Sur la séparation de sources dans des mélanges nonlinéaire et convolutifs

Massoud BABAIE-ZADEH

Résumé

Dans cette thèse, la séparation aveugle de sources dans des mélanges Convolutif Post Non-Linéaire (CPNL) est étudiée. Pour séparer ce type de mélanges, nous avons d'abord développé de nouvelles méthodes pour séparer les mélanges convolutifs et les mélanges Post Non-Linéaires (PNL). Ces méthodes sont toutes basées sur la minimisation de l'information mutuelle des sorties. Pour minimiser l'information mutuelle, nous calculons d'abord sa "différentielle", c'est-à-dire, sa variation en fonction d'une petite variation de son argument. Cette différentielle est alors utilisée pour concevoir des approches de type gradient pour minimiser l'information mutuelle des sorties. Ces approches peuvent être appliquées pour séparation aveugle des mélanges linéaires instantanés, convolutif, PNL et CPNL.

1 Introduction

La séparation aveugle de source est un problème important en traitement des signaux et a été largement étudié dans la dernière décennie. Dans le cas instantané linéaire, le mélange est de la forme :

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{1}$$

où $\mathbf{s}$ est le vecteur des sources (supposées statistiquement indépendantes), $\mathbf{x}$ est le vecteur des observations, et $\mathbf{A}$ est la matrice mélangeante (supposée régulière). Pour séparer les sources, on estime une matrice de séparation, $\mathbf{B}$, telle que :

$$\mathbf{y} = \mathbf{B}\mathbf{x} \tag{2}$$

Comon a montré [1] que l'indépendance des composantes de $\mathbf{y}$ est suffisante pour obtenir la séparation (avec des indéterminations sur les ordres et le gain des sources) pour vu qu'au plus une source soit Gaussienne. Autrement dit, les mélanges linéaires instantanés sont *séparables*.

Dans le cas linéaire convolutif, les matrices de mélanges et de séparation sont des matrices de filtres, fréquemment des filtres à réponse impulsionnelle finie (RIF) [2, 3, 4]. Pour ce type de mélanges aussi, Yellin et Weinstein ont montré que l'indépendance des sorties est suffisante pour obtenir la séparation [5] : les mélanges linéaires convolutifs sont aussi *séparables*. Mais, le critère d'indépendance de y_1 et y_2 ne se restreint pas à l'indépendance de $y_1(n)$ et $y_2(n)$. Il doit inclure l'indépendance des paires $y_1(n)$ et $y_2(n-m)$ pour toutes les paires (m, n). De plus, les indéterminations sont plus importantes : dans ce cas-là, les signaux sont obtenus à un filtre près. Néanmoins, après la séparation,

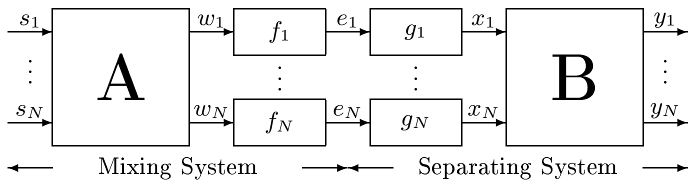

FIG. 1 – Les systèmes mélangeurs et séparants pour des mélanges PNL.

il est possible de trouver comme source estimée, les sources vue au niveau de chaque capteur en absence de toutes les autres [3].

Par contre, les mélanges nonlinéaires *ne sont en général pas séparables*. En effet, on peut construire des mélanges nonlinéaires qui rendent les sorties indépendantes tout en mélangeant des sources [6]. En conséquence, pour séparer les mélanges nonlinéaires il faut ajouter des contraintes structurelles. Une régularisation par lissage des fonctions est insuffisante : on a mentionné (dans la version complète de la thèse) un exemple d'un système lisse qui préserve l'indépendance tout en mélangeant des sources.

Taleb et Jutten [6] ont introduit les mélanges Post Non-Linéaires (PNL), comme un cas particulier, mais assez réaliste, de mélanges non linéaires séparables. Comme il est montré dans Fig. 1, il s'agit d'un mélange linéaire suivi par des nonlinéarités voie par voie. Ceci correspond à un canal linéaire et à des capteurs qui introduisent des distortions : par exemple, en conséquence des nonlinéarités des amplificateurs, comme saturation. Pour séparer ce type des mélanges, on utilise une structure adaptée au mélange : d'abord compenser les nonlinéarités des captures, puis séparer le mélange linéaire obtenu.

Comme extension des mélanges PNL, on peut envisager les mélanges Convolutifs Post Non-Linéaires (CPNL). Ce type de mélanges ressemble à celui de la Fig. 1, mais les matrices A et B sont remplacées par des matrices de filtres. Ce type de mélange correspond à un mélange linéaire convolutif suivi par des capteurs nonlinéaires. La séparabilité de mélanges CPNL peut être justifiée par la séparabilité de mélanges PNL. Dans ce résumé, nous allons introduire des approches pour séparer les mélanges CPNL.

Pour cela, on prend l'information mutuelle des sorties comme le critère d'indépendance et on va calculer sa différentielle. Pour ce calcul, on va avoir besoin de définir les fonctions scores conjointes et marginales d'un vecteur aléatoire. Ensuite, on va profiter cette différentielle pour construire des approches pour minimiser l'information mutuelle. Finalement, on va utiliser ces approches pour séparer les mélanges linéaires instantanés, convolutifs, PNL et CPNL.

Ce document est un résumé de la version complète de la thèse qui compte plus que 140 pages (en anglais). Aussi, certains résultats moins importants ne figurent pas dans ce document. Par exemple, les théorèmes qui concernent la séparabilité des mélanges nonlinéaires, notamment la méthode géométrique pour séparer des mélanges PNLs [7] sont supprimés. D'autre part, aucun résultat expérimental avec les algorithmes n'est présenté. Les résultats détaillés de ces travaux sont accessibles dans la thèse [8] et les articles [9, 7, 4, 10, 11, 12, 13].

