NEURAL NETWORKS
Lesson 8 -
Blind Source Separation

Prof. Michele Scarpiniti
INFOCOM Dpt. - “Sapienza” University of Rome
http://ispac.ing.uniroma1.it/scarpiniti/index.htm
michele.scarpiniti@uniroma1.it

Rome, 19 November 2009
1 Introduction
   - Basic Concepts
   - The InfoMax algorithm

2 Other BSS techniques
   - Nonlinear BSS
   - Complex-value BSS
The general **mixing model** is shown by the figure

\[ x[n] = \mathcal{F}\{s[n]\} = \mathcal{F}\{s_1[n], \ldots, s_N[n]\} \]

where:
- \(s[n]\) is the vector of the \(N\) original and independent sources;
- \(\mathcal{F}\) is a **mixing operator**;
- \(x[n]\) is the vector of the observed signals, called **mixtures**.
The general **de-mixing model** is shown by the figure

\[
\begin{align*}
\mathbf{s} & \rightarrow \mathcal{F} \quad \mathbf{x} \rightarrow \mathcal{G} \quad \mathbf{u} \\
\mathbf{u}[n] &= \mathcal{G} \{\mathbf{x}[n]\} = \mathcal{G} \circ \mathcal{F} \{\mathbf{s}[n]\}
\end{align*}
\]

where:
- \(\mathcal{G}\) is the **recovering** or **de-mixing operator**;
- \(\mathbf{u}[n]\) is the source estimate.

One can use only the **statistically independence** of the sources: \(\Rightarrow\) **Independent Component Analysis (ICA)**.
Ambiguity of the solution

- If $\mathcal{F}$ has not a particular structure, separation could be not possible: in particular one can have infinity solutions $\mathcal{G}$.
- For this reason we consider only linear and instantaneous models $\mathcal{F}$ or Post Nonlinear (PNL).
- Remembering the trivial ambiguities of the ICA algorithms, it is:

$$u[n] = P \Lambda s[n]$$

where:
- $P$ is a permutation matrix;
- $\Lambda$ is a diagonal or scaling matrix.
- For the nonlinear case it is present an off-set ambiguity too.
History of ICA

- **Classic ICA**: linear mixtures
  1. ICA was born on 1986 (Herault)
  2. Organic development by 1994 (Comon, Jutten, Cardoso, etc.)
   - Statistical approach (HOS)
     - Cumulant based method (Amari & al.)
     - Kurtosis maximization (Amari & al., Jutten)
   - Information theoretic approach
     - FAST ICA (Hyvarinen & al.)
     - INFOMAX (Bell & Seynowski)
     - Maximization of Joint Entropy (ME)
     - Minimization of Mutual Information (MMI)
   - Different heuristics
     - Geometric approach (Babaieh-Zadeh)

- **Nonlinear ICA**
  1. Organic development by 1999 (Jutten & Taleb)
  2. Open problem
Different **ICA** approaches

\[ I(Y_1, Y_2) = \sum_{i=1}^{2} H(Y_i) - H(Y_1, Y_2) \]

- Minimization of Mutual Informazione (MMI)
- Maximization of Non-Gaussianity (MNG)
- Maximization of Likelihood or Entropy (ME)

- Mutual Information
- Marginal Entropy
- Joint Entropy
DifferentICAapproaches

\[ I(Y_1, Y_2) = \sum_{i=1}^{2} H(Y_i) - H(Y_1, Y_2) \]
Different ICA approaches

\[ I(Y_1, Y_2) = \sum_{i=1}^{2} H(Y_i) - H(Y_1, Y_2) \]

Mutual Information  
Marginal Entropy  
Joint Entropy
**Linear BSS**

The simplest problem formulation is in the linear and instantaneous environment: we observe $N$ random variables $x_1, \ldots, x_N$, which are modeled as linear combinations of $N$ random variables $s_1, \ldots, s_N$:

$$x = As$$  \hspace{1cm} (1)

where $a_{ij}$ are the entries of the mixing $A$ matrix.

The aim of BSS algorithms is to identify a de-mixing matrix $W$, in order to obtain that the components of the output vector

$$u = Wx$$  \hspace{1cm} (2)

are as statistically independent as possible.
Given a look to the linear (and instantaneous) mixing model more in depth, we can obtain the following model:

\[
\begin{align*}
    x_1 &= a_{11}s_1 + a_{12}s_2 + \ldots + a_{1N}s_N \\
    x_2 &= a_{21}s_1 + a_{22}s_2 + \ldots + a_{2N}s_N \\
    \vdots \\
    x_N &= a_{N1}s_1 + a_{N2}s_2 + \ldots + a_{NN}s_N
\end{align*}
\]

We want a similar model for the linear de-mixing model too.
The solution to this problem can be addressed by the well-known ICA algorithms, like FastICA, JADE, etc. Another well-performing algorithm is the InfoMax, proposed by Bell & Sejnowski in 1995.

- **InfoMax** addresses the problem of maximizing the mutual information $I(y; x)$, between the input vector $x$ and an invertible nonlinear transform of it, $y$ obtained as

$$y = h(u) = h(Wx)$$

where $W$ is the $N \times N$ de-mixing matrix and $h(u) = [h_1(u_1), \ldots, h_N(u_N)]^T$ is a set of $N$ nonlinear function.

Note that the network used by the Bell & Sejnowski algorithm is a *single layer neural network*. In this way the set of the nonlinear functions are the *activation function* of the neural network. For this reason the functions $h_i(u_i)$ are usually called *activation function* or *AF*. 

---

**M. Scarpiniti**  
**Neural Networks Lesson 8 - Blind Source Separation**
A cost function measuring the statistic independence of the network output $y$ is optimized, then the network free parameters (matrix weights, or nonlinear function parameters) are changed during the learning.
The **InfoMax** algorithm

We can analyze as the **InfoMax** algorithm works. Because the mapping between *input* and *output* is deterministic, maximizing $I(y, x)$ is the same that maximizing the joint entropy $H(y)$. In fact the following relation holds:

$$I(y, x) = H(y) - H(y|x)$$

where $H(y|x)$ is whatever entropy the output has which did not come from the input. In the case that we do not know what is noise and what is signal in the input, the mapping between $x$ and $y$ is deterministic and $H(y|x)$ has its lowest possible value (it diverges to $-\infty$). The above equation can be differentiated as follows, with respect to a parameter, $w$ involved in the mapping from $x$ to $y$:

$$\frac{\partial}{\partial w} I(y, x) = \frac{\partial}{\partial w} H(y)$$  \hspace{1cm} (3)$$

because $H(y|x)$ does not depend on $w$.

In this way **InfoMax** is equivalent to the entropy maximization or **ME**.
The **InfoMax** algorithm

In order to derive the learning algorithm let we pose $p_x(x)$ and $p_y(y)$ the **probability density functions** (pdf) of the network input and output respectively which have to satisfy the relation:

$$p_y(y) = \frac{p_x(x)}{|\text{det } J|}$$

where $|\bullet|$ denotes the absolute value and $J$ the **Jacobian matrix** of the transformation: $J = [\partial y_i/\partial x_j]_{ij}$.

