BSS using FastICA in C

Rohan Gunjal[1.]
1. Department of Electronics and Communication Engineering
National Institute of Technology, Warangal - 506004
Warangal, India
Email: rg21ecb0a49@student.nitw.ac.in
GitHub: /theZeenX1

***Abstract*—This project investigates signal separation in Digital Signal Processing (DSP) using Fast Independent Component Analysis (FastICA). FastICA, emphasizing non-gaussianity maximization, proves robust for batch processing with large datasets. Implemented in C, the algorithm focuses on maximizing negentropy to enhance source signal separation, addressing limitations of statistical methods and neural networks. Two tests evaluate its efficacy: the first with digitally generated waveforms for controlled assessment, and the second with digitally mixed audio files to assess real-world adaptability. This exploration highlights FastICA's advantages and practical implementation in DSP, offering insights into its effectiveness and potential applications in diverse signal processing scenarios.**

***Index Terms*—Keywords: FastICA, BSS, Uncorrelated, Independence, Whitening Transform, Pre-processing, C**

I. Introduction

The goal of independent component analysis is to transform a multivariate dataset so that the resulting components are as independent as possible. The idea is to separate latent components from the dataset which is assumed to consist of linear mixtures of them.

A basic interpretation for modeling received signals is to model them as a simple linear equation with $\mathbf{s}$ as a matrix of size M, with each $\mathbf{s_i}$ of length N, $\mathbf{A}$ as the mixing matrix, and $\mathbf{x}$ as the actual M source signals of length N.

$$\mathbf{s} = \mathbf{A}\mathbf{x} \tag{1a}$$

$$\begin{bmatrix} \mathbf{s_1} \\ \mathbf{s_2} \\ \vdots \\ \mathbf{s_M} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MM} \end{bmatrix} \begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \\ \vdots \\ \mathbf{x_M} \end{bmatrix} \tag{1b}$$

The sources, $\mathbf{x}$, can be reconstructed if we somehow find the mixing matrix $\mathbf{A}$. Then figuring out $\mathbf{x}$ is as trivial as finding out the inverse matrix of $\mathbf{A}$, say $\mathbf{W}$.

$$\mathbf{x} = \mathbf{W}\mathbf{s} \tag{2}$$

Here, we would be considering a noise-free model as modeling the noise-free separation is difficult by itself.

What is independence?

One of the biggest challenges that we are going to face is the fact that both $\mathbf{A}$ as well as $\mathbf{x}$ are unknown to us. To simplify the matter, we would be assuming that the actual signals are *non-Gaussian* as well as *statistically independent* to one another, i.e.,

$$\mathbf{E}|\mathbf{x}\mathbf{x}^{\mathbf{T}}| = \mathbf{I} \tag{3}$$

As an example we can consider two uniformly distributed random variable, x_1 & x_2. Here, we can multiply $\mathbf{x} = [x_1 x_2]^T$ (*Fig. 1*) with any random $2 * 2$ matrix. The matrix transforms the vector space $\mathbf{x}$ by scaling and rotating the vector space (*Fig. 2*). We can see why the separation of the two variables would be easier if they were orthogonal or as uncorrelated as possible. However one must not equate independence with uncorrelated-ness. Two variables can be uncorrelated if they satisfy the condition:

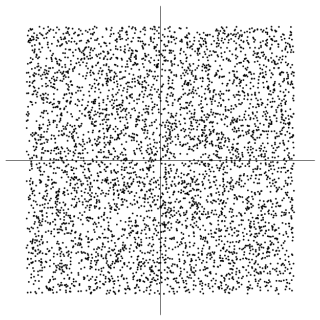
Fig. 1. Vector space for two uniformly distributed random variables x_1 & x_2

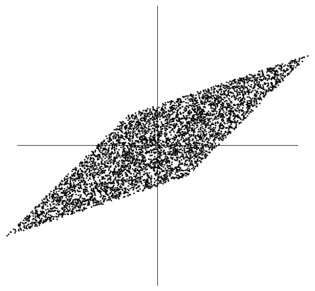
Fig. 2. Vector space after multiplying with mixing matrix

$$E\{y_1 y_2\} = E\{y_1\}E\{y_2\} \tag{4}$$

But to be independent, the 2 variables must also satisfy the condition:

$$E\{h_1(y_1)h_2(y_2)\} = E\{h_1(y_1)\}E\{h_2(y_2)\} \tag{5}$$

Here, h_1 & h_2 are any 2 real functions.