2 Le critère d'indépendance

2.1 L'information mutuelle

L'information mutuelle des variables aléatoires $x_1, x_2, \ldots, x_N$ est définie comme la divergence Kullback-Leibler entre $p_{\mathbf{x}}(\mathbf{x})$ et $\prod_i p_{x_i}(x_i)$:

$$I(\mathbf{x}) = \int_{\mathbf{x}} p_{\mathbf{x}}(\mathbf{x}) \ln \frac{p_{\mathbf{x}}(\mathbf{x})}{\prod_i p_{x_i}(x_i)} d\mathbf{y} = \sum_i H(x_i) - H(\mathbf{x}) \tag{3}$$

où H est l'entropie de Shannon. Par les propriétés de la divergence de Kullback-Leibler, on sait que $I(\mathbf{x})$ est toujours positif, et s'annule si et seulement si $p_{\mathbf{x}}(\mathbf{x}) = \prod_i p_{x_i}(x_i)$, c'est-à-dire les variables x_i sont indépendantes.

Par conséquent, on peut utiliser l'information mutuelle comme un critère pour mesurer l'indépendance des sorties dans un système séparateur de sources. Alors, l'algorithme de séparation peut être basé sur la minimisation de l'information mutuelle des sorties.

Pour cette minimisation, on peut utiliser une approche de type gradient. Pour cela il sera très utile d'avoir une expression pour la variation de l'information mutuelle en fonction d'une petite variation dans son argument (la "différentielle" de l'information mutuelle), valable quelle que soit la forme, paramétrique ou non, du système de séparation à estimer.

Avant présenter le théorème concernant la différentielle de l'information mutuelle, il est nécessaire de définir les fonctions scores multi-variables pour un vecteur aléatoire.

2.2 Les fonctions score multi-variables

2.2.1 Définitions

Rappelons d'abord la définition de la fonction score d'une variable aléatoire :

Définition 1 *La fonction score de la variable aléatoire x est définie par :*

$$\psi_x(x) \triangleq -\frac{d}{dx} \ln p_x(x) = -\frac{p'_x(x)}{p_x(x)} \tag{4}$$

où p_x est la densité de probabilité de x.

Maintenant, pour un vecteur aléatoire $\mathbf{x} = (x_1, \ldots, x_N)^T$, on définit deux types différentes des fonctions scores.

Définition 2 *La Fonction Score Marginale (MSF[1]) de* $\mathbf{x}$ *est le vecteur des fonctions scores de ses composantes, c'est-à-dire :*

$$\boldsymbol{\psi}_{\mathbf{x}}(\mathbf{x}) \triangleq (\psi_1(x_1), \ldots, \psi_N(x_N))^T \tag{5}$$

où :

$$\psi_i(x_i) \triangleq -\frac{d}{dx_i} \ln p_{x_i}(x_i) = -\frac{p'_{x_i}(x_i)}{p_{x_i}(x_i)} \tag{6}$$

[1]Marginal Score Function : On gardera les abréviations tirées de l'anglais, pour que les notations restent homogènes avec les articles et le mémoire de la thèse.

Définition 3 *La Fonction Score conJointe (JSF*[2]*) de* $\mathbf{x}$ *est le gradient de "*$-\ln p_{\mathbf{x}}(\mathbf{x})$*", c'est-à-dire :*

$$\boldsymbol{\varphi}_{\mathbf{x}}(\mathbf{x}) \triangleq (\varphi_1(\mathbf{x}), \ldots, \varphi_N(\mathbf{x}))^T \tag{7}$$

où :

$$\varphi_i(\mathbf{x}) \triangleq -\frac{\partial}{\partial x_i} \ln p_{\mathbf{x}}(\mathbf{x}) = -\frac{\frac{\partial}{\partial x_i} p_{\mathbf{x}}(\mathbf{x})}{p_{\mathbf{x}}(\mathbf{x})} \tag{8}$$

On définit aussi :

Définition 4 *La Différence des Fonctions Scores (SFD*[3]*) de* $\mathbf{x}$ *est la différence entre sa MSF et sa JSF :*

$$\boldsymbol{\beta}_{\mathbf{x}}(\mathbf{x}) \triangleq \boldsymbol{\psi}_{\mathbf{x}}(\mathbf{x}) - \boldsymbol{\varphi}_{\mathbf{x}}(\mathbf{x}) \tag{9}$$

2.2.2 Propriétés

On présente ici quelques propriétés importantes des fonctions scores multivariables. Les démonstrations de ces propriétés sont supprimées ici et peuvent être trouver dans la version complète de la thèse.

Propriété 1 *Les composantes du vecteur* $\mathbf{x} = (x_1, \ldots, x_N)^T$ *sont indépendantes si et seulement si sa SFD s'annule, c'est-à-dire :*

$$\boldsymbol{\varphi}_{\mathbf{x}}(\mathbf{x}) = \boldsymbol{\psi}_{\mathbf{x}}(\mathbf{x}) \tag{10}$$

Propriété 2 *Pour le vecteur aléatoire* $\mathbf{x} = (x_1, \ldots, x_N)^T$ *on a :*

$$\beta_i(\mathbf{x}) = \frac{\partial}{\partial x_i} \ln p(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N | x_i) \tag{11}$$

où $\beta_i(\mathbf{x})$ *désigne la i-ème composante de la SFD de* $\mathbf{x}$*.*

Notons la liaison entre ces deux propriétés. Par exemple, pour le cas bidimensionnel, la propriété 2 s'écrit :

$$\beta_1(x_1, x_2) = \frac{\partial}{\partial x_1} \ln p(x_2 \mid x_1) \tag{12}$$

$$\beta_2(x_1, x_2) = \frac{\partial}{\partial x_2} \ln p(x_1 \mid x_2) \tag{13}$$

Donc, $\beta_1(x_1, x_2) = 0$ si $p(x_2|x_1)$ ne dépend pas de x_1, c'est-à-dire quand x_1 et x_2 sont indépendantes.

Propriété 3 *Soit* $\mathbf{x}$ *un vecteur aléatoire de densité* $p_{\mathbf{x}}$ *et de JSF* $\boldsymbol{\varphi}_{\mathbf{x}}$*. Soit* $f(\mathbf{x})$ *une fonction multi-variable dont les dérivées partielles existent et sont continues. Alors, si on suppose que :*

$$\lim_{x_i \to \pm\infty} \int_{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N} f(\mathbf{x}) p_{\mathbf{x}}(\mathbf{x}) \, dx_1 \cdots dx_{i-1} \, dx_{i+1} \cdots dx_N = 0 \tag{14}$$

on a :

$$E\left\{f(\mathbf{x})\varphi_i(\mathbf{x})\right\} = E\left\{\frac{\partial f}{\partial x_i}(\mathbf{x})\right\} \tag{15}$$

[2]Joint Score Function
[3]Marginal Score Function

Notez bien que la condition (14) n'est pas trop restrictive. Par exemple, elle est satisfaite pour tous les vecteurs aléatoires bornés. Cette propriété sera utile pour une estimation par les moindres de carrés de la JSF.