Since the **joint entropy** of network output is defined as $H(y) = -E\{\ln p_y(y)\}$, we obtain:

$$H(y) = E\{\ln |\text{det } J|\} + H(x).$$

Now we can note that $\frac{\partial y_i}{\partial x_j} = \frac{\partial y_i}{\partial u_i} \frac{\partial u_i}{\partial x_j} = h'_i(u_i) \cdot w_{ij}$, so we obtain

$$\ln |\text{det } J| = \ln \text{det } W + \sum_{i=1}^{N} \ln |h'_i|.$$ 

Hence, the expression of the **joint entropy** $H(y)$ (ignoring the expected value operator $E\{\bullet\}$, replacing by instantaneous values) is:

$$H(y) = H(x) + \ln \text{det } W + \sum_{i=1}^{N} \ln |h'_i|.$$ 

\(4\)
The **InfoMax** algorithm

The maximization (or minimization) of a generic cost function $\mathcal{L} \{\Phi\}$ with respect a parameter $\Phi$ can be obtained by the application of the stochastic gradient method at $(l + 1)$-th iteration

$$\Phi (l + 1) = \Phi (l) + \eta_\Phi \frac{\partial \mathcal{L} \{\Phi (l)\}}{\partial \Phi} = \Phi (l) + \eta_\Phi \Delta \Phi (l)$$

where $\eta_\Phi$ is the learning rate.

Remembering that $H(x)$ is not affected by the parameters that we are learning, it is possible to write the learning rule for the matrix $W$ as follows:

$$\Delta W = \frac{\partial H(y)}{\partial W} = W^{-T} + \Psi x^T \quad (5)$$

where $W^{-T} = (W^{-1})^T$, $\Psi = [\Psi_1, \ldots, \Psi_N]^T$ and $\Psi_k = h''_k(u_k)/h'_k(u_k)$. 

M. Scarpiniti  
**Neural Networks Lesson 8 - Blind Source Separation**
The **InfoMax** algorithm: choice of nonlinearity

The nonlinear transformations $h_i(u_i)$ are necessary for *bounding* the entropy in a finite range. Indeed, when $h_i(u_i)$ is bounded $c \leq h_i(u_i) \leq d$, for any random variable $u_i$ the entropy of $y_i = h_i(u_i)$ has an upper bound:

$$H(y_i) \leq \ln(d - c)$$

Therefore, the joint entropy of the transformed output vector is upper bounded:

$$H(y) \leq \sum_{i=1}^{N} H(y_i) \leq N \ln(d - c)$$

In fact, the above inequality holds for any bounded transforms, so the global maximum of the entropy $H(y)$ exists. $H(y)$ may also have many *local maxima* determined by the functions $h_i$ used to transform $u$. If we chose $c = 0$ and $d = 1$, the the global maximum is reached at $H(y) = 0$. 
The **InfoMax** algorithm: relation to ICA

But what is the relationship between INFOMAX and ICA? For this scope we can introduce the mutual information of the linear outputs $u$ as the Kullback-Leibler divergence of the output distribution $I(u) = E \left\{ \ln \left( \frac{p_u(u)}{\prod_{i=1}^{N} p_{u_i}(u_i)} \right) \right\}$, where $p_u(u)$ is the joint pdf of the output vector $u$ and $p_{u_i}(u_i)$ are the marginal pdfs. Then:

$$H(y) = - \int p_y(y) \ln p_y(y) dy = -E \{ \ln p_y(y) \} =$$

$$= -E \left\{ \ln \frac{p_u(u)}{\prod_{i=1}^{N} |h_i'|} \right\} = -E \{ \ln p_u(u) \} + E \left\{ \ln \prod_{i=1}^{N} |h_i'| \right\} =$$

$$= -E \{ \ln p_u(u) \} + E \left\{ \ln \prod_{i=1}^{N} p_{u_i}(u_i) \right\} - E \left\{ \ln \prod_{i=1}^{N} p_{u_i}(u_i) \right\} + E \left\{ \ln \prod_{i=1}^{N} |h_i'| \right\} =$$

$$= -E \left\{ \ln \frac{p_u(u)}{\prod_{i=1}^{N} p_{u_i}(u_i)} \right\} + E \left\{ \sum_{i=1}^{N} \ln \frac{|h_i'|}{p_{u_i}(u_i)} \right\} =$$

$$= -I(u) + E \left\{ \sum_{i=1}^{N} \ln \frac{|h_i'|}{p_{u_i}(u_i)} \right\}$$

and we obtain:

$$H(y) = -I(u) + E \left\{ \sum_{i=1}^{N} \ln \frac{|h_i'|}{p_{u_i}(u_i)} \right\}$$
The **InfoMax** algorithm: **ME** and **MMI** approaches

\[ H(y) = -I(u) + E \left\{ \sum_{i=1}^{N} \ln \frac{|h'_i|}{p_{u_i}(u_i)} \right\} \]  

(6)

- Thus if \(|h'_i| = p_{u_i}(u_i) (\forall i)\) then maximizing the joint entropy \(H(y)\) is equivalent to minimizing the mutual information, that is the Kullback-Leibler divergence (which is a measure of the independence of the \(u_i\) signals) and so the **ICA problem is solved**. In this way \(h_i(u_i)\) should be the cumulative density function (cdf) of the \(i\)-th estimated source. The use of an adaptive AF can successfully fulfills the matching of \(h_i(u_i)\) to the cdf of the \(i\)-th source.

- Moreover the use of a cdf-like function for the \(h_i\) functions allows the joint entropy \(H(y)\) to have in \(H(y) = 0\) its global maximum (because \(c = 0\) and \(d = 1\)).

- From (6) the **InfoMax** algorithm can be performed by two equivalent approach: maximizing the joint entropy of the network output (**ME approach**) or minimizing the mutual information (**MMI approach**).
In order to obtain an *adaptive* or *flexible* activation function one can use the spline neuron.

- The spline neuron can be described by the following figure:

  ![Spline Neuron Diagram](image)

  - In the *second block* the terms $M_i$ are the columns of the $M$ matrix, which depends by the particular used basis.
The flexible **AF**: spline neuron

A matrix formulation of the output value $y_i$ can be described as:

$$y_i = h_i(u) = T \cdot M \cdot Q_i$$

where

$$T = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix},$$

$$Q_i = \begin{bmatrix} Q_i & Q_{i+1} & Q_{i+2} & Q_{i+3} \end{bmatrix}^T,$$

$$M = \frac{1}{2} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}.$$ 

Matrix $M$ determines the characteristic of the interpolant curve: **CR-Spline** or **B-Spline**

- In order to grant a **monotonic characteristic** of the overall function, the following constraint must be imposed: $Q_1 < Q_2 < \ldots < Q_N.$
Different spline basis

1. B-Spline:

\[ M = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \]

2. Bezier Spline:

\[ M = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \]

3. Catmull-Rom Spline:

\[ M = \frac{1}{2} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \]

4. Hermite Spline:

\[ M = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \]
A new **InfoMax** algorithm

A new algorithm can be derived using the **InfoMax** principle: maximizing the joint entropy of the network output. Now we have two sets of free parameters of the network: entries of the $W$ matrix and the spline control points. We have to derive two learning rules.

