Independent Component Analysis

By Aracely A[illegible], Pablo F[illegible], Corey Warren, and Ericka G[illegible]

School of Computer Science and Engineering, [illegible]

Abstract

Independent Component Analysis is a computational method for separating a multivariate signal into subcomponents. In other words, ICA decomposes complex data into subcomponents. Our project focuses on taking wave data made of several overlain waves, and separating them using Independent Component Analysis. This entails several operations involving matrices, including: centering the signal, whitening the signal, using de-mixing matrices, normalizing the data, looping an algorithm to separate the signals, and taking the dot product of our de-mixing matrix with our now-whitened signal matrix. Through this process, the input signals start as one mixed signal, but are output as separate signals.

1 Introduction

ICA can be a confusing topic without the right context. So, imagine you are out with a bunch of different friends, talking to one another. Everyone in your group is talking simultaneously, however, you only want to listen to the friend directly in front of you.

As a human, the ability to do this is second nature. Humans do not even need to consciously think about how to focus on just one voice or sound.

However, with a computer, the process of differentiating between different voice signals requires some advanced mathematical logic. This is a situation in which Independent Component Analysis can be used. ICA can separate all the different voices into their own ".wav" files. This will enable your computer to not only listen to the correct person, but also to differentiate between any of the voices around it–given that the mixed signal input is suitable.

2 Related work

There were several papers floating about online regarding ICA, though it was difficult to find much on practical applications of Binary Independent Component Analysis. Thus, we had to settle with regular ICA. One useful resource we had was a coding-based article, aptly titled "Independent Component Analysis (ICA) In Python," published on 'towardsdailyscience.com,' which served as the basis for our code.

This article served as a great resource in getting our code started. It helped us understand how to approach the problem of ICA from a programming perspective, including the creation of signals, different math-related python functions, and how to work with scipy and matplotlib.

However, simply following the guide was insufficient to understanding how exactly the code functioned, and it did not explain the results we later received upon changing the initial values of our functions. It took re-arrangement of the code, cutting out parts unnecessary to our project, additional commenting, and creation of new code to make the project work to our liking.

3 Method

The overall goal of the project is to separate multiple signals, but this process requires some preparation.

(1) The first step is to center X by subtracting the mean. Then, center the

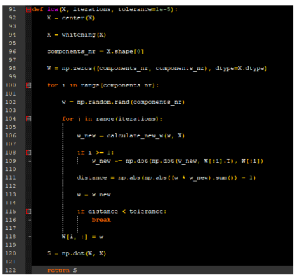

Figure 1: Independent Component Analysis function in Python 3.0 (supporting functions not shown)

data around 0 and subtract the mean from each dimension and add it back to the estimation of S at the end of the problem.

(2) Whiten the signal data in matrix x. This means to transform your data to remove any possible correlation. This will transform X into X(hat).

(3) Choose a random initial value for the de-mixing matrix W.

(4) Calculate the new value for W.

(5) Normalize W.

(6) Check whether the algorithm has converged or not. If it has not, then return to step 4 and repeat the process.

(7) Finally, after sufficient loops, take the dot product of W and x to get the independent source signals.

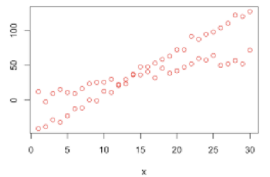

Figure 2: Example of a data source with one signal.

$$I_1 = [1],\ I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},\ \cdots,\ I_n = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}.$$

$$\tilde{x} = ED^{-1/2}E^T x \qquad D = \begin{pmatrix} \lambda_1 & 0 & 0 & ... \\ 0 & \lambda_2 & 0 & ... \\ 0 & 0 & \lambda_3 & ... \\ ... & ... & ... & ... \end{pmatrix}$$

for 1 *to the number of components* :

repeat until $w_p^T w_{p-1} \approx 1$:

$$w_p = \frac{1}{n}\sum_i^n X g(W^T X) - \frac{1}{n}\sum_i^n g'(W^T X) W$$

$$w_p = w_p - \sum_{j=1}^{p-1} \langle w_p^T w_j \rangle w_j$$

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

$W = [w1, w2...]$

4 Experiment

(1) To differentiate between independent sources, a source must be chosen to extract data from. Then, the data should be centered around 0.

(2) Whiten your data to remove any correlations. Transform X to X(hat). The whitened signal of covariance matrix will equal to the identity matrix.

x-hat is an eigenvalue that is the decompose of the covariance matrix, while D equals the diagonal matrix of eigenvalues.

(3-6, Looping) For these steps we need to determine W, Calculate new value for W, Normalize W, then check it but if it has no convergence then we need to go back to step 4.

(7) For the last step we need to take the dot product of W and X to get the independent signal sources(S).

