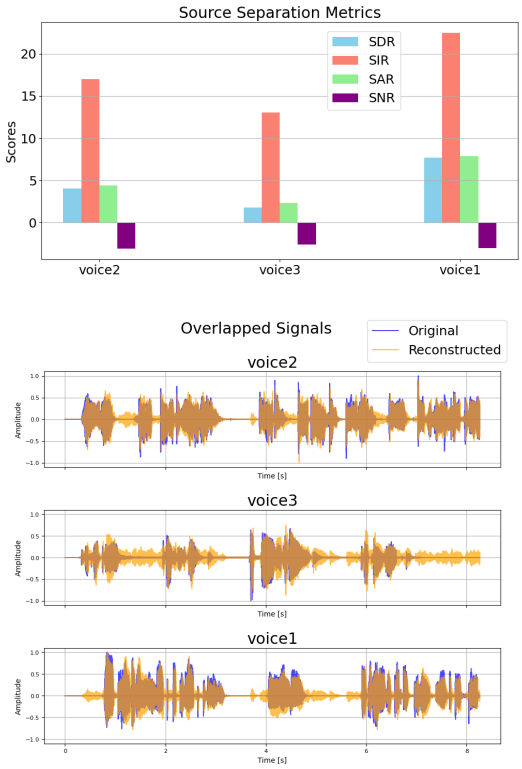

Figure 5: Results and plots of the real cocktail-party scenario.

The sound clips used were of concrete origin, non-stationary, and exhibited their own temporal evolution. Some sources proved to be more prominent, especially when other sound clips had moments of silence. This is evident in voice and musical signals where phrases, sentences, or melodies are articulated over time, with pauses and silences in between. During these silent periods, other sources can create interference.

Spectral content plays a pivotal role in source separation alongside amplitude differences. The FastICA algorithm leverages spectral information to exploit statistical independence among sources. Signals with diverse spectral characteristics exhibit distinct statistical properties, making them more amenable to separation. By considering spectral content in addition to amplitude, FastICA can effectively disentangle mixed signals and extract meaningful sources.

In the jazz-club jam scenario, each instrument occupies distinct spectral sections. Some musical phrases flow continuously, while others are punctuated and percussive. These diverse characteristics pose challenges for source separation algorithms. Figure 6 displays the results of this scenario. While most sources are separated accurately, the ”drum” source presents difficulties. Drum signals, marked by punctuality and impulsiveness, are prone to interference from other instruments. This interference alters the reconstructed drum signal, leading to discrepancies from the original.

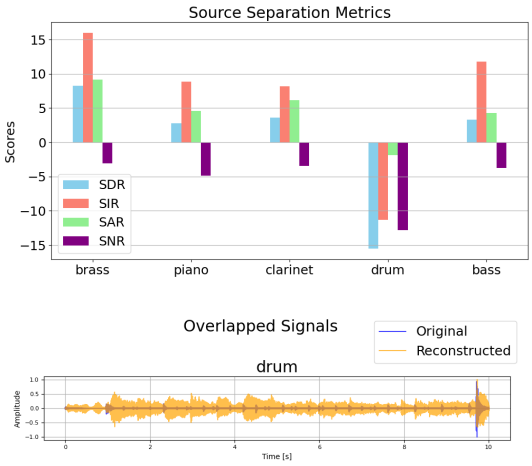

Figure 6: Results of the third scenario. Focus on ’drum’ source.

7. Conclusions

In conclusion, our study highlights the effectiveness of FastICA based on negentropy in blind source separation, particularly when sources exhibit distinctive statistical properties. The consistency of results across all scenarios underscores the robustness and coherence of our approach. However, our comparison between real-world simulations and the ideal scenario reveals crucial insights into the algorithm’s limitations when applied to complex acoustic environments. In real-world acoustic environments, sound propagation introduces alterations not only in amplitude but also in other signal properties, including phase and spectral content. Additionally, the placement of microphones at varying distances can introduce time delays in recordings. While FastICA operates on mixtures containing convolution artifacts from the ambient’s impulse response, it is unable to precisely recover the original sources in their ’unprocessed’ state. This discrepancy between the algorithm’s assumptions and real-world conditions, where sources may have also different onsets across various mixtures, poses significant challenges in direct comparison between the extracted sources and the originals.

These limitations underscore the significance of continued research and the advancement of blind source separation techniques capable of effectively addressing the complexities of real-world acoustic environments. The most significant issue we encountered with FastICA is the delays of the various signals in the microphones, which greatly affect its effectiveness in real-world scenarios. Perhaps by enhancing FastICA itself, by incorporating information about the distances between the sources to be extracted and the microphones into the algorithm, we could improve its performance.

SIR emerged as the primary indicator of separation quality, aligning closely with the subjective interpretation of human listening. Conversely, SNR, relying solely on amplitude differences, proved inadequate for assessing separation accuracy. Notably, our evaluation employed metrics considering the

entire signal, without temporal evolution tracking.

In addition, in the cases we encountered, we resolved the challenge of correctly assigning serial numbers to sources post-separation through a matching approach utilizing the highest Signal-to-Interference Ratio (SIR). Furthermore, we addressed the issue of phase inversion in the signals reconstructed by ICA by observing the positions of the extreme values of the reconstructed sources compared to the original ones.

References

[1] Tharwat, A. (2022). Independent Component Analysis: An Introduction. Springer.

[2] Shijie, T., & Hang, C. (2021). Blind Source Separation of Underwater Acoustic Signal by Use of Negentropy-based Fast ICA Algorithm. Journal of Marine Science and Engineering, 9(3), 354-366.

[3] Scheibler, R., Bezzam, E., & Dokmanić, I. (2018). Pyroomacoustics: A Python package for room acoustics and audio signal processing. `https://github.com/LCAV/pyroomacoustics`.

[4] Vincent, E., Araki, S., Theis, F., & Yeredor, A. (2006). Performance measurement in blind audio source separation. IEEE Transactions on Audio, Speech, and Language Processing, 14(4), 1462-1469.

[5] Freesound. `https://freesound.org`.

[6] Audacity. `https://www.audacityteam.org`.

[7] The code used in this paper can be found at: `https://github.com/gabricarr/FastICA-Information-Theory`.