1. We can derive a similar **learning rule** for the *matrix weights*:

   $$ \Delta W = \frac{\partial H(y)}{\partial W} = W^{-T} + \Psi x^T $$

   But now it is

   $$ \Psi_i = \frac{h''_i(u_i)}{h'_i(u_i)} = \frac{1}{\Delta} \frac{\ddot{T}MQ_i}{\dot{T}MQ_i} $$

   where $\Delta = Q_{i+1} - Q_i$ is the distance between two adjacent control point, $\dot{T} = \begin{bmatrix} 3u^2 & 2u & 1 & 0 \end{bmatrix}$ and $\ddot{T} = \begin{bmatrix} 6u & 2 & 0 & 0 \end{bmatrix}$.

2. For the **learning rule** of the *spline control points*, we have:

   $$ \Delta Q_i^m \propto \frac{\partial H(y)}{\partial Q_i^m} = \frac{\partial \left[ \sum_{i=1}^{N} \ln h'_i(u_i) \right]}{\partial Q_i^m} = \frac{1}{h'_i(u_i)} \frac{\partial h'_i(u_i)}{\partial Q_i^m} = $$

   $$ = \frac{\Delta}{\dot{T}MQ_i} \cdot \frac{\partial (\ddot{T}MQ_i)}{\Delta \partial Q_i^m} = \frac{\ddot{T}M_m}{\dot{T}MQ_i}, \quad m = 0, \ldots, 3 $$

   where $M_m$ is the $m$-th column of the matrix $M$. 
The Natural Gradient

The stochastic gradient is a very simple approach, but if data lie in a Riemannian space $S = \vartheta \in \mathbb{C}$, convergence can be very slow or one can obtain local solutions.

- In order to avoid these problems, given a generic functional $L(\vartheta)$ defined in $S$, the concept of natural gradient was introduced:

$$\tilde{\nabla} L(\vartheta) = G^{-1}(\vartheta) \nabla L(\vartheta)$$

where $\nabla L(\vartheta)$ is the classical stochastic gradient, $G^{-1}(\vartheta)$ is the inverse of the Riemann curvature tensor and $\tilde{\nabla} L(\vartheta)$ is the natural gradient.

The Riemann curvature tensor expresses curvature of Riemannian manifolds. In this way the natural gradient can evaluate a gradient considering the curvature of the space in which data are defined. A classical example is the distance between two cities in the world: the “true” distance is an arc on the surface of the world and not a line between the two points.
Amari has demonstrated that the inverse of the *Riemann curvature tensor* is very simple for the set of non-singular matrices $W_{n \times n}$, denoted by $GL(n, \mathbb{R})$. We have:

$$\tilde{\nabla} L(W) = \nabla L(W) W^T W$$

The **InfoMax** algorithm can be reformulated in this new way:

$$\Delta W = \eta \left( W^{-T} + \Psi x^T \right) W^T W = \eta \left( I + \Psi x^T W^T \right) W = \eta \left( I + \Psi u^T \right) W$$

This algorithm is more efficient than the classic one, essentially for two reasons:

1. it avoids the local minima;
2. it avoids the evaluation of the inverse of the $W$ matrix.
The Natural Gradient

- In addition **four new Riemannian metrics** have been introduced by Arcangeli et al., 2004 and Squartini et al., 2005, for the improving of the convergence speed.

- In this way we can obtain 5 **new** expressions for the **natural gradient**: 

\[
\tilde{\nabla}_R \mathcal{L}(W) = \nabla \mathcal{L}(W) W^T W \tag{7}
\]

\[
\tilde{\nabla}_L \mathcal{L}(W) = WW^T \nabla \mathcal{L}(W) \tag{8}
\]

\[
\tilde{\nabla}_{LR} \mathcal{L}(W) = (WW^T) \nabla \mathcal{L}(W)(W^T W) \tag{9}
\]

\[
\tilde{\nabla}_{RR} \mathcal{L}(W) = \nabla \mathcal{L}(W)(W^T W^T WW) \tag{10}
\]

\[
\tilde{\nabla}_{LL} \mathcal{L}(W) = (WWW^T W^T) \nabla \mathcal{L}(W) \tag{11}
\]

defined as **right natural gradient** (7) (that is the standard natural gradient), **left natural gradient** (8), **right/left natural gradient** (9), **right/right natural gradient** (10) and **left/left natural gradient** (11).
The Natural Gradient

- Using the previous expressions, we can obtain the new formulations for the learning rules of the de-mixing matrix $W$: from the equation

$$\Delta W = W^{-T} + \Psi x^T$$

we obtain the five rules:

$$\Delta W = (I + \Psi u^T)W \quad (12)$$

$$\Delta W = W(I + W^T \Psi x^T) \quad (13)$$

$$\Delta W = WW^T(I + \Psi u^T)W \quad (14)$$

$$\Delta W = (I + \Psi u^T)W^TWW \quad (15)$$

$$\Delta W = WWW^T(I + W^T \Psi x^T) \quad (16)$$
The Natural Gradient

Not all these new gradient definitions work efficiently for BSS problem: some of these equations do not satisfy the **equivariance** property (Cardoso & Laheld, 1996).

**Definition**

An estimator $\hat{A}$ of $A$ ($\hat{A} = A(X)$) is **equivariant** if for every invertible matrix $M$ is:

$$A(MX) = MA(X)$$

- This implies that, defined $C = WA$, the natural gradient algorithm (12) depends only from $C$:

$$\Delta W \cdot A = \Delta C = (I + \Psi u^T)WA = (I + \Psi u^T)C$$
The Natural Gradient

- The **equivariance** property is not satisfied by all algorithms (13)-(16).
- If we consider that the mixing matrix $A$ is **unitary** ($AA^T = A^T A = I$), then:
  1. $\Delta W \cdot A = W(I + \psi x^T)A$
  2. $\Delta W \cdot A = WW^T(I + \psi u^T)WA = WAA^T W^T(I + \psi u^T)C = CC^T(I + \psi u^T)C$
  3. $\Delta W \cdot A = (I + \psi u^T)W^T WWA = (I + \psi u^T)W^T A^T AWC = (I + \psi u^T)C^T CC$
  4. $\Delta W \cdot A = WWW^T(I + \psi x^T)A$

- It is possible to note that the **left** (13) and **left/left** (16) algorithms do not satisfy the **equivariance property**: we suppose that they do not work efficiently in BSS solution.
The **InfoMax** algorithm with the natural gradient, converges if (Mathis et al., 2002)

\[
E \{ \Psi'(u) \} E \{ u^2 \} + E \{ \Psi(u) u^T \} > 0 \tag{17}
\]

Mathis has demonstrated that there not exist an “universal” nonlinear function, which satisfy the previous equation for every input signal.