S = Wx

The code for this experiment is written in Python. It follows the outlined steps above. It generates three different wave signals: a sine wave, a square wave, and a sawtooth wave. It mixes them in order to present a problem: our signals are mixed together and we need to have them separated. Using our independent component analysis function (and several supporting functions provided by libraries such as numpy and scipy), we are able to re-separate these artificially mixed signals.

5 Evaluation and Discussion

Pictured in Figure 3 are the visual results of our Python code, plotted using the Python matplotlib library. Our program begins by creating three wave signals, which are pictured in the second row, titled "real sources," while the first row consists of the artificially mixed matrix x. This x matrix is visualized in its pre-ICA state, meaning this is what the signals look like before they are de-mixed. In this state, it is not easily discernible to the

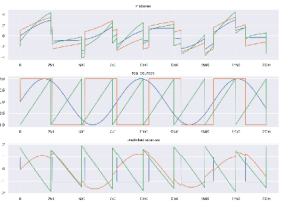

Figure 3: Independent Component Analysis function (supporting functions not shown)

human eye as to what type of signals are present in the mixture.

This brings us to row three; titled "predicted sources," these are the signals we obtain after ICA. The signals are fairly clean and have only mild distortions when compared to their original signals. However, upon closer inspection, we found that something was off about these results. It only becomes clear when you realize that the colors are switched.

This was unexpected. It seems that the signals seem to have switched spots. Basically, what this means is that our program was not able to perform its ICA algorithm and spit out the signals in a predictable manner. Rather, these signals could perhaps be output in any arbitrary order, entirely dependent on how they are mixed before ICA. Thus, in a real-world application where sound signals must be separated, a human may have to step in and discern which signals are which. If not, then additional algorithms would need to be added to figure out which signals are which.

Upon further investigation as to why this might be, we realized that this makes perfect sense, considering that the signals were randomly mixed together in an arbitrary manner. Of course ICA pays no mind as to which signal we labelled "s1," "s2," or "s3." It has no way of knowing that for certain, especially if the matrix x is mixed very well.

Additionally, based on our readings, there are several known aspects of ICA that support this hypothesis:

1) "Since ICA is dealing with clouds of points, changing the order in which the points are plotted has virtually no effect on the outcome of the algorithm."

2) "Even when the sources are not independent, ICA finds a space where they are maximally independent." This could hint that ICA's primary focus is not on preserving signal order, or column order in the resulting de-mixed signal matrix, but rather, it focuses on maximizing the clarity and independence of each individual signal, regardless of the order they were initially in.

3) "Changing the channel order has no effect on the outcome of the algorithm." Channels in this case may be analogous to our signals and how they are organized in our matrix x, which is de-mixed by our de-mixing matrix W.

One less shocking revelation is that the sawtooth wave has been flipped horizontally. Initially, the sawtooth wave would rise linearly, then drop quickly, forming the sawtooth pattern. Now, it does the opposite. It falls linearly, then pops up quickly, forming a chronologically reversed sawtooth wave. This is of little consequence, all things considered. To the human ear it would be indiscernible.

6 Conclusion

While our program does a good job of separating the signals, it does not necessarily do so in a neat, predictable way. This is because the signals may switch spots due to the mixing process, be it artificial mixing, like in our experiment, or true analog mixing that happens in the real world. Thus, after ICA is applied, further intervention is needed to label all of the independent signals correctly.

References

[1] Brown, Glen D., et al. "Independent Component Analysis at the Neural Cocktail Party." Trends in Neurosciences, vol. 24, no. 1, 2001, pp. 54–63., doi:10.1016/s0166-2236(00)01683-0.

[2] Hyvärinen, Aapo, and Erkki Oja. "Independent Component Analysis: Algorithms and Applications." Neural Networks, 2000, doi: http://ric.uthscsa.edu/personalpages/lancaster/SPM_Class/Lecture_18/ica_tutorial.pdf. \IndependentComponentAnalysis."YouTube, YouTube, 28Apr.2020, www.youtube.com/watch?v = FJ5jd3D1UP4.

[3] "Independent Component Analysis." YouTube, YouTube, 28 Apr. 2020, www.youtube.com/watch?v=FJ5jd3D1UP4.

[4] Maklin, Cory. "Independent Component Analysis (ICA) In Python." Medium, Towards Data Science, 22 Aug. 2019, towardsdatascience.com/independent-component-analysis-ica-in-python-a0ef0db0955e.

[5] Seo, Jae Duk. "[Paper Summary] An Introduction to Independent Component Analysis: InfoMax and FastICA Algorithms." Medium, Towards Data Science, 5 Sept. 2018, towardsdatascience.com/paper-summary-an-introduction-to-independent-component-analysis-infomax-and-fastica-algorithms-7b44d18ab393.