MLSP 2005 Competition:
Large Scale, Ill-Conditioned Independent Component Analysis With Limited Number Of Samples

Andrzej CICHOCKI
Riken, Brain Science Institute,
Lab. for Advanced Brain Signal Processing,
Wako-Shi, JAPAN

Deniz ERDOGMUS
Dept. of Comp. Science &Electrical Engineering,
Oregon Health & Science University
Portland, USA

We assume the standard linear model described in matrix form as:

$$\mathbf{X} = \mathbf{A}\,\mathbf{S},$$

where $\mathbf{X}$ is the n by T matrix representing n observations for T consecutive time instants, $\mathbf{A}$ is an n by n nonsingular mixing matrix and $\mathbf{S}$ is an n by T matrix representing sources; n is the number of sources (equal to the number of sensors) and T is the number of available samples.

It is assumed that only the matrix $\mathbf{X}$ is available, while the matrices $\mathbf{A}$ and $\mathbf{S}$ are unknown and should be estimated. The objective of this problem is to investigate the effect of increasing dimensionality n, decreasing number of samples T, increasing ill-conditioning of the mixing matrix $\mathbf{A}$ and/or increasing the level of additive noise in the sensor level for performance and reliability of independent component analysis algorithms.

The original non-negative source signals are to be generated in MATLAB as follows:

```
for k=1:n
        if rand <= 0.5
                S(k,:) = rand(1,T); %(sub-Gaussian)
        else
                S(k,:) = -log(rand(1,T)).*max(0,sign(rand(n,T)-0.5)); %(super-Gaussian)
        end
end
```

The mixing matrices are to be generated in MATLAB as follows:

```
A = rand(n)-0.5;
```

The performance is measured using the average signal-to-interference measure (SIR) which is calculated in MATLAB using:

```
G = W * A; SIR = mean(10*log10(max(G.^2,[],2)./(sum(G.*G,2)-max(G.^2,[],2))));
```

where $\mathbf{W}$ is the separation matrix such that $\mathbf{Y} = \mathbf{W}\,\mathbf{X}$, where $\mathbf{Y}$ is the matrix of separated components.

The problem consists of four sub-problems. Performance of the competing algorithms MUST be provided for ALL four sub-problems. The algorithms need not be new propositions, however, in the case of a tie, novel approaches will be announced as the winner.

1. Large scale problem

Determine the largest mixture dimension n for which the algorithm achieves an SIR greater than 15dB using T=5000 samples. The experiment must be conducted using Monte Carlo runs where for each run the sources and the mixing matrix are generated randomly as described above. The SIR>15dB condition must be satisfied in 90% of the Monte Carlo runs for a particular value of n. Results should be presented in a figure that shows the following curves: SIR_10% vs n, SIR_50% vs n, and SIR_90% vs n. In general SIR_P% is the maximum real number such that P% of the Monte Carlo SIR values for a particular n are greater than this number. Note that SIR_90% vs n should be that last curve to cross over the desired 15dB threshold as n increases.

2. Small training set problem

Determine the smallest number of samples T for which the algorithm achieves an SIR greater than 15dB for n=50. The experiment must be conducted using Monte Carlo runs where for each run the sources and the mixing matrix are generated randomly as described above. The SIR>15dB condition must be satisfied in 90% of the Monte Carlo runs for a particular value of T. Results should be presented in a figure that shows the following curves: SIR_10% vs T, SIR_50% vs T, and SIR_90% vs T. SIR_P% is described similar to sub-problem 1.

3. Highly ill-conditioned problem

Determine the highest dimension n for which the algorithm achieves an SIR greater than 15dB for T=5000. The experiment must be conducted using Monte Carlo runs where for each run the sources are generated randomly as described above. Only in this sub-problem, the increasingly ill-conditioned mixing matrix is generated randomly as $\mathbf{A}=\mathbf{RHR}^T$ where $\mathbf{H}$ = hilb(n) is the Hilbert matrix as generated in MATLAB and $\mathbf{R}$ is a random rotation matrix generated as shown below:

ind = randperm(n); theta = 2*pi*rand; i = ind(1); j = ind(2);
$\mathbf{R}$ = eye(n); $\mathbf{R}(i,i)$ = cos(theta); $\mathbf{R}(j,j) = \mathbf{R}(i,i)$; $\mathbf{R}(i,j)$ = sin(theta); $\mathbf{R}(j,i) = -\mathbf{R}(i,j)$;

The SIR>15dB condition must be satisfied in 90% of the Monte Carlo runs for a particular value of n. Results should be presented in a figure that shows the following curves: SIR_10% vs n, SIR_50% vs n, and SIR_90% vs n.

4. Noisy mixture problem

Suppose that $\mathbf{X=AS+N}$, where $\mathbf{N}$ is an n by T matrix representing additive spatially white Gaussian distributed noise. Determine the lowest SNR for which the algorithm achieves an SIR greater than 15dB for n=50 and T=5000. SNR is defined as the ratio of the average mixture power mean($E[x_i^2]$), where the mean is over mixture channels, to the noise power σ^2, converted to dB using $10\log_{10}(\text{mean}(E[x_i^2]) / \sigma^2)$. The experiment must be conducted using Monte Carlo runs where for each run the sources and the mixing matrix are generated randomly as described above. The SIR>15dB condition must be satisfied in 90% of the Monte Carlo runs for a particular value of SNR. Results should be presented in a figure that shows the following curves:

SIR_10% vs SNR, SIR_50% vs SNR, and SIR_90% vs SNR. Notice that this measure does not consider the noise corruption levels at the separated outputs, rather it is only concerned with the performance of the (inverse) model estimation.

Remarks:

1. Sources are non-negative so alternative methods to ICA such as NMF (Non-negative Matrix Factorization) or ICA with non-negativity constraints can be also implemented and tested. The source distributions are assumed to be unknown, therefore, preset fine-tuning that matches the source distributions for optimal performance is not allowed.

2. The proposed algorithms should solve any sub-problem in reasonable computation time, say, in a few minutes on a typical PC (so that Monte Carlo runs can be run by the organizing committee using submitted Matlab codes if necessary).

3. Report the results in a document (*.doc or *.pdf), where a brief description of the algorithm and appropriate references, as well as experimental results demonstrating performance on the sub-problems are included. In your submission, please also include a self-contained Matlab script of your code that is in ready-to-run condition to replicate the Monte Carlo results/figures to facilitate the replication of results if required.

4. Regarding the performance index, the user can additionally use the recently provided MATLAB package BSS_EVAL BSS_EVAL is a MATLAB toolbox to compute reliably performance measures in (blind) source separation within an evaluation framework where the original sources are available as ground truth.
Download page: http://www.irisa.fr/metiss/bss_eval/

5. Upon request, Matlab function to generate the random mixing matrix for subproblem 3 is as follows:

```
function A = generateA(n)
% Generates an nxn random mixing matrix A that has the
% same eigenvalues as an nxn Hilbert matrix...
H = hilb(n);
ind = randperm(n); theta = 2*pi*rand; i = ind(1); j = ind(2);
R = eye(n); R(i,i) = cos(theta); R(j,j) = R(i,i);
R(i,j) = sin(theta); R(j,i) = - R(i,j);
A = R*H*R';
```