- Using **spline functions** the previous constraint is always satisfied.
- Function \( \Psi_i \triangleq \frac{p'_i(x)}{p_i(x)} \equiv \frac{h''_i(u)}{h'_i(u)} \) is defined as **score function**. Because we use an adaptive function, at convergence the activation function, must assume the profile of the cdf of the input signal, in order to minimize the last terms (6). In this way the functions \( \Psi_i \) tends to the **score functions** and can be demonstrated that condition (17) always holds.
Under-determined BSS

If the number $M$ of sensors is less than the number $N$ of sources, the problem (known as under-determined or over-complete BSS) is more complicated. A useful approach in this case could be the use of a geometric algorithm.

- This approach is particular performing if the source are sparse. Sparsity is a measurement of zero elements in the source signal. Greater the sparsity is, more super-gaussian the signal is.
- If signals are enough sparse, the scatter-plot of the mixtures reveals some “characteristic” directions. From these directions the mixing matrix $\mathbf{A}$ can be estimated as follows from 2 sensors and $N$ sources:

\[
\hat{\mathbf{A}} = \begin{bmatrix}
\cos \alpha_1 & \cos \alpha_2 & \cdots & \cos \alpha_N \\
\sin \alpha_1 & \sin \alpha_2 & \cdots & \sin \alpha_N
\end{bmatrix}
\]
Under-determined BSS

In order to improve the sparsity of the mixtures it is possible to perform some time-frequency transformation on the time signal. A good choice is

1. the Short-Time Fourier Transform (STFT): the function to be transformed is multiplied by a window function which is nonzero for only a short period of time. The Fourier transform (a one-dimensional function) of the resulting signal is taken as the window is slid along the time axis, resulting in a two-dimensional representation of the signal, written as:

\[
\text{STFT}\{x(t)\} \equiv X(\omega, \tau) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-j\omega t} dt
\]

2. some Wavelet Transformation. A wavelet is a wave-like oscillation with an amplitude that starts out at zero, increases, and then decreases back to zero. A wavelet transform is used to divide a time signal into wavelets. It possesses the ability to construct a time-frequency representation of a signal that offers very good time and frequency localization. The wavelet transform at scale \(a > 0\) and translation value \(b \in \mathbb{R}\) is

\[
X_\omega(a, b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t)\Psi^*(\frac{t-b}{a}) dt
\]
Under-determined BSS

The sparsity enhancement is evident from the following pdf plot of a time-domain and a STFT-domain signal.

- After this preprocessing step, a clustering algorithm, like a K-means, can be used in order to detect the angles $\alpha_i$ for the estimation of the mixing matrix (Blind Mixing Model Recovery (BMMR) step);
- Another problem is the recovering the original sources, solving the corresponding under-determined (non-square) system (Blind Source Recovery (BSR) step). In literature there exist several ways to solve such a system.
Blind Source Extraction (BSE)

BSS can be computationally very demanding if the number of source signals is large. Fortunately, Blind Source Extraction (BSE) overcomes this difficulty by attempting to recover only a small subset of desired independent components from a large number of sensor signals: most of the existing BSE criteria and associated algorithms only recover one independent component at a time.

- A single processing unit (a neuron) is used in the first step to extract one independent source signal with the specified stochastic properties. In the next step a deflection technique is used in order to eliminate the already extracted signals from the mixtures.
Blind Source Extraction (BSE)

A learning rule for the extraction of the first independent component, can be derived, after whitening, minimizing the following cost function:

$$J_1(w_1) = -\frac{1}{4} |k(y_1)| = -\frac{\beta}{4} k(y_1)$$

where $k(y_1)$ is the kurtosis of the first output $y_1$ and $\beta = \pm 1$ determines the sign of the kurtosis of the extracted signal. Applying the standard gradient descent, we have:

$$\Delta w_1 = -\mu_1 \frac{\partial J_1}{\partial w_1} = \mu_1 \beta \frac{m_4(y_1)}{m_2(y_1)} \left[ \frac{m_2(y_1)}{m_4(y_1)} E \{ y_1^3 x_1 \} - E \{ y_1 x_1 \} \right]$$

where $\mu_1 > 0$ is the learning rate, $m_2(y_1) = E \{ |y|^2 \}$ and $m_4(y_1) = E \{ |y|^4 \}$. The term $m_4(y_1)/m_2^3(y_1)$ is always positive and can be absorbed in the learning rate. Let us pose

$$\varphi(y_1) = \beta \frac{m_4(y_1)}{m_2^3(y_1)} \left[ \frac{m_2(y_1)}{m_4(y_1)} y_1^3 - y_1 \right]$$

we obtain the following learning rule:

$$\Delta w_1 = \mu_1 \varphi(y_1) x_1 \quad (18)$$
Blind Source Extraction (BSE)

After successful extraction of the first source signal $y_1 \approx s_1$, we can apply the deflation procedure which removes previously extracted signals from the mixtures. This procedure may be recursively applied to extract all source signals:

$$x_{j+1} = x_j - \tilde{w}_j y_j \quad j = 1, 2, \ldots$$

where the filter $\tilde{w}_j$ can be optimally estimated by minimization of the cost function

$$\tilde{J}_j (\tilde{w}_j) = E \left\{ x_{j+1}^T x_{j+1} \right\} = E \left\{ x_j^T x_j \right\} - 2 \tilde{w}_j^T E \left\{ x_j y_j \right\} + \tilde{w}_j^T \tilde{w}_j E \left\{ y_j^2 \right\}$$

with respect to $\tilde{w}_j$ leads to the simple updating equation:

$$\tilde{w}_j = \frac{E \left\{ x_j y_j \right\}}{E \left\{ y_j^2 \right\}} = \frac{E \left\{ x_j x_j^T \right\}}{E \left\{ y_j^2 \right\}} w_j$$
Other **BSS** techniques
Nonlinear BSS

Unfortunately linear instantaneous mixing models are too unrealistic and unsatisfactory in many applications. A more realistic mixing system inevitably introduces a nonlinear distortion in the signals. In this way the possibility of taking into account these distortions can give better results in signal separation. The problem is that in the nonlinear case the uniqueness of the solution is not guaranteed. The solution becomes easier in a particular case, called Post Nonlinear (PNL) mixture, well-known in literature in the case of the real domain. In this context the solution is unique too.

