PWC-ICA: A MATLAB toolbox for BSS on vector signals.

Kenneth Ball

January, 2015

Pair-wise Complex Independent Component Analysis (PWC-ICA) is an algorithm for BSS of sequentially ordered signal data that incorporates time-ordering by mapping real-valued signals to a complex vector space, performs complex ICA, and reinterprets the complex demixing matrix in the original, real signal space.

The pwcica-toolbox package includes MATLAB functions that allow PWC-ICA to be performed from the MATLAB command prompt, or integrates with EEGLAB as a plugin with a gui.

Notice: Technical details on the PWC-ICA algorithm may be found in the paper:
Ball, K.R., Bigdely-Shamlo, N., Mullen, T., Robbins, K. [2015]. PWC-ICA: A Method for Ordered Blind Source Separation with Application to EEG. *Submitted for internal review.*

The included code is free to use, modify, and distribute, with the following caveats.

1. If you use PWC-ICA or a derivate in your own work, we ask that you include the above citation.

2. Modifications and/or redistributions of the code included in this toolbox (excluding any packaged Complex ICA methods) must retain this notice in the README or equivalent file, and must include the above citation.

0.1 A note on Complex ICA packages

`pwcica` depends on separate functions that perform complex ICA on complex vector-valued signals. These may be placed in the `pwcica\ComplexICA` subdirectory, or can be included as an optional function handle, described below. In addition to the above citation, authors that use PWC-ICA should cite the complex ICA approach that they use.

Functionality for the following Complex ICA implementations, which are freely available on the Internet, is built in to the PWC-ICA toolbox.

- Complex improved FastICA (`FicaCPLX`), the default Complex ICA method. Available from Petr Tichavský at `http://si.utia.cas.cz/downloadPT.htm` and `http://si.utia.cas.cz/CODE/FicaCPLX.html`.

- Robust ICA (`robustica`), available from Vincent Zarzoso at `http://www.i3s.unice.fr/~zarzoso/robustica.html`.

- Complex Entropy Bound Minimization (`ebm`), available from Li and Adali from UMBC at `http://mlsp.umbc.edu/ica_ebm.html`.

1 Installation

1.1 Unpackaging

After downloading the pwcica-toolbox package, move the contents of the package to a directory in your MATLAB path. Alternately, if you would like to use the pwcica EEGLAB plugin, move the contents of the package to the ”plugins” subdirectory of your eeglab directory. For example, the file ”pwcica.m” should appear as: ”../MATLAB/eeglab/plugins/pwcica/pwcica.m”.

1.2 Requirements

PWC-ICA requires a valid MATLAB installation. It has been tested on MATLAB R2014a, but we expect reasonable backward compatibility. To use the Savitzky-Golay smoothing option, `pwcica` requires the MATLAB Signal Processing Toolbox.

2 Using PWC-ICA

2.1 Command-line interface

PWC-ICA is called from the MATLAB command-line using the function `pwcica`. The basic syntax is:

```
>> W = pwcica(data,'key1',val1,...,'keyN',valN);
```

where:

- `data` is an $[n, N]$ or $[n, N, K]$ array of real valued signal data.
- `W` is the output, an $[n, n]$ real demixing matrix.
- `'key',val` is an optional key-value pair.

2.1.1 Input:

- `data` is an $[n, N]$ or $[n, N, K]$ array of real valued signal data. At each time frame/observation, the signal is an n-dimensional real vector. There are N observations divided into K epochs. Signal data may either be input unepoched, where $K = 1$ for a total of N observations, or epoched, where $K > 1$ for a total of NK observations.
- (OPTIONAL) `'key',val` are optional key-value pairs.
  - `'ComplexICAMethod' , ( ['fastICA'] 'ebm' 'robust' @customfunction )`
    The complex ICA method to be used in complex phase space.
    ALTERNATELY:
    The value of `ComplexICAMethod` may be specified as a function handle for a custom complex ICA method satisfying:

`>>     complexW = @customfunction(complexData);` where `complexData` is the $[n,N]$ array of complex signal data, and `complexW` is the output $[n,n]$ complex demixing matrix. Note, pwcica automatically (pseudo) whitens complexData input.
EXAMPLE:
Given a complex ICA method:
`>>     W = complexICA(data,options);`
and you can define a handle as:
`>>     f = @(x) complexICA(x,options);`
and feed the key-value pair `...,'ComplexICAMethod',f,...` into `pwcica`.
Note, you may need to write a separate wrapper function if complexICA has multiple outputs.

