PowerICA

1 Introduction

PowerICA is a stable ICA algorithm, which is especially useful when the sample size is not orders of magnitude larger than the data dimensionality [1]. This is the finite-sample regime in which the fixed-point FastICA algorithm [2] is often reported to have convergence problems [3].

This MATLAB package is an implementation of Algorithm 1 proposed in paper below.

S. Basiri, E. Ollila and V. Koivunen, "Alternative Derivation of FastICA With Novel Power Iteration Algorithm," in IEEE Signal Processing Letters, vol. 24, no. 9, pp. 1378-1382, Sept. 2017.

If you use this function in your publication please cite the paper using the above citation info.

2 The package

The PowerICA package contains the following files:

README.pdf: This file.

PowerICA.m: The main function.

parnode.m: Auxiliary function, called by the main function in *parallel* mode.

Example.m: A synthetic example of utilizing the PowerICA method in extracting independent source signals from their observed mixture recordings.

demosig.m: Auxiliary function, called by Example.m to generate test source signals (independent components). This function is taken from the original FastICA MATLAB package.

3 Download

Download the package and extract the files into a folder with "full control" permission. Set the Matlab home directory to the above folder,
e.g. cd C:\localwork\Matlab\PowerICA.

4 Syntax

[W , flg] = PowerICA(X, nonlin, W0, mode)

4.1 Input parameters

X: A real valued $d \times n$ array of mixture recordings, where d and n denote the dimensionality and the number of observations respectively. Data X should be centered and whitened.

nonlin: ICA nonlinearities. It can be either a single string or a $d \times 1$ array of strings. The following nonlinearities are supported.
tanh, pow3, gaus, skew, rt06, lt06, bt00, bt02, bt06, bt10, bt12, bt14, bt16, tan1, tan2, tan3, tan4, gau1, gau2, gau3.
We refer the reader to [4] for detailed description of the other nonlinearities besides the standard ones (*tanh, gaus, pow3, skew*). The default value is *tanh.*

W0: An orthogonal $d \times d$ matrix used as the initial start of the algorithm.

mode: Can be set either to *serial* or *parallel.* The *serial* mode is used when only one computing node is available or the dataset is of small size. The default mode is *serial.* The *parallel* mode runs two parallel Matlab instances on different CPU cores. The two instances communicate via a Java socket. Make sure you have installed and updated Java on your system. In order to use the *parallel* mode in MacOS, line 81 of the PowerICA function should be edited according to your installed Matlab version. For example:
!/Applications/MATLAB_R2017a.app/bin/matlab -r parnode &

4.2 Output parameters

W: PowerICA estimate of orthogonal $d \times d$ demixing matrix.

flg: Returns 1, when the algorithm has converged successfully and 0 when the algorithm has failed to converge.

5 Example

Open the Example.m file in Matlab and follow the steps of the example. Function *demosig.m* generates $n = 500$ samples of $d = 4$ ICs shown in Figure 5. The data consists of $p = 5$ random mixture of the ICs shown in Figure 5. The data are first whitened and centered. Then, PowerICA algorithm is used to extract ICs from the mixtures up to the sign and permutation ambiguities. The extracted ICs are shown in Figure 5

References

[1] S. Basiri, E. Ollila, and V. Koivunen, "Alternative derivation of FastICA with novel power iteration algorithm," *IEEE Signal Processing Letters*, vol. 24, no. 9, pp. 1378–1382, Sept 2017.

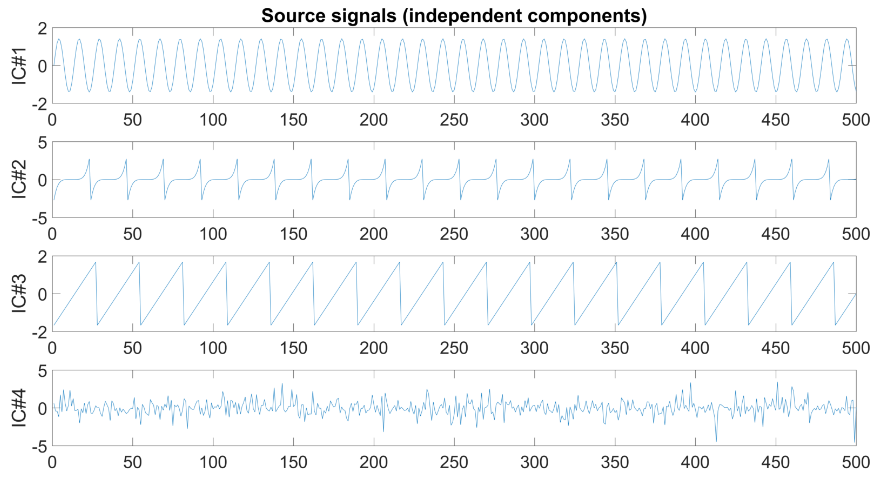

Figure 5.1: Source signals (independent components)