The mixing procedure is realized in two steps: first there is a linear mixing, then each mixtures is obtained as a nonlinear distorted version of the linear mixtures. The PNL mixture model is quite probable in many practical situations.
Nonlinear BSS

If the mixing-separating system is nonlinear and no other assumption is given for the mixing operator, a generic de-mixing model does not assure the existence and uniqueness of the solution. In order to better illustrate this aspect, see the following example.

Example

Consider two independent random variables $s_1$ with uniform distribution in $[0, 2\pi)$ and $s_2$ with Rayleigh distribution, so that its pdf is $p_{s_2}(s_2) = \frac{s_2}{\sigma_2^2} e^{-s_2^2/2}$ with variance $\sigma_2^2 = 1$.

Given the two nonlinear transformations $y_1 = s_2 \cos s_1$ and $y_2 = s_2 \sin s_1$, the random variables $y_1$ and $y_2$ are still independent but are Gaussian distributed, so they cannot be separated as a consequence of the Darmois-Skitovich's theorem. In fact the Jacobian $J$ of this transformation is:

$$\det (J) = \det \left( \begin{array}{cc} \frac{\partial y_1}{\partial s_1} & \frac{\partial y_1}{\partial s_2} \\ \frac{\partial y_2}{\partial s_1} & \frac{\partial y_2}{\partial s_2} \end{array} \right) = \det \left( \begin{array}{cc} -s_2 \sin s_1 & \cos s_1 \\ s_2 \cos s_1 & \sin s_1 \end{array} \right) = -s_2$$

so the joint pdf of $y = [y_1, y_2]$ can be expressed as

$$p_{y_1, y_2}(y_1, y_2) = \frac{p_{s_1, s_2}(s_1, s_2)}{|\det(J)|} = \frac{1}{2\pi} \exp \left( -\frac{y_1^2 + y_2^2}{2} \right) = \left( \frac{1}{\sqrt{2\pi}} \exp \left( -\frac{y_1^2}{2} \right) \right) \left( \frac{1}{\sqrt{2\pi}} \exp \left( -\frac{y_2^2}{2} \right) \right) \equiv p_{y_1}(y_1) \cdot p_{y_2}(y_2)$$
Nonlinear BSS: the *mirror model*

In order to solve the BSS problem in nonlinear environment, using PNL mixtures, the following de-mixing model is adopted:

This model is known as *mirror model*, because the de-mixing model is the mirror image of the mixing one. In formulas

\[
\begin{align*}
v &= As \\
x &= F(v), \quad x_i = f_i(v_i) \\
r &= G(x), \quad r_i = g_i(x_i) \\
u &= Wr
\end{align*}
\]
The whole system is shown in the following learning scheme:

*The learning architecture must adapt the matrix weights* $w_{ij}$, *the nonlinear activation function* $h_i(u_i)$ *and the compensating functions* $g_i(x_i)$ *of the distorting functions* $f_i(v_i)$.
Nonlinear BSS: a flexible solution

In order to obtain an adaptive function representation for activation function and compensating the distorting functions, we can adopt splines. In this way we have three sets of free parameters:

1. the entries $w_{ij}$ of the de-mixing matrix;
2. the control points $Q_i^h$ of the AFs;
3. the control points $Q_i^G$ of the compensating the distorting functions $g_i(x_i)$.

Using the InfoMax algorithm, we have to maximize the following cost function:

$$\frac{\partial}{\partial \Phi} \mathcal{L} \{y, \Phi\} = \frac{\partial}{\partial \Phi} \left[ \ln |\det (W)| + \sum_{k=1}^{N} \ln \dot{y}_k + \sum_{k=1}^{N} \ln \dot{r}_k \right]$$

where the free parameters, are $\Phi = \{w_{ij}, Q_i^h, Q_i^G\}$. 
Nonlinear BSS: a flexible solution

Maximization of the previous equation by the stochastic gradient method yields to three learning rules:

1. The learning rule for the network’s weights is:
   \[
   \Delta W = W^{-T} + \Psi r^T
   \]
   where \( \Psi_i = h_i''(u_i)/h_i'(u_i) \) as usual.

2. The learning rule for the spline activation functions is:
   \[
   \Delta Q_{i+m}^h = \frac{\dot{\mathbf{M}}_m^h}{\dot{\mathbf{MQ}}_{i+m}^h}
   \]
   where \( \mathbf{M}_m \) is the \( m \)-th column of the \( \mathbf{M} \) matrix.

3. The learning rule for the spline compensating functions is:
   \[
   \Delta Q_{i+m}^G = \frac{\dot{\mathbf{M}}_m^G}{\dot{\mathbf{MQ}}_{i+m}^G} + \Psi \mathbf{W}_k^H \mathbf{T}\mathbf{M}_m
   \]
   where \( \mathbf{W}_k \) is a vector composed by the \( k \)-th column of the matrix \( \mathbf{W} \).
Indeterminacy of nonlinear BSS

In PNL BSS we have not only the usual scaling and permutation ambiguity.

**Definition**

Let \( A \in G_l(n) \) be an invertible matrix, then \( A = (a_{ij})_{i,j=1...n} \) is said to be **absolutely degenerate** if there are two columns \( l \neq m \) such that \( a^2_{il} = \lambda a^2_{im} \) for a \( \lambda \neq 0 \), i.e. the normalized columns differ only by the signs of the entries.

- A not so obvious indeterminacy occurs if \( A \) is absolutely degenerate: only the matrix \( A \) but not the nonlinearities can be recovered.
- An additional indeterminacy come into play because of translation: the distorting functions \( f_i \) can only be recovered up to a constant off-set.
Other **BSS** environment

The nature of some problem involves a natural solution in a complex environment, due to the need of frequency domain signal processing which is quite common in telecommunication and biomedical applications.

The problem formulation and solution is similar to the real-value one, except that all quantities involved in model are complex-value. For example let us consider a vector \( \mathbf{s} = [s_1, \ldots, s_N]^T \) of \( N \) complex sources (\( \mathbf{s} \in \mathbb{C}^N \)). The \( k \)-th source can be expressed as \( s_k = s_{Rk} + js_{Ik} \), where \( s_{Rk} \) and \( s_{Ik} \) are the real and imaginary parts of the \( k \)-th complex-valued source signal and \( j = \sqrt{-1} \) is the imaginary unit. The goal of complex BSS is to recover the complex signal \( \mathbf{s} \) from observations of the complex mixture \( \mathbf{x} = [x_1, \ldots, x_N]^T \), where the \( k \)-th mixture can be expressed as \( x_k = x_{Rk} + jx_{Ik} \), \( x_{Rk} \) and \( x_{Ik} \) are its real and imaginary part. In this way the model in real-valued case are still valid but the mixing matrix \( \mathbf{A} \) and the de-mixing matrix \( \mathbf{W} \) are complex matrices \( (a_{ij} \in \mathbb{C} \text{ and } w_{ij} \in \mathbb{C}) \):

\[
\mathbf{x} = \mathbf{As}
\]
\[
\mathbf{u} = \mathbf{Wx}
\]

In this way all the real-valued algorithm, like the **InfoMax**, can be generalized to the complex-valued environment.
Example 1: *coctail-party* problem

A first example is the well-known *coctail-party problem* in reverberant environment:

\[
x(t) = A(t) \ast s(t) \quad \Longleftrightarrow \quad X(f, t) = A(f) S(f, t)
\]

A **convolutive BSS problem** can be addressed as a set on \( N \) instantaneous problem, on for each frequency bin.
Example 1: *coctail-party* problem

An example of spectrogram of two sources and two mixtures.
Example 2: **fMRI** problem

A second example is the Functional Magnetic Resonance Imaging or **fMRI**.