Propriété 4 *Pour le vecteur aléatoire* $\mathbf{x} = (x_1, \dots, x_N)^T$ *on a :*

$$\psi_i(x) = E\left\{\varphi_i(\mathbf{x}) \mid x_i = x\right\} \tag{16}$$

où φ_i *et* ψ_i *désignent la* i*-ème composante de la JSF et la MSF de* $\mathbf{x}$*, respectivement.*

Autrement dit, la SFD est une mesure de d'écart de la JSF à sa moyenne.

2.3 La différentielle de l'information mutuelle

Le théorème qui suit est le théorème principal qui nous permet de construire des algorithmes de type gradient pour minimiser une information mutuelle.

Théorème 1 *Soit* $\mathbf{x}$ *un vecteur aléatoire borné et* $\boldsymbol{\Delta}$ *un vecteur aléatoire "petit" en même dimension. Alors :*

$$I(\mathbf{x} + \boldsymbol{\Delta}) - I(\mathbf{x}) = E\left\{\boldsymbol{\Delta}^T \boldsymbol{\beta}_{\mathbf{x}}(\mathbf{x})\right\} + o(\boldsymbol{\Delta}) \tag{17}$$

où $\boldsymbol{\beta}_{\mathbf{x}}$ *est la SFD de* $\mathbf{x}$*, et* $o(\boldsymbol{\Delta})$ *désigne les termes d'ordre supérieurs en* $\boldsymbol{\Delta}$*.*

D'après ce théorème, on remarque que la SFD est le «gradient stochastique» de l'information mutuelle.

A partir de ce théorème et de la propriété 1 on peut déduire le théorème suivant :

Théorème 2 *Soit* $\mathbf{x}_0$ *un vecteur aléatoire, dont la densité possede des dérivés continues. Si pour tout vecteur aléatoire petit* $\boldsymbol{\Delta}$*,* $I(\mathbf{x}_0) \leq I(\mathbf{x}_0 + \boldsymbol{\Delta})$*, alors* $I(\mathbf{x}_0) = 0$*.*

Ce théorème très important montre que, «*l'information mutuelle n'a pas de minimum local*».

2.4 Estimation de fonctions scores multi-variables

Les densités des sources étant inconnues, la conception d'algorithmes basée sur la minimisation de l'information mutuelle par le résultat (17), requirent l'estimation des JSF et MSF.

2.4.1 Estimation de JSF

Estimation par noyaux Une fonction $k(\mathbf{x}) = k(x_1, \dots, x_N)$ est appelée *noyau* si : (a) $\forall \mathbf{x} \in \mathbb{R}^N, k(\mathbf{x}) \geq 0$, (b) $\int_{\mathbb{R}^N} k(\mathbf{x})\, d\mathbf{x} = 1$ et (c) $\int_{\mathbb{R}^N} \mathbf{x}\, k(\mathbf{x})\, d\mathbf{x} = \mathbf{0}$.

Soit $\mathbf{x}$ un vecteur aléatoire, dont T échantillons $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T\}$ sont observés. Alors, l'estimateur à noyaux de sa densité s'écrit :

$$\hat{p}_{\mathbf{x}}(\mathbf{x}) \triangleq \frac{1}{T} \sum_{t=1}^{T} k(\mathbf{x} - \mathbf{x}_t) \tag{18}$$

d'où l'on en déduit l'estimation de la i-ème composante de sa JSF :

$$\hat{\varphi}_i(\mathbf{x}) \triangleq -\frac{\frac{\partial}{\partial x_i} \hat{p}_{\mathbf{x}}(\mathbf{x})}{\hat{p}_{\mathbf{x}}(\mathbf{x})} = -\frac{\sum_{t=1}^{T} \frac{\partial k}{\partial x_i}(\mathbf{x} - \mathbf{x}_t)}{\sum_{t=1}^{T} k(\mathbf{x} - \mathbf{x}_t)} \tag{19}$$

Estimation de maximum de vraisemblance Cette méthode est basée sur la propriété 3. Supposons que l'on veuille estimer $\varphi_i(\mathbf{x})$ comme une combinaison linéaire des fonctions multi-variables $\{k_1(\mathbf{x}), \ldots, k_L(\mathbf{x})\}$, c'est-à-dire :

$$\hat{\varphi}_i(\mathbf{x}) = \sum_{j=1}^{L} w_j k_j(\mathbf{x}) = \mathbf{k}^T(\mathbf{x})\,\mathbf{w} \tag{20}$$

par la propriété 3, on peut montrer que le $\mathbf{w}$ qui minimise l'erreur :

$$\mathcal{E} = E\left\{\left(\varphi_i(\mathbf{x}) - \hat{\varphi}_i(\mathbf{x})\right)^2\right\} \tag{21}$$

est donné par :

$$E\left\{\mathbf{k}(\mathbf{x})\mathbf{k}^T(\mathbf{x})\right\}\mathbf{w} = E\left\{\frac{\partial \mathbf{k}}{\partial x_i}(\mathbf{x})\right\} \tag{22}$$

2.4.2 Estimation de SFD

Estimations indépendantes de la JSF et la MSF Pour estimater la SFD, une méthode est estimer les JSF et MSF séparément, et puis calculer leur différence, c'est-à-dire :

$$\hat{\boldsymbol{\beta}}_{\mathbf{x}}(\mathbf{x}) = \hat{\boldsymbol{\psi}}_{\mathbf{x}}(\mathbf{x}) - \hat{\boldsymbol{\varphi}}_{\mathbf{x}}(\mathbf{x}) \tag{23}$$

Exemple.