– `'TimeInvariant' , ([1] , 0)`
  If set to 0, utilizes a transformation that scales the difference (tangent) vectors according to the `'SamplingRate'`. Default set to 1; in this case the mapping V is time-invariant and is a complex linear structure.
– `'SamplingRate' , ([1], any real)`
  The sampling rate of observations. Set `'TimeInvariant'` to 0 in order to implement a sampling rate.
– `'EpochLength' ([K], an integer divisible by N if K = 1)`
  Define epoch length in case data should be epoched but is input as stream. Ignored if $K \neq 1$.
– `'OptObj' , (['pow3'] , 'hyvar')`
  Specify optimization objective for `FicaCPLX`.
– `'Level' , ([1] , any natural number > 0)`
  Width of time interval for sum/difference mapping into complex phase space.
– `'SGSmoothing' , ([0] , 1)`
  Turn on to use Savitzky-Golay smoothing for mapping time series data to complex phase space.
– `'SGWindow' , ([11], any odd natural number)`
  Specifiy window for Savitzky-Golay smoothing.
– `'SGOrder' , ([5], any natural number > 0)`
  Specify order of polynomial for Savitzky-Golay filter to fit.
– `'ComplexICAPath' , ([working directory\ComplexICA], string, or []);`
  Optionally specify the directory that external complex ICA algorithms are stored in. If not set, `pwcica` assumes that algorithms are located in a subdirectory caled `\ComplexICA` of `pwcica`'s working directory. Value may be set to a string of the full path name of the relevant directory. If value is empty, `pwcica` will NOT add anything to MATLAB's search path.

2.1.2 Output:

- `W` is the demixing matrix, an $[n,n]$ array. Note, `W` automatically incorporates the whitening operations internally computed in `pwcica`.

2.1.3 Examples

Suppose `x` is an $[32, 256, 500]$ array of real doubles, representing 500 epochs of a 32-dimensional signal sampled at 128 Hz, where each epoch is 2 seconds long (hence, $N = 2 \cdot 128 = 256$). Then the following are examples of `pwcica` usage.

- `>> W = pwcica(x);`

  Default, time invariant method, with step size 1 between observation pairs.

- `>> W = pwcica(x,'IncludeTime',1,'SamplingRate',128);`

  Time-variant, sampling rate manually specified.

- `>> W = pwcica(x,'Level',4);`

  Complex vectors are formed from observation pairs separated by 4 time-steps.

- `>> W = pwcica(x,'ComplexICAMethod','ebm');`

  EBM used for Complex ICA instead of default (FicaCPLX).

- `>> W = pwcica(x,'SGSmoothing',1,'SGWindow',7,'SGOrder',3);`

  S-G filters are used to map to the complex vector space.

2.2 EEGLAB Plugin

If the `\pwcica` directory and its comments are placed in the `..\plugins\` directory of your EEGLAB install, then upon opening eeglab the PWC-ICA toolbox will be accessible from the "Tools" context menu of the EEGLAB GUI. If a dataset with valid signal data is loaded, then the option "Run PWC-ICA" is available and opens a sub-menu with two options: "Pair-wise Complex ICA" and "PWC-ICA w/ Savitzky–Golay Smoothing." Each option opens a separate dialogue box.

2.2.1 Pair-wise Complex ICA

- "Time-Invariant" :: Radio button, default checked. If on, `'TimeInvariant'` is set to 1. Else, `'TimeInvariant'` is 0.
- "Step-size" :: Text input, $n \in \mathbb{N}$, default is $n = 1$. Sets `'Level'` to n, defining the lag, or step-size, between pairs of observations for PWC-ICA.
- "Complex ICA Method" :: Text input, default is "fastiCA." Sets `'ComplexICAMethod'` to the specified value, telling `pwcica` which complex ICA method to use. Can also input a predefined function handle, as described above.

2.3 PWC-ICA w/ Savitzky–Golay Smoothing

Note: selecting "PWC-ICA w/ Savitzky–Golay Smoothing" will automatically set the option `'SGSmoothing'` to 1.

- "Time-Invariant" :: Radio button, default checked. If on, `'TimeInvariant'` is set to 1. Else, `'TimeInvariant'` is 0.

- "SG Filter Order" :: Text input, $n \in mathbbN$, default is $n = 3$. Sets `'SGOrder'` to the value of n.
- "SG Window Length" :: Text input, n must be odd and greater than the value of "SG Filter Order," default is $n = 7$. Sets `'SGWindow'` to the value of n.
- "Complex ICA Method" :: Text input, default is "fastiCA." Sets `'ComplexICAMethod'` to the specified value, telling `pwcica` which complex ICA method to use. Can also input a predefined function handle, as described above.