Why do Gaussian distributions fail?

Let $\mathbf{s}$ be Gaussian and $\mathbf{A}$ be orthogonal and uncorrelated, the the joint distribution of x_1 and x_2, which themselves are Gaussian, is given by:

$$p(x_1, x_2) = \frac{1}{2\pi} \exp\left(-\frac{(x_1 + x_2)^2}{2}\right) \tag{6}$$

We can now see in *Fig. 3* that there is no sense of direction

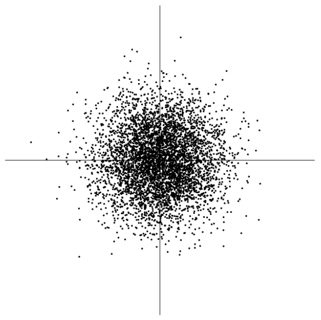

Fig. 3. Joint probability distribution for x_1 & x_2

and it can be proven that: any orthogonal transformation for x_1 & x_2 is going to be Gaussian itself.

II. Methodology

Proof:

We start by assuming that the vector $\mathbf{s}$ is independent. Let w be a vector to be determined. Then, if w is actually one of the rows from the matrix $\mathbf{A^{-1}}$, we would end up getting one of the independent component $\mathbf{x_i} = w^T * \mathbf{s}$. Now how can we make use of the *Central Limit Theorem* such that the vector w is one of the rows of $\mathbf{A}$?

For that, let's first define $z = \mathbf{A^T}w$. Then we have $y_i = w^T\mathbf{s} = w^T\mathbf{Ax} = z^T\mathbf{x}$. Thus y_i is a linear combination of $\mathbf{x}$, with weights given by z_i. Therefore, y_i is more *Gaussian* than $\mathbf{x_i}$ and becomes least *Gaussian* when it in fact becomes $\mathbf{x_i}$ itself.

Therefore we could say that a matrix w maximizes the *non-Gaussianity* of $w^T\mathbf{x}$.

Negentropy:

To Maximize non-Gaussianity, we need to first measure non-Gaussianity by using negentropy. The entropy of any random variable $\mathbf{y}$ is given by:

$$\mathbf{H}(\mathbf{y}) = -\int f(\mathbf{y}) \log(f(\mathbf{y}))\, dy$$

The entropy of a *Gaussian* random variable is always *zero*. The negentropy is defined as the differential form between y and y_{Gauss}:

$$\mathbf{J}(\mathbf{y}) = \mathbf{H}(\mathbf{y_{Gauss}}) - \mathbf{H}(\mathbf{y}) \tag{7}$$

Therefore, negentropy of any *non-Gaussian* random variable is always negative and *zero* in the case of a *Gaussian* random variable.

Algorithm:

Some pre-processing is always good to simplyfy the further implementation of the algorithm.

Centering: Before starting, the received signal must be center along *zero*. This can be done by:

$$\mathbf{s} = \mathbf{s} - \mu_{\mathbf{s}} \tag{8}$$

Whitening: It's important to whiten the signal before the application of ICA. We linearly transform the matrix $\mathbf{s}$ such that we obtain a white matrix $\widetilde{\mathbf{s}}$. This new matrix $\widetilde{\mathbf{s}}$ is uncorrelated or:

$$\mathbf{E}(\widetilde{\mathbf{s}}\widetilde{\mathbf{s}}^T) = \mathbf{I} \tag{9}$$

The whitened matrix $\widetilde{\mathbf{s}}$ is given by:

$$\widetilde{\mathbf{s}} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^T\mathbf{s} \tag{10}$$

where, $\mathbf{E}$ is the *eigenvector matrix* and the $\mathbf{D}^{-1/2}$ matrix is the diagonal matrix of *eigenvalues* element-wise square-root.

