PRACTICAL WORK # 3 - BLIND SOURCE SEPARATION

We strongly advise the use of either **Matlab** or **Python** for these practical works.
Participants that opt for Python will find the following modules helpful :

- ipython
- scipy/numpy
- matplotlib
- scikit-learn
- pyfits.

Please note that for python codes, all the necessary routines are contained in the module pyBSS.py

Most of them can be set up with easily using standard porting tools (apt-get, macport ... etc).

All the necessary material is available at http://jbobin.cosmostat.org/teaching/master-2-mva/

1. Pre-processing and principal component analysis - requires course #5

• Assuming that you have access to multichannel data $\mathbf{X} \in \mathbb{R}^{m \times t}$ (i.e. m observations of t samples), **implement** a code that :

(1) *performs the whitening of the data* $\mathbf{X}$

(2) *provides its best rank-n approximation according to the Euclidean norm.*

• *Apply the PCA to a random mixture of Gaussian sources. For that purpose, the code GenerateMixture.m / .py allows generating Gaussian sources and mixtures. Does PCA work in this case ?*

2. Independent component analysis - requires course #5

• The code GenerateMixture.m / .py allows to generate mixtures of uniform random variables.

(1) *Apply the FastICA algorithm on* 2 *mixtures of* 2 *uniformly distributed sources. Could you comment on the results ?*

(2) *Evaluate the performances of the FastICA algorithm when the condition number of the mixing matrix varies (for instance, [1,2,10,50,100]).*

The evaluation can be carried out by using the code Eval_BSS.m / .py, which returns the mixing matrix criterion that evaluates a discrepancy between the input and estimated mixing matrices (i.e. the lower it is, the better the separation is). For more reliable results, you can compute the average mixing matrix criterion from several simulations.

(3) *The file ChandraSims.mat contains simulated data from X-ray observations. The data X are composed of* 8 *sources. Apply the FastICA method to these data and evaluate the ability of ICA to estimate the sources and the mixing matrix. Could you comment the results ?*

3. **Sparse blind source separation - requires course #6**

• *Apply the GMCA algorithm to uniformly distributed sources (e.g.* 2 *mixtures of* 2 *sources). Could you comment on the results ?*

• *The code GenerateMixture.m / .py allows to generate mixtures of sparse random variables.*

(1) Apply the GMCA algorithm to 2 *mixtures of* 2 *sparse sources. Could you comment on the results ?*

(2) Add homoscedastic Gaussian noise with standard deviation σ *(i.e. the noise standard deviation is the same for every data channel) to the previous mixtures of sparses sources.*

(3) Apply the GMCA algorithm and the FastICA algorithm to these data for various values of the signal-to-noise ratio ; for instance SNR $= [40, 30, 20, 10]$ *dB.*

(4) Evaluate the performances of both methods using the mixing matrix criterion when the noise level varies. For more reliable results, you can compute the average mixing matrix criterion from several simulations. Comment the results.

• *Going further with sparse BSS.*

(1) Implement a code based on proximal algorithms to tackle the problem

$$\min_{\mathbf{A},\mathbf{S}} \sum_{i=1}^{n} \lambda_i \|s_i \mathbf{\Phi}^T\|_1 + \frac{1}{2}\|\mathbf{X} - \mathbf{AS}\|$$

Notice that if $\mathbf{\Phi}$ *is assumed to be orthogonal, this problem can be recast in a simpler problem in the sparse domain.*

(2) Apply the resulting algorithm and GMCA algorithm to the Chandra data. Comment the results.

Appendix : how to use the module pyBSS.py

GenerateMixture :

— Inputs

(1) n - number of sources (default is 2*)*

(2) t - number of samples per observation/sources (default is 1024*)*

(3) m - number of observations (default is 2*)*

(4) SType - type of sources - 1 for Gaussian, 2 for uniform and 3 for sparse (default is 1)

(5) noise_level - noise level in dB (default is 120)

— Outputs

(1) X - the data

(2) A - the mixing matrix

(3) S - the sources

perform_FastICA :

— Inputs

(1) X - input data

(2) n - number of sources

— Outputs

(1) A - the mixing matrix

(2) S - the sources

perform_GMCA :

— Inputs

(1) X - input data

(2) n - number of sources

— Outputs

(1) S - the sources

(2) A - the mixing matrix

(3) PinvA - the pseudo-inverse of the mixing matrix

Eval_BSS :

— Inputs

(1) A0 - true mixing matrix

(2) S0 - true sources

(3) A - estimated mixing matrix

(4) S - estimated sources

— Outputs

(1) mixing matrix criterion