(Estimation polynomiale de SFD) Cet estimateur peut être appliqué dans des mélanges linéaires (instantanés ou convolutifs) de 2 sources. Dans cette méthode, on estime $\varphi_i(x_1, x_2)$ par :

$$\hat{\varphi}_i(x_1, x_2) = \sum_{j=1}^{7} w_{ij}\, k_j(x_1, x_2) \tag{24}$$

où :

$$\begin{array}{llll} k_1(x_1,x_2) = 1, & k_2(x_1,x_2) = x_1, & k_3(x_1,x_2) = x_1^2, & k_4(x_1,x_2) = x_1^3 \\ & k_5(x_1,x_2) = x_2, & k_6(x_1,x_2) = x_2^2, & k_7(x_1,x_2) = x_2^3 \end{array} \tag{25}$$

et les valeurs optimales de w_{ij}'s sont obtenues de (22).

De manière similaire, $\psi_i(x_i)$ est estimée par :

$$\hat{\psi}_i(x_i) = w_1^{(i)} + w_2^{(i)} x_i + w_3^{(i)} x_i^2 + w_4^{(i)} x_i^3 \tag{26}$$

Finalement, on calcule SFD par (23).

En réalité, cette méthode est peu satisfaisante. En effet, un algorithme de type gradient pour minimiser l'information mutuelle s'arrête quand la SFD est zéro. Donc, si on estime la MSF et la JSF séparément, les erreurs de l'estimations sont indépendantes, et alors l'algorithme peut s'arrêter sur des mauvaises solutions.

Estimation de SFD par le lissage de JSF Nous proposons estimer la MSF a partir de l'estimation de la JSF en utilisant la propriété 4.

Autrement dit, on estime d'abord la JSF par n'importe quelle méthode, puis on estime la MSF par (16) (en utilisant une méthode de régression ; par exemple des splines de lissage), et enfin, on calcule leur différence.

Estimation par histogramme Dans cette méthode (pour le cas bidimensionnel), on utilise d'abord un histogramme pour estimer $p_{\mathbf{x}}(x_1, x_2)$. Soit $p(n_1, n_2)$ cet histogramme. Alors, l'histogramme qui estime la densité de x_1 est :

$$p_1(n_1) = \sum_{n_2} p(n_1, n_2) \tag{27}$$

et donc l'estimation de $p(x_2|x_1)$ sera :

$$p(n_2|n_1) = \frac{p(n_1, n_2)}{p_1(n_1)} = \frac{N(n_1, n_2)}{N(n_1)} \tag{28}$$

Finalement, en utilisant (12), on obtient une estimation de $\beta_1(x_1, x_2)$:

$$\beta_1(n_1, n_2) = \frac{p(n_2|n_1) - p(n_2|n_1 - 1)}{p(n_2|n_1)} \tag{29}$$

$\beta_2(n_1, n_2)$ est estimée par une manière similaire.

Malgré sa simplicité, cet estimateur donne des bons résultat dans la séparation de mélanges linéaires instantanés (voir la version complète de la thèse), même avec un découpage en un petit nombre de bandes (vers 10).

La méthode de Pham Récemment, D. T. Pham [14] a proposé une méthode, basée sur des splines et l'entropie, pour l'estimation de la fonction score conditionnelle définie par :

$$\boldsymbol{\psi}_{x_N|x_{N-1}\cdots x_1}(x_N|x_{N-1}, \dots, x_1) \triangleq -\nabla \ln p_{x_N|x_{N-1}\cdots x_1}(x_N|x_{N-1}, \dots, x_1) \tag{30}$$

où $p_{x_N|x_{N-1}\cdots x_1}(x_N|x_{N-1}, \dots, x_1)$ est la densité de probabilité conditionnelle de x_N en sachant $x_1, \dots, x_{N-1}$. Par la propriété 2 on peut également utiliser cet estimateur pour estimer la SFD.

2.5 La minimisation de l'information mutuelle

On a vu qu'un algorithme de séparation de source peut être basé sur la minimisation de l'information mutuelle des sorties, c'est-à-dire que les paramètres du système séparateur doivent être calculés afin que l'information mutuelle des sorties soit minimale. On présente ici deux approches différentes pour faire cette minimisation.

2.5.1 Approche gradient

Dans cette approche, la dérivée de l'information mutuelle des sorties par rapport au chaque paramètre est calculée, puis un algorithme de type gradient est appliqué sur chaque paramètre. Un paramètre peut être un scalaire, un vecteur ou une fonction (pour les mélanges PNL, par exemple). Autrement dit, si on indique le système de séparation par $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$, l'algorithme de séparation est :

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \mu \frac{\partial}{\partial \boldsymbol{\theta}} I(\mathbf{y}) \tag{31}$$

2.5.2 Approche projection (Minimisation-Projection)

Soit $\mathbf{y}_n$ un vecteur aléatoire de SFD $\boldsymbol{\beta}_{\mathbf{y}_n}(\mathbf{y})$. S'il existe une région sur laquelle la fonction $\boldsymbol{\beta}_{\mathbf{y}_n}(\mathbf{y})$ est nonnulle et continue, alors $E\left\{\|\boldsymbol{\beta}_{\mathbf{y}_n}(\mathbf{y})\|^2\right\}$ ne peut pas être égale à zéro. Définissons le nouveau vecteur aléatoire $\mathbf{y}_{n+1}$ comme suit :

$$\mathbf{y}_{n+1} = \mathbf{y}_n - \mu \boldsymbol{\beta}_{\mathbf{y}_n}(\mathbf{y}_n) \tag{32}$$

où μ est un petit scalaire positif. Par le théorème 1 on a :

$$I(\mathbf{y}_{n+1}) - I(\mathbf{y}_n) = -\mu E\left\{\|\boldsymbol{\beta}_{\mathbf{y}_n}(\mathbf{y}_n)\|^2\right\} < 0 \Rightarrow I(\mathbf{y}_{n+1}) < I(\mathbf{y}_n) \tag{33}$$

Donc, l'algorithme $\mathbf{y} \leftarrow \mathbf{y} - \mu\boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y})$ fait diminuer $I(\mathbf{y})$ à chaque itération et par le théorème 2 il finira par produire les composantes indépendantes. Cependant, cet algorithme n'assure pas que la transformation finale $\mathbf{x} \mapsto \mathbf{y}$ appartient à la famille $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$.