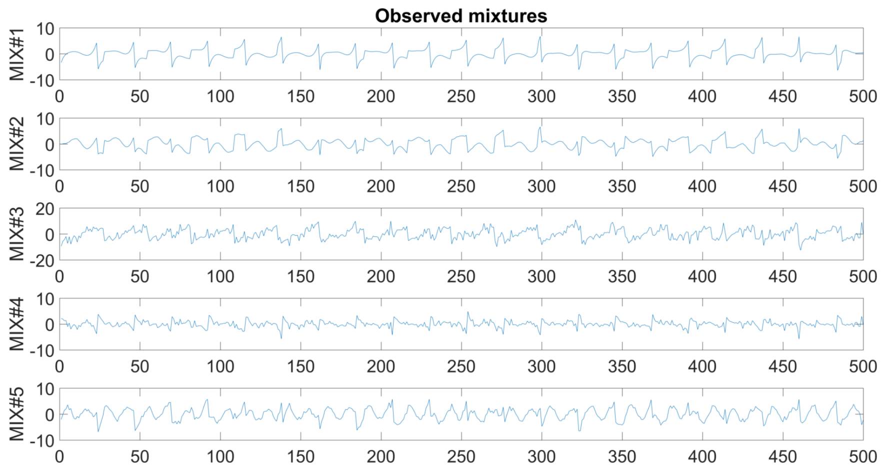

Figure 5.2: The observed data (Random mixtures)

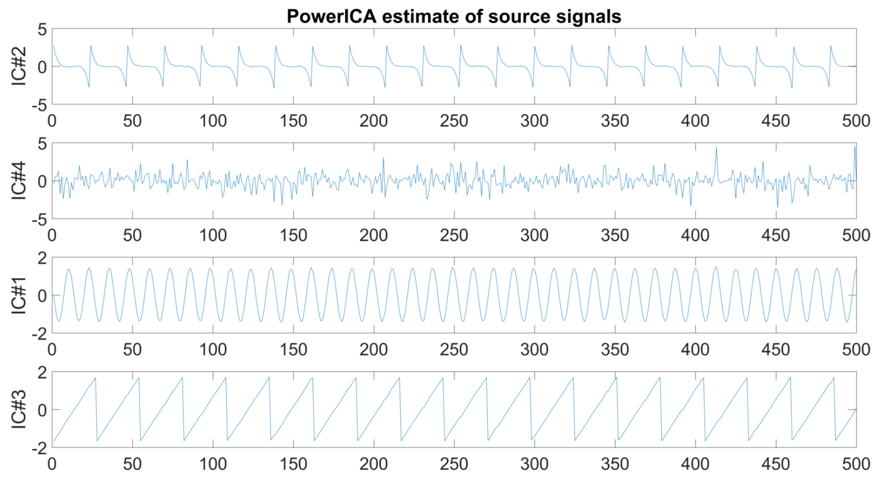

Figure 5.3: PowerICA estimate of the ICs

[2] A. Hyvärinen and E. Oja, “A fast fixed-point algorithm for independent component analysis,” *Neural Computation*, vol. 9, pp. 1483–1492, 1997.

[3] P. Tichavsky, Z. Koldovsky, and E. Oja, “Performance analysis of the FastICA algorithm and Cramér-Rao bounds for linear independent component analysis,” *IEEE Transactions on Signal Processing*, vol. 54, no. 4, pp. 1189–1203, April 2006.

[4] J. Miettinen, K. Nordhausen, H. Oja, and S. Taskinen, “Deflation-based FastICA with adaptive choices of nonlinearities,” *IEEE Transactions on Signal Processing*, vol. 62, no. 21, pp. 5716–5724, Nov 2014.