Higher the eigenvalue, the more independent the orthogonal basis (corresponding eigenvector) it has and vice-versa.

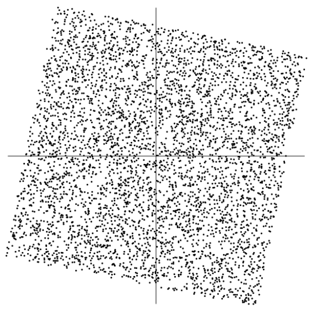

Fig. 4. Whitened distribution for the random transformed vector-space in *Fig. 2*

We can see that in *Fig. 4*, the vectors are now orthogonal (uncorrelated) after *whitening*, but the rotation is still not correct.

ICA: We implement the algorithm by finding out the weights for each component one-by-one. The following steps are followed for extracting C ($C \leq M$) independent sources:

1) Initialize a random weight vector, $\mathbf{w}_i$, of size M.
2) Let $\mathbf{w}_i^+ = \mathbf{E}(\mathbf{s}g(\mathbf{w}_i^T\mathbf{s}) - \mathbf{E}(g'(\mathbf{w}_i^T\mathbf{x}))\mathbf{w}_i$
3) Let $\mathbf{w}_i = \mathbf{w}_i^+/||\mathbf{w}_i^+||$
4) If not yet converged, go to 2

Once each $\mathbf{w}_i$ has been calculated, the matrix $\mathbf{w}$ is then created by concatenating all of the $\mathbf{w}_i$ and then the desired output is obtained by:

$$\mathbf{x} = \mathbf{w}^T \mathbf{s} \tag{11}$$

where, $\mathbf{x}$ is of dimension $C * N$, where C ($C \leq M$) is the number of independent sources to be extracted.
The implementation of the above algorithm can be found on my GitHub: github.com/theZeenX1/BSS-using-FastICA-in-C.

III. Results

Waveform Test:

Here, the waveforms were digitally generated and mixed, where each waveform had a sample size of 100. The actual signals ($\mathbf{x}$) is given in *Fig. 5*, the mixed signals ($\mathbf{Ax}$ or $\mathbf{s}$) is given in *Fig. 6*, and finally the FastICA output ($\mathbf{w}^T\mathbf{s}$)is given in *Fig. 7*.

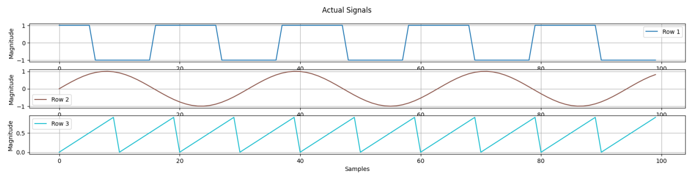

Fig. 5. Actual source signal ($\mathbf{x}$)

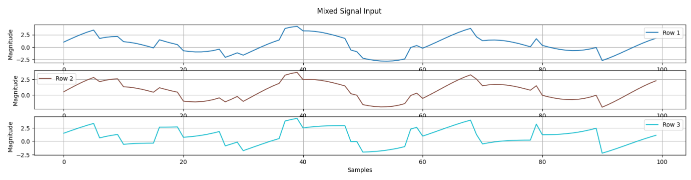

Fig. 6. Mixed Signal ($\mathbf{s}$)

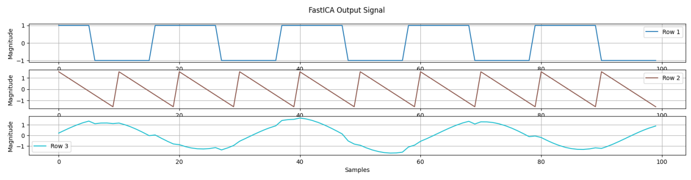

Fig. 7. FastICA output ($\mathbf{w}^T\mathbf{s}$)