Pour résoudre ce problème, à chaque itération on remplace la transformation $\mathbf{x} \mapsto \mathbf{y}$ par sa projection sur la famille $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$. Autrement dit, chaque itération de l'approche projection de séparation de source consiste en étapes suivantes :

- Minimisation :
  1. $\mathbf{y} \leftarrow \mathbf{y} - \mu\boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y})$,
- Projection :
  2. $\boldsymbol{\theta}_0 = \operatorname{argmin}_{\boldsymbol{\theta}} E\left\{\|\mathbf{y} - f(\mathbf{x}; \boldsymbol{\theta})\|^2\right\}$,
  3. $\mathbf{y} \leftarrow f(\mathbf{x}; \boldsymbol{\theta}_0)$.

3 Les mélanges linéaires instantanés

3.1 L'approche gradient

Pour ce type de mélanges, le système séparateur est $\mathbf{y} = \mathbf{B}\mathbf{x}$, et il faut calculer le gradient de $I(\mathbf{y})$ par rapport à la matrice $\mathbf{B}$. En utilisant le théorème 1 on peut montrer que :

$$\frac{\partial}{\partial \mathbf{B}} I(\mathbf{y}) = E\left\{\boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y})\,\mathbf{x}^T\right\} \tag{34}$$

Par conséquent, le gradient relatif [15, 16] sera :

$$\nabla_{\mathbf{B}} I \triangleq \frac{\partial I}{\partial \mathbf{B}}\,\mathbf{B}^T = E\left\{\boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y})\,\mathbf{y}^T\right\} \tag{35}$$

L'algorithme final de séparation (en considérant les indéterminations) est présenté dans la figure 2.

3.2 L'approche projection

Pour utiliser cette approche, dans la phase de projection, on a besoin de calculer la matrice $\mathbf{B}$ qui minimise $E\left\{\|\mathbf{y} - \mathbf{B}\mathbf{x}\|^2\right\}$. C'est facile a montré que cette matrice est calculée par :

$$\mathbf{B}_{\text{opt}} = E\left\{\mathbf{y}\mathbf{x}^T\right\}\left(E\left\{\mathbf{x}\mathbf{x}^T\right\}\right)^{-1} \tag{36}$$

L'algorithme final de séparation (en considérant les indéterminations) est présenté dans la figure 3.

- Initialization: $\mathbf{B} = \mathbf{I}$.
- Loop:
  1. $\mathbf{y} = \mathbf{B}\mathbf{x}$.
  2. Estimate $\boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y})$ (*e.g.* by histogram method).
  3. $\nabla_{\mathbf{B}} I = E\left\{\beta_{\mathbf{y}}(\mathbf{y})\,\mathbf{y}^T\right\}$
  4. $\mathbf{B} \leftarrow (\mathbf{I} - \mu \nabla_{\mathbf{B}} I)\,\mathbf{B}$.
  5. Normalization: Divide the i-th row of the matrix $\mathbf{B}$ by σ_i, where σ_i^2 is the energy of y_i.
- Repeat until convergence.

Figure 2: L'algorithme du gradient pour séparer des mélanges linéaires instantanés.

- Initialization: $\mathbf{y} = \mathbf{x}$.
- Loop:
  1. $\mathbf{y} \leftarrow \mathbf{y} - \mu \boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y})$.
  2. Remove the DC of each output, and normalize its energy to 1.
  3. $\mathbf{B} = E\left\{\mathbf{y}\mathbf{x}^T\right\}\left(E\left\{\mathbf{x}\mathbf{x}^T\right\}\right)^{-1}$
  4. $\mathbf{y} = \mathbf{B}\mathbf{x}$.
- Repeat until convergence.

Figure 3: L'algorithme de projection pour séparer des mélanges linéaires instantanés.

4 Les mélanges convolutifs

On n'envisage ce type de mélanges que pour le cas bidimensionnel. Le système de séparation est supposé être RIF de degré p :

$$\mathbf{y}(n) = [\mathbf{B}(z)]\mathbf{x}(n) = \sum_{k=0}^{p} \mathbf{B}_k \mathbf{x}(n-k) \tag{37}$$

et pour obtenir la séparation, $y_1(n)$ et $y_2(n-m)$ doivent être indépendantes pour toutes le valeurs de m et n. Le critère d'indépendance est alors :

$$J \triangleq \sum_{m=-M}^{+M} I\big(y_1(n), y_2(n-m)\big) \tag{38}$$

En théorie, M doit être infinie, mais on n'a pas besoin de plus de termes que de retards introduit par les paramètres. En pratique, on prend $M = 2p$.

Malgré tout, J est très lourd à calculer. Donc, en notant que nos algorithmes sont itératifs, dans chaque itération nous prenons un m par hasard entre $-M$ et M, et utilisons le critère $I\big(y_1(n), y_2(n-m)\big)$. Le critère (38) est alors minimisé en moyenne lors que l'on réalise de nombreuses itérations.

Dans ce résumé, pour un vecteur $\mathbf{x}(n) = (x_1(n), x_2(n))^T$, on utilise la notation $\mathbf{x}^{(m)}(n)$ pour le vecteur $(x_1(n), x_2(n-m))^T$. Donc, le critère d'indépendance dans chaque itération est $I\big(\mathbf{y}^{(m)}(n)\big)$.

4.1 L'approche gradient

Pour cette approche, on a besoin du gradient de $I\big(\mathbf{y}^{(m)}(n)\big)$ par rapport aux matrices $\mathbf{B}_k$, pour $k = 0, \ldots, p$. Dans la version complète de la thèse, on montre que :

$$\frac{\partial}{\partial \mathbf{B}_k} I\big(y_1(n), y_2(n-m)\big) = E\left\{\boldsymbol{\beta}_m(n)\, \mathbf{x}(n-k)^T\right\} \tag{39}$$

où :

$$\boldsymbol{\beta}_m(n) \triangleq \boldsymbol{\beta}^{(-m)}_{\mathbf{y}^{(m)}}(n) = \begin{bmatrix} \boldsymbol{\beta}_{\mathbf{y}^{(m)},1}(n) \\ \boldsymbol{\beta}_{\mathbf{y}^{(m)},2}(n+m) \end{bmatrix} \tag{40}$$

Autrement dit :

$$\begin{pmatrix} y_1(n) \\ y_2(n) \end{pmatrix} \xrightarrow{\text{Décalage}} \begin{pmatrix} y_1(n) \\ y_2(n-m) \end{pmatrix} \xrightarrow{\text{SFD}}$$
$$\begin{pmatrix} \beta_1^\star(n) \\ \beta_2^\star(n) \end{pmatrix} \xrightarrow{\text{Décalage}} \begin{pmatrix} \beta_1^\star(n) \\ \beta_2^\star(n+m) \end{pmatrix} \triangleq \boldsymbol{\beta}_m(n)$$

L'algorithme final (en considérant les indéterminations) est présenté dans la figure 4.