Two orthogonal scanners acquire a slice of temporal voxels in order to measure the **Blood-oxygen-level dependent (BOLD)**.
Example 2: fMRI problem

Example of fMRI images, we have 5 super-gaussian, 1 gaussian and 2 sub-gaussian signals:
Example 3: communication problem

A third example is the \textit{band-pass transmission}

Usually it is used the \textit{complex envelope}

\[ x(t) = x_R(t) + jx_S(t) \]
Example 3: communication problem

Example of an 8-PSK, a 16-QAM, a 4-QAM and a uniform noise signal.
The complex activation function (AF)

One of the main issues in designing complex neural networks is the presence of complex nonlinear activation functions $h(z)$ (where $z = z_R + jz_I$) involved in the learning processing.

- The main challenge is the dichotomy between boundedness and analyticity in the complex domain, as stated by the Liouville’s theorem: complex functions, bounded on the whole complex plane, are either constant or not analytic. Thus this kind of complex nonlinear functions are not suitable as activation functions of neural networks.

- In particular Georgiou & Koutsougeras defined the following set of properties:
  1. $h(z)$ is nonlinear in $z_R$ e $z_I$;
  2. $h(z)$ is bounded: $|h(z)| \leq c < \infty$;
  3. partial derivatives $h^R_{z_R}$, $h^R_{z_I}$, $h^l_{z_R}$ e $h^l_{z_I}$ exist and are bounded;
  4. $h(z)$ is not entire;
  5. $h^R_{z_R} h^l_{z_I} \neq h^R_{z_I} h^l_{z_R}$.

where $h^R(z_R, z_I)$ and $h^l(z_R, z_I)$ are known as the real part function and imaginary part function of the complex function $h(z)$ respectively.
In this context Kim and Adali proposed the use of the so-called elementary transcendental functions (ETF). They classified the ETFs into two categories of unbounded functions, depending on which kind of singularities they possess:

1. **Circular functions**: \( \tan(z) \), \( \sin(z) \) and \( \cot(z) \);
2. **Inverse circular functions**: \( \tan^{-1}(z) \), \( \sin^{-1}(z) \) and \( \cos^{-1}(z) \);
3. **Hyperbolic functions**: \( \tanh(z) \), \( \sinh(z) \) and \( \coth(z) \);
4. **Inverse hyperbolic functions**: \( \tanh^{-1}(z) \), \( \sinh^{-1}(z) \) and \( \cosh^{-1}(z) \).

As expected the trigonometric and the corresponding hyperbolic functions behave very similarly.
The complex activation function (AF): the *splitting solution*

According to the properties listed above, in order to overcome the dichotomy between **boundedness** and **analyticity**, complex nonlinear *splitting functions* have been introduced. In this approach real and imaginary parts are processed separately by real-valued nonlinear functions. The splitting function

\[
h(z) = h(z_R, z_I) = h^R(z_R) + jh^I(z_I)
\]

avoids the problem of unboundedness of complex nonlinearities, as stated above, but it cannot be analytic.
The complex activation function (AF): the generalized splitting solution

The splitting model of a nonlinear complex valued function is not realistic because usually the real and imaginary part are correlated. According to this issue it is useful to perform a more realistic model of the nonlinear functions. In this way, Vitagliano et al. (2003) proposed a complex neural network based on a couple of bi-dimensional functions called generalized splitting function:

\[ h(z) = h(z_R, z_I) = h^R(z_R, z_I) + jh^I(z_R, z_I) \]

In this way \( h(z) \) is bounded but it is not analytic. The Cauchy-Riemann conditions \( (h^R_{z_R} = h^I_{z_I}, h^I_{z_R} = -h^R_{z_I}) \) are not satisfied by the complex function itself, but can be imposed by an algorithm constraint during the learning process.

\[
\begin{align*}
    z^{\text{in}} &= z^{\text{in}}_R + jz^{\text{in}}_I \\
    h \left[ z^{\text{in}} \right] &= \hat{h} \left[ z^{\text{in}} \right] \\
    z^{\text{out}} &= z^{\text{out}}_R + jz^{\text{out}}_I \\
    h^R(z^{\text{in}}_R, z^{\text{in}}_I) &= h^R(z^{\text{out}}_R, z^{\text{out}}_I) \\
    h^I(z^{\text{in}}_R, z^{\text{in}}_I) &= h^I(z^{\text{out}}_R, z^{\text{out}}_I) \\
    z^{\text{out}} &= z^{\text{out}}_R + jz^{\text{out}}_I
\end{align*}
\]
The new idea is to use spline-based functions for implementing the splitting activation function. Thus, remembering the spline matrix formulation we can firstly evaluate the indexes span $i_R$ and $i_I$, and the local parameters $\nu_R$ and $\nu_I$, then we have

$$y_k = h_k^R(u_{Rk}) + jh_k^I(u_{Ik}) = T_R \cdot M \cdot Q_{iR}^R + jT_I \cdot M \cdot Q_{iI}^I$$
The generalized spline neuron

A matrix formulation of the output $y_i$ can be written as:

$$y_i = h_i(u_R, u_I) = T_{\nu,2} \cdot M \cdot (T_{\nu,1} \cdot M \cdot Q_i)^T$$

dove

$$T_{\nu,k} = \begin{bmatrix} u_k^3 & u_k^2 & u_k \end{bmatrix}, \quad k = 1, 2$$

$$Q_i^k = \begin{bmatrix} Q_{i1}^{(i_1,i_2)} & Q_{i1}^{(i_1,i_2+1)} & Q_{i1}^{(i_1,i_2+2)} & Q_{i1}^{(i_1,i_2+3)} \\ Q_{i2}^{(i_1+1,i_2)} & Q_{i2}^{(i_1+1,i_2+1)} & Q_{i2}^{(i_1+1,i_2+2)} & Q_{i2}^{(i_1+1,i_2+3)} \\ Q_{i3}^{(i_1+2,i_2)} & Q_{i3}^{(i_1+2,i_2+1)} & Q_{i3}^{(i_1+2,i_2+2)} & Q_{i3}^{(i_1+2,i_2+3)} \\ Q_{i4}^{(i_1+3,i_2)} & Q_{i4}^{(i_1+3,i_2+1)} & Q_{i4}^{(i_1+3,i_2+2)} & Q_{i4}^{(i_1+3,i_2+3)} \end{bmatrix}$$

$$M = \frac{1}{2} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}$$

The matrix $M$ shows the characteristic of the interpolant curve: CR-Spline or B-Spline

- In order to satisfy a monotonic characteristic of the overall function, the following constraint must be imposed: $Q_1^k < Q_2^k < \ldots < Q_N^k$. 