Audio Test:

Here, 9 audio samples (.wav) files (8kHz Sampling Frequency, 8kbps Bit Rate, Mono-channel) were digitally mixed and the algorithm was applied to separate the files:

As we can see in *Fig. 8* and *Fig. 9*, we can't figure out the energies (variances) of the given signal. The reason is that, both $\mathbf{x}$ and $\mathbf{A}$ being unknown, any scalar multiplier in one of the sources $\mathbf{s}_i$ could always be cancelled by dividing the corresponding column $\mathbf{A}_i$ of $\mathbf{A}$ by the same scalar.

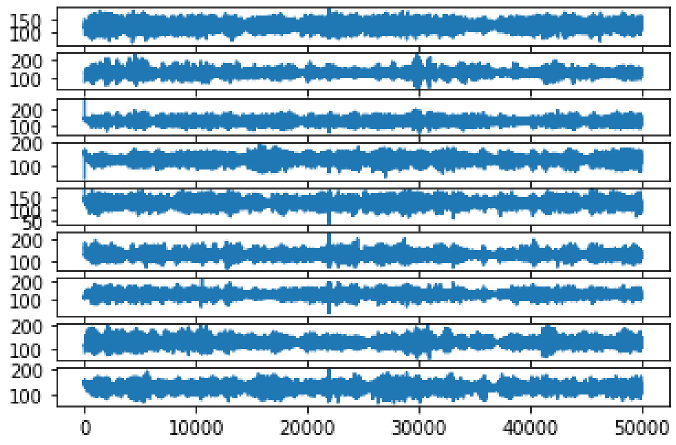

Fig. 8. Input Audio Files

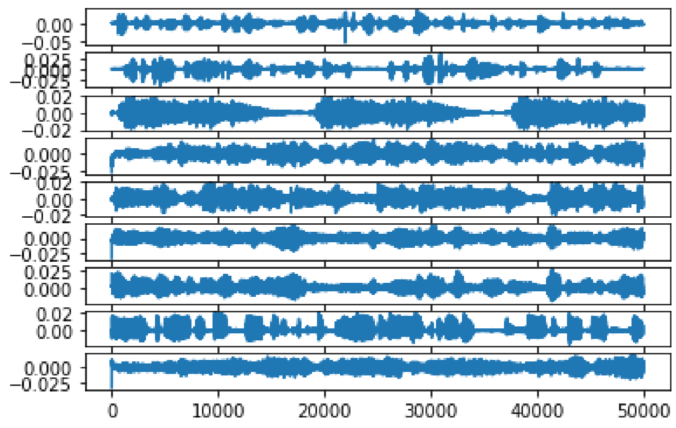

Fig. 9. Output Audio Files

IV. Conclusion

Independent Component Analysis (ICA) is a powerful statistical tool that transforms observed random data into components with both maximum independence and interesting distributions. In simpler terms, it uncovers hidden patterns in data. The process involves optimizing functions like maximizing non-Gaussianity or using classical methods such as maximum likelihood estimation. Surprisingly, these methods are roughly equivalent. The FastICA algorithm efficiently carries out the estimation process. ICA finds practical use in various fields like audio processing, biomedical signal processing, image processing, telecommunications, and econometrics. Overall, ICA is a versatile approach for extracting meaningful information from complex datasets.

Referenes

1) A Radar Signal Sorting Algorithm Based on Improved FastICA - Yonghua He, Xin Sang, Yonggang Li (2023)
2) Independent Component Analysis: A Tutorial - Aapo Hyvärinen and Erkki Oja (1999)

3) fICA: FastICA Algorithms and Their Improved Variants - Jari Miettinen, Klaus Nordhausen, and Sara Taskinen (2018)
4) A Tutorial on Independent Component Analysis - Jonathon Shlens (2014)
5) scikit-learn/sklearn