4.2 L'approche projection

Pour utiliser cette approche, il faut résoudre le problème de la projection d'une transformation générale sur la famille des transformations linéaires convolutives. C'est-à-dire, il faut trouver les matrices $\mathbf{B}_k$ ($k = 0, \ldots, p$) qui minimisent l'erreur $E\left\{\|\mathbf{y}(n) - [\mathbf{B}(z)]\,\mathbf{x}(n)\|^2\right\}$.

- Initialization:
  1. $\mathbf{B}_0 = \mathbf{I}$.
  2. For $k = 1 \dots p$, $\mathbf{B}_k = \mathbf{0}$.
  3. $\mathbf{y}(n) = \mathbf{x}(n)$.
- Loop:
  1. Choose a random m from the set $\{-M, \dots, +M\}$.
  2. Estimate $\boldsymbol{\beta}^\star(n)$, the SFD of $\big(y_1(n), y_2(n-m)\big)^T$.
  3. Let $\boldsymbol{\beta}_m(n) = (\beta_1^\star(n), \beta_2^\star(n+m))^T$.
  4. For $k = 0 \dots p$:
  $$\mathbf{B}_k \leftarrow \mathbf{B}_k - \mu E\left\{\boldsymbol{\beta}_m(n)\, \mathbf{x}(n-k)^T\right\}$$
  5. Calculate $\mathbf{y}(n) = [\mathbf{B}(z)]\mathbf{x}(n)$.
  6. Normalization:
     - Let $y_i = y_i/\sigma_i$, where σ_i^2 is the energy of y_i.
     - Divide the i-th raw of $\mathbf{B}(z)$ by σ_i.
- Repeat until convergence.

Figure 4: L'algorithme du gradient pour séparer des mélanges linéaires convolutifs.

- Initialization: $\mathbf{y}(n) = \mathbf{x}(n)$.
- Loop:
  1. Choose a random m from the set $\{-M, \ldots, +M\}$.
  2. Estimate $\boldsymbol{\beta}_{\mathbf{y}^{(m)}}$, the SFD of $\big(y_1(n), y_2(n-m)\big)^T$.
  3. Update the outputs by:
  $$\mathbf{y}^{(m)} \leftarrow \mathbf{y}^{(m)} - \mu \boldsymbol{\beta}_{\mathbf{y}^{(m)}}(\mathbf{y}^{(m)})$$
  4. Remove the DC of each output, and normalize its energy.
  5. Compute the matrices $\mathbf{B}_k$, $k = 0, \ldots, p$, from (44).
  6. Let $\mathbf{y}(n) = [\mathbf{B}(z)]\, \mathbf{x}(n)$.
- Repeat until convergence.

Figure 5: L'algorithme de projection pour séparer des mélanges linéaires convolutifs.

Ce problème est résolu dans la version complète de la thèse. En résumé, si on remplace l'espérance mathématique (E) par la moyenne empirique ($\hat{E}$), et on définit :

$$\hat{\mathbf{R}}_{\mathbf{xx}}(j,k) \triangleq \hat{E}\left\{\mathbf{x}(n-j)\mathbf{x}^T(n-k)\right\} \tag{41}$$
$$\hat{\mathbf{R}}_{\mathbf{yx}}(j,k) \triangleq \hat{E}\left\{\mathbf{y}(n-j)\mathbf{x}^T(n-k)\right\} \tag{42}$$

alors :

$$\sum_{j=0}^{p} \mathbf{B}_j \hat{\mathbf{R}}_{\mathbf{xx}}(j,k) = \hat{\mathbf{R}}_{\mathbf{yx}}(0,k) \quad , \quad k = 1, \ldots, p \tag{43}$$

et en forme matricielle :

$$\begin{aligned} &[\mathbf{B}_0 \quad \mathbf{B}_1 \quad \cdots \quad \mathbf{B}_p] \begin{bmatrix} \hat{\mathbf{R}}_{\mathbf{xx}}(0,0) & \hat{\mathbf{R}}_{\mathbf{xx}}(0,1) & \cdots & \hat{\mathbf{R}}_{\mathbf{xx}}(0,p) \\ \hat{\mathbf{R}}_{\mathbf{xx}}(1,0) & \hat{\mathbf{R}}_{\mathbf{xx}}(1,1) & \cdots & \hat{\mathbf{R}}_{\mathbf{xx}}(1,p) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\mathbf{R}}_{\mathbf{xx}}(p,0) & \hat{\mathbf{R}}_{\mathbf{xx}}(p,1) & \cdots & \hat{\mathbf{R}}_{\mathbf{xx}}(p,p) \end{bmatrix} \\ &\qquad = \left[\ \hat{\mathbf{R}}_{\mathbf{yx}}(0,0) \quad \hat{\mathbf{R}}_{\mathbf{yx}}(0,1) \quad \cdots \quad \hat{\mathbf{R}}_{\mathbf{yx}}(0,p)\ \right] \end{aligned} \tag{44}$$

qui determine les matrices $\mathbf{B}_k, k = 0, \ldots, p$.

L'algorithme final (en considérant les indéterminations) est présenté dans la figure 5.

5 Les mélanges Post Non-Linéaires (PNL)

Les systèmes mélangeurs-séparateurs pour ce type des mélanges ont été montrés dans la figure 1. Pour séparer le mélange, les fonctions g_i et la matrice $\mathbf{B}$

doivent être calculées pour produire des sorties indépendantes.