M. Scarpiniti  NEURAL NETWORKS Lesson 8 - Blind Source Separation
Unfortunately the real and imaginary parts of a complex signal are usually correlated, not split in separate channels. In this way we need a better model of the complex AF. We can render each of the two bi-dimensional real functions $h^R(u_r, u_I)$ and $h^I(u_r, u_I)$ with bi-dimensional splines: one plays the role of the real part and one the imaginary part of the complex activation function.
The de-mixing algorithm

In order to derive the de-mixing algorithm, let us pose

\[
\tilde{x}[n] = \begin{bmatrix} x_R[n] \\ x_I[n] \end{bmatrix} = \begin{bmatrix} x_{R1}(n) & \cdots & x_{RN}(n) \\ x_{I1}(n) & \cdots & x_{IN}(n) \end{bmatrix}
\]

\[
\tilde{u}[n] = \begin{bmatrix} u_R[n] \\ u_I[n] \end{bmatrix} = \begin{bmatrix} u_{R1}(n) & \cdots & u_{RN}(n) \\ u_{I1}(n) & \cdots & u_{IN}(n) \end{bmatrix}
\]

\[
\tilde{y}[n] = \begin{bmatrix} y_R[n] \\ y_I[n] \end{bmatrix} = \begin{bmatrix} h(u_R[n]) \\ h(u_I[n]) \end{bmatrix}
\]

- We utilize the **InfoMax** algorithm.
- As cost function we use the joint entropy (ME approach).
The de-mixing algorithm

- Let us pose $\Phi$ the free parameters, then the cost function is:

$$
\mathcal{L} \{\tilde{y}[n], \Phi\} = H(\tilde{y}[n]) = H(\tilde{x}) - E \left\{ \log \left( \det \left( \tilde{J} \right) \right) \right\} =
$$

$$
= H(x) + \log |\det W| + 2 \cdot \sum_{n=1}^{N} E[\log \left| h'_n(u_n) \right|]
$$

- The free parameters for our architecture are:

$$
\Phi = \left\{ w_{ij}, Q^h \right\}
$$
The de-mixing algorithm

- Evaluating the derivatives with respect the de-mixing matrix, we obtain:
  \[
  \frac{\partial \log |\det W|}{\partial W} = W^{-H} \\
  \frac{\partial \sum_n \log |h'_n(u_n)|}{\partial W} = \Psi(u)x^H
  \]

- \(\Psi(u)\) is a function depending on spline parameters.

- We obtain the following learning rule that is a generalization of the Bell & Sejnowski algorithm:
  \[
  \Delta W = \frac{\partial \mathcal{L} \{\tilde{y}, \Phi\}}{\partial W} = \eta \left( W^{-H} + \Psi(u)x^H \right)
  \]
The de-mixing algorithm

- The function $\Psi(u) = \Psi(u_R, u_I) + j\Psi(u_R, u_I)$ is formed as:

$$
\begin{align*}
\psi_{iR} &= 2 \frac{\frac{\partial y_{iR}}{\partial u_{iR}} \frac{\partial^2 y_{iR}}{\partial u_{iR}^2} + \frac{\partial y_{iR}}{\partial u_{iR}} \frac{\partial y_{iR}}{\partial u_{iI}} \frac{\partial y_{iR}}{\partial u_{iI}}}{(\frac{\partial y_{iR}}{\partial u_{iR}})^2 + (\frac{\partial y_{iR}}{\partial u_{iI}})^2}}{\Delta} \\
\psi_{iI} &= 2 \frac{\frac{\partial y_{iI}}{\partial u_{iI}} \frac{\partial^2 y_{iI}}{\partial u_{iI}^2} + \frac{\partial y_{iI}}{\partial u_{iI}} \frac{\partial y_{iI}}{\partial u_{iR}} \frac{\partial y_{iI}}{\partial u_{iR}}}{(\frac{\partial y_{iI}}{\partial u_{iI}})^2 + (\frac{\partial y_{iI}}{\partial u_{iR}})^2}
\end{align*}
$$

- and in matrix notation:

$$
\psi_{iR} = \frac{2}{\Delta} \left( \frac{\left(T_{li} \cdot M \cdot (\hat{T}_{ri} \cdot M \cdot Q_{ri})^T\right) \left(T_{li} \cdot M \cdot (\hat{T}_{ri} \cdot M \cdot Q_{ri})^T\right)}{\left(T_{li} \cdot M \cdot (\hat{T}_{ri} \cdot M \cdot Q_{ri})^T\right)^2 + (\hat{T}_{li} \cdot M \cdot (T_{ri} \cdot M \cdot Q_{ri})^T)^2} + \frac{(\hat{T}_{li} \cdot M \cdot (T_{ri} \cdot M \cdot Q_{ri})^T) \left(T_{li} \cdot M \cdot (\hat{T}_{ri} \cdot M \cdot Q_{ri})^T\right)}{\left(T_{li} \cdot M \cdot (\hat{T}_{ri} \cdot M \cdot Q_{ri})^T\right)^2 + (\hat{T}_{li} \cdot M \cdot (T_{ri} \cdot M \cdot Q_{ri})^T)^2} \right)
$$

and similar for $\psi_{iI}$.
The de-mixing algorithm

- Evaluating the derivatives with respect the spline control points, we obtain:

\[
2 \frac{\partial \sum_n \log |h'_n(u_n)|}{\partial Q_{i_R+i+m_l}} = \begin{cases} 
0 & i \neq j \\
2 \frac{\partial y_{jR}}{\partial u_{jR}} \frac{\partial}{\partial Q_{i_R+i+m_l}} + \frac{\partial y_{jR}}{\partial u_{jI}} \frac{\partial}{\partial Q_{i_R+i+m_l}} & i = j \\
\left( \frac{\partial y_{jR}}{\partial u_{jR}} \right)^2 + \left( \frac{\partial y_{jR}}{\partial u_{jI}} \right)^2 & i = j 
\end{cases}
\]

- and in matrix notation:

\[
\Delta Q_{i_R+i+m_l} = 2 \eta_Q \left( \frac{T_{lj} \cdot M \cdot (T_{Rj} \cdot M \cdot Q_{Rj})^T}{T_{lj} \cdot M \cdot (T_{Rj} \cdot M \cdot Q_{Rj})^T} + \frac{T_{lj} \cdot M}{T_{lj} \cdot M} \cdot (T_{Rj} \cdot M \cdot Q_{Rj})^T \right)^2 + \frac{T_{lj} \cdot M \cdot (T_{Rj} \cdot M \cdot Q_{Rj})^T}{T_{lj} \cdot M \cdot (T_{Rj} \cdot M \cdot Q_{Rj})^T} \left( \frac{\dot{T}_{lj} \cdot M \cdot (T_{Rj} \cdot M \cdot Q_{Rj})^T}{\dot{T}_{lj} \cdot M \cdot (T_{Rj} \cdot M \cdot Q_{Rj})^T} \right)^2
\]

where \(M_k\) is a zero matrix, except the \(k\)-th column, that is equal to the \(k\)-the column of the \(M\) matrix.
A similar architecture can be found for the PNL mixtures case. In this new architecture a third learning rule can be derived for the adaptation of the spline control point of the compensating nonlinear function:

$$
\Delta Q_{G_{R, i+m}}^G = \frac{\dot{T}_{RM_m}}{T_{RMQ_{G_{R, i+m}}}} + \text{Re} \left\{ \psi W_k^H \right\} T_{RM_m},
$$

$$
\Delta Q_{I_{1,i+m}}^G = \frac{\dot{T}_{IM_m}}{T_{IMQ_{I_{1,i+m}}}} + \text{Im} \left\{ \psi W_k^H \right\} T_{IM_m}.
$$
The scaling ambiguity of BSS algorithm is reflected in a phase ambiguity (rotation) in the complex domain. We can analyze as recover this information.

Assume that $x = [x_1, \ldots, x_N]^T$ and $s = [s_1, \ldots, s_N]^T$, we obtain:

$$
\begin{bmatrix}
    x_1R \\
    x_1I \\
    x_2R \\
    x_2I \\
    \vdots \\
    x_NR \\
    x_NI
\end{bmatrix}
= \begin{bmatrix}
    a_{11}R & -a_{11}I & a_{12}R & -a_{21}I & \cdots & a_{1N}R & -a_{1N}I \\
    a_{11}I & a_{11}R & a_{12}I & a_{12}R & \cdots & a_{1N}I & a_{1N}R \\
    a_{21}R & -a_{21}I & a_{22}R & -a_{22}I & \cdots & a_{2N}R & -a_{2N}I \\
    a_{21}I & a_{21}R & a_{22}I & a_{22}R & \cdots & a_{2N}I & a_{2N}R \\
    \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
    a_{N1}R & -a_{N1}I & a_{N2}R & -a_{N2}I & \cdots & a_{NN}R & -a_{NN}I \\
    a_{N1}I & a_{N1}R & a_{N2}I & a_{N2}R & \cdots & a_{NN}I & a_{NN}R
\end{bmatrix}
\begin{bmatrix}
    s_1R \\
    s_1I \\
    s_2R \\
    s_2I \\
    \vdots \\
    s_NR \\
    s_NI
\end{bmatrix}
$$

That is:

$$
\begin{bmatrix}
    x_1 \\
    x_2 \\
    \vdots \\
    x_N
\end{bmatrix}
= \begin{bmatrix}
    A_{11} & A_{12} & \cdots & A_{1N} \\
    A_{21} & A_{22} & \cdots & A_{2N} \\
    \vdots & \vdots & \ddots & \vdots \\
    A_{N1} & A_{N2} & \cdots & A_{NN}
\end{bmatrix}
\begin{bmatrix}
    s_1 \\
    s_2 \\
    \vdots \\
    s_N
\end{bmatrix}
$$

where $A_{ij} = \begin{bmatrix} a_{ijR} & -a_{ijI} \\ a_{ijI} & a_{ijR} \end{bmatrix}$ and $\det(A_{ij}) = a_{ijR}^2 + a_{ijI}^2 = |a_{ij}|^2$. 

\[ (19) \]
We desire a similar de-mixing model

\[
\begin{bmatrix}
    u_{1R} \\
    u_{1I} \\
    u_{2R} \\
    u_{2I} \\
    \vdots \\
    u_{NR} \\
    u_{NI}
\end{bmatrix}
= \begin{bmatrix}
    w_{11R} & -w_{11I} & w_{12R} & -w_{21I} & \cdots & w_{1NR} & -w_{1NI} \\
    w_{11I} & w_{11R} & w_{12I} & w_{12R} & \cdots & w_{1NI} & w_{1NR} \\
    w_{21R} & -w_{21I} & w_{22R} & -w_{22I} & \cdots & w_{2NR} & -w_{2NI} \\
    w_{21I} & w_{21R} & w_{22I} & w_{22R} & \cdots & w_{2NI} & w_{2NR} \\
    \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
    w_{N1R} & -w_{N1I} & w_{N2R} & -w_{N2I} & \cdots & w_{NNR} & -w_{NNI} \\
    w_{N1I} & w_{N1R} & w_{N2I} & w_{N2R} & \cdots & w_{NNI} & w_{NNR}
\end{bmatrix}
\begin{bmatrix}
    x_{1R} \\
    x_{1I} \\
    x_{2R} \\
    x_{2I} \\
    \vdots \\
    x_{NR} \\
    x_{NI}
\end{bmatrix}
\]

that is

\[
\begin{bmatrix}
    u_1 \\
    u_2 \\
    \vdots \\
    u_N
\end{bmatrix}
= \begin{bmatrix}
    W_{11} & W_{12} & \cdots & W_{1N} \\
    W_{21} & W_{22} & \cdots & W_{2N} \\
    \vdots & \vdots & \ddots & \vdots \\
    W_{N1} & W_{N2} & \cdots & W_{NN}
\end{bmatrix}
\begin{bmatrix}
    x_1 \\
    x_2 \\
    \vdots \\
    x_N
\end{bmatrix}
\]

(20)

where \( W_{ij} = \begin{bmatrix}
    w_{ijR} & -w_{ijI} \\
    w_{ijI} & w_{ijR}
\end{bmatrix} \) and \( \det(W_{ij}) = w_{ijR}^2 + w_{ijI}^2 = |w_{ij}|^2 \).
Phase recovery

So we can have a **phase recovery** with the following constraint:

\[
W_{ij} = \frac{1}{2} \left\{ W_{ij} + \text{det}(W_{ij}) \cdot W_{ij}^{-T} \right\}
\]  

(21)

It can be demonstrated that the previous constraint is *equivalent* to utilize a splitting activation function:

\[
h(z) = h^R(z_R) + jh^I(z_I)
\]
A. Hyvarinen, J. Karhunen, E. Oja.
Independent Component Analysis.

A. Cichocki and S.-I. Amari.
Adaptive Blind Signal and Image Processing.

A.J. Bell and T.J. Sejnowski.
An information-maximisation approach to blind separation and blind deconvolution.


S-I. Amari.
Natural gradient works efficiently in learning.

Generalized Splitting Functions for Blind Separation of Complex Signals.

Flexible Nonlinear Blind Signal Separation in the Complex Domain.