5.1 L'approche gradient

Dans cette approche, nous devons calculer les gradients de $I(\mathbf{y})$ par rapport aux fonctions g_i et par rapport à la matrice $\mathbf{B}$. Dans la version complète, on a montré que :

$$\frac{\partial}{\partial \mathbf{B}} I(\mathbf{y}) = E\left\{\boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y})\,\mathbf{x}^T\right\} \tag{45}$$

et :

$$(\nabla_{g_i} I)(x) = E\left\{\alpha_i(\mathbf{y}) \mid x_i = x\right\} \tag{46}$$

où :

$$\boldsymbol{\alpha}(\mathbf{y}) \triangleq \mathbf{B}^T \boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y}) \tag{47}$$

Pour la partie linéaire, on peut utiliser le gradient relatif :

$$\nabla_{\mathbf{B}} I(\mathbf{y}) = E\left\{\boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y})\,\mathbf{y}^T\right\} \tag{48}$$

L'algorithme final est présenté dans la figure 6.

5.2 L'approche projection

Pour utiliser cette approche, il faut résoudre le problème de la projection, c'est-à-dire, il faut calculer les fonctions g_i et la matrice $\mathbf{B}$ qui minimisent l'erreur $E\{\|\mathbf{y} - \mathbf{B}\mathbf{g}(\mathbf{e})\|^2\}$.

Pour résoudre ce problème, on propose un algorithme itératif. En effet, si on connaissait les fonctions g_k, on pourrait calculer la matrice $\mathbf{B}$ qui minimise l'erreur $E\{\|\mathbf{y} - \mathbf{B}\mathbf{x}\|^2\}$ par (36). D'autre part, les seuls contraintes sur les fonctions g_k sont le lissage et la monotonie. Alors, on propose l'algorithme de la figure 7 pour faire la projection.

Notez bien que l'on a besoin d'appliquer cet algorithme itératif à chaque itération de l'algorithme projection pour séparer les sources. Cependant, comme il est montré expérimentalement dans la version complète de la thèse, cet algorithme pour calculer la transformation projetée converge très vite (environ 5 itérations). Par conséquent, et en notant que l'algorithme général est lui-même itératif et dans chacune de ses itérations il y a peu de changement dans le système séparateur, on peut répéter l'algorithme de calcule de la transformation projetée pour un nombre petit et fixé d'itérations.

L'algorithme final est présenté dans la figure 8.

6 Les mélanges Convolutifs Post Non-Linéaires (CPNL)

Le système mélangeur-séparateur pour ce type des mélanges est montré dans la figure 9. L'algorithme de séparation pour ces mélanges est obtenu en combinant les algorithmes pour séparer des mélanges convolutifs et des mélanges PNL.

On prend $\mathbf{B}(z)$ comme un filtre RIF en degrée p, c'est-à-dire $\mathbf{B}(z) = \mathbf{B}_0 + \mathbf{B}_1 z^{-1} + \cdots + \mathbf{B}_p z^{-p}$. Pour séparer le mélange, les fonctions g_i et les matrices

- Initialization:
  1. $\mathbf{B}$=$\mathbf{I}$
  2. $\mathbf{x}$= $\mathbf{e}$
- Loop:
  1. Compute outputs by $\mathbf{y} = \mathbf{B}\mathbf{x}$.
  2. Estimate $\boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y})$ (SFD of $\mathbf{y}$).
  3. Let $\boldsymbol{\alpha}_{\mathbf{y}}(\mathbf{y}) \triangleq \mathbf{B}^T \boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y})$.
  4. For $i = 1, \ldots, N$, estimate $(\nabla_{g_i} I)(x_i)$, by fitting a smoothing spline on (x_i, α_i).
  5. For $i = 1, \ldots, N$, modify x_i by $x_i \leftarrow x_i - \mu_2 (\nabla_{g_i} I)(x_i)$.
  6. Let g_i be the smoothing spline which fits on the (e_i, x_i) points.
  7. For $i = 1, \ldots, N$, normalize x_i by $x_i \leftarrow \frac{x_i - \hat{\mu}_{x_i}}{\hat{\sigma}_{x_i}}$.
  8. Absorb the effect of the above normalization in $\mathbf{B}$:
  $$\mathbf{B} \leftarrow \mathbf{B} \begin{bmatrix} \hat{\sigma}_{x_1} & & 0 \\ & \ddots & \\ 0 & & \hat{\sigma}_{x_N} \end{bmatrix}$$
  9. Estimate $\mathbf{D} \triangleq \nabla_{\mathbf{B}} I$, using (48).
  10. Replace the main diagonal of $\mathbf{D}$ by $\mathrm{diag}(\sigma_{y_1}^2 - 1, \ldots, \sigma_{y_N}^2 - 1)$.
  11. Modify $\mathbf{B}$ by $\mathbf{B} \leftarrow (\mathbf{I} - \mu_1 \mathbf{D})\mathbf{B}$.
- Repeat until convergence

Figure 6: L'algorithme du gradient pour séparer des mélanges PNLs.

- Initialization: $\mathbf{x} = \mathbf{e}$.
- Loop:
  1. Let $\mathbf{B} = E\left\{\mathbf{y}\mathbf{x}^T\right\}\left(E\left\{\mathbf{x}\mathbf{x}^T\right\}\right)^{-1}$.
  2. Let $\mathbf{x} = \mathbf{B}^{-1}\mathbf{y}$.
  3. For $i = 1, \ldots, N$, change the order of $x_i(k)$ such that the function $x_i = g_i(e_i)$ become ascending (see the text).
- Repeat until convergence

Figure 7: L'algorithme pour trouver la transformation PNL optimale.

- Initialization: $\mathbf{y} = \mathbf{x} = \mathbf{e}$.
- Loop:
  1. Estimate $\boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y})$, the SFD of $\mathbf{y}$.
  2. Modify the outputs by $\mathbf{y} \leftarrow \mathbf{y} - \mu\boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y})$.
  3. For $k = 1, \ldots, K$, do:
     (a) Let $\mathbf{B} = E\left\{\mathbf{y}\mathbf{x}^T\right\}\left(E\left\{\mathbf{x}\mathbf{x}^T\right\}\right)^{-1}$.
     (b) Let $\mathbf{x} = \mathbf{B}^{-1}\mathbf{y}$.
     (c) For $i = 1, \ldots, N$, change the order of $x_i(k)$ such that the function $x_i = g_i(e_i)$ be ascending.
  4. For $i = 1, \ldots, N$, remove the DC of x_i and normalize its energy.
  5. For $i = 1, \ldots, N$, let g_i be the smoothing spline which fits on (e_i, x_i).
  6. Let $\mathbf{y} = \mathbf{B}\mathbf{g}(\mathbf{e})$.
  7. For $i = 1, \ldots, N$, remove the DC of y_i and normalize its energy.
- Repeat until convergence

Figure 8: L'algorithme de projection pour séparer des mélanges PNLs.

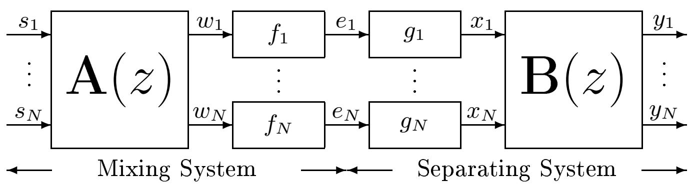

FIG. 9 – Les systèmes mélangeurs et séparateurs pour les mélanges CPNLs.

$\mathbf{B}_k$'s $(k = 0, \ldots, p)$ doivent être calculé pour que $y_1(n)$ et $y_2(n-m)$ soient indépendantes pour tous les valeurs de m et n.

Comme pour les mélanges convolutifs, on utilise $I(y_1(n), y_2(n-m))$ comme critère d'indépendance, mais dans chaque itération on prend aléatoirement un m différent entre $-M$ et M, où $M = 2p$.

6.1 L'approche gradient

Pour utiliser cette approche, les gradients de $I(y_1(n), y_2(n-m))$ par rapport aux fonctions g_i et aux matrices $\mathbf{B}_k$'s doivent être calculés. Nous avons montré dans la version complète de la thèse que :

$$\frac{\partial}{\partial \mathbf{B}_k} I\big(y_1(n), y_2(n-m)\big) = E\left\{\boldsymbol{\beta}_m(n)\,\mathbf{x}(n-k)^T\right\}, \qquad k = 0, \ldots, p \tag{49}$$

- Initialization:
  1. $\mathbf{B}(z) = \mathbf{I}$, that is: $\mathbf{B}_0 = \mathbf{I}$ and for $k = 1 \dots p$, $\mathbf{B}_k = \mathbf{0}$.
  2. $\mathbf{x}(n) = \mathbf{e}(n)$.
  3. $\mathbf{y}(n) = \mathbf{x}(n)$.
- Loop:
  1. Choose a random m from the set $\{-M, \dots, +M\}$.
  2. Compute $\boldsymbol{\beta}^\star(n)$, the SFD of $\big(y_1(n), y_2(n-m)\big)^T$.
  3. Let $\boldsymbol{\beta}_m(n) = (\beta_1^\star(n), \beta_2^\star(n+m))^T$.
  4. Estimate $(\nabla_{g_i} I)(x_i)$, by fitting a smoothing spline on (x_i, α_i).
  5. Modify x_i: $x_i \leftarrow x_i - \mu_1 (\hat{\nabla}_{g_i} I)(x_i)$.
  6. Let g_i be the smoothing spline which fits on the (e_i, x_i) points.
  7. Normalize x_i: $x_i \leftarrow (x_i - \hat{\mu}_{x_i}) / \hat{\sigma}_{x_i}$.
  8. Absorb the effect of the above normalization in $\mathbf{B}(z)$:
  $$\mathbf{B}(z) \leftarrow \mathbf{B}(z) \begin{bmatrix} \hat{\sigma}_{x_1} & & 0 \\ & \ddots & \\ 0 & & \hat{\sigma}_{x_N} \end{bmatrix}$$
  9. For $k = 0 \dots p$:
  $$\mathbf{B}_k \leftarrow \mathbf{B}_k - \mu_2 \hat{E} \left\{ \boldsymbol{\beta}_m(n)\, \mathbf{x}(n-k)^T \right\}$$
  10. Calculate $\mathbf{y}(n) = [\mathbf{B}(z)]\mathbf{x}(n)$.
  11. Normalization:
      - Let $y_i = y_i / \sigma_i$, where σ_i^2 is the energy of y_i.
      - Divide the i-th raw of $\mathbf{B}(z)$ by σ_i.
- Repeat until convergence.

Figure 10: L'algorithme du gradient pour séparer des mélanges CPNLs.

où $\boldsymbol{\beta}_m(n)$ est définit dans (39). De plus :

$$(\nabla_{g_i} I)(x) = E\left\{\alpha_i \mid x_i = x\right\} \qquad i = 1, 2 \tag{50}$$

où :

$$\boldsymbol{\alpha}(n) \triangleq \sum_{k=0}^{p} \mathbf{B}_k^T \boldsymbol{\beta}_m(n+k) = \left[\mathbf{B}^T \left(\frac{1}{z}\right)\right] \boldsymbol{\beta}_m(n) \tag{51}$$

L'algorithme final est présenté dans la figure 10.

6.2 L'approche projection

Pour la séparation des mélanges CPNLs, la projection consiste à trouver les fonctions g_i et les matrices $\mathbf{B}_k$ qui minimisent l'erreur $E\left\{ \left\| \mathbf{y}(n) - [\mathbf{B}(z)]\, \mathbf{g}\big(\mathbf{e}(n)\big) \right\|^2 \right\}$.

- Initialization: $\mathbf{x}(n) = \mathbf{e}(n)$.
- Loop:
  1. Using (44), find the best $\mathbf{B}(z)$ which maps $\mathbf{x}(n)$ to $\mathbf{y}(n)$.
  2. Compute $\mathbf{x}(n) = [\mathrm{inv}(\mathbf{B}(z))]\,\mathbf{y}(n)$ using the recursive equation:

     $$\mathbf{x}(n) = \mathbf{B}_0^{-1}\big[\mathbf{y}(n) - \mathbf{B}_1\mathbf{x}(n-1) - \cdots - \mathbf{B}_p\mathbf{x}(n-p)\big]$$

  3. For $i = 1, \ldots, N$, change the order of $x_i(k)$ such that the function $x_i = g_i(e_i)$ become ascending.
- Repeat until convergence

Figure 11: L'algorithme pour trouver la transformation CPNL optimale.

Selon une approche similaire à l'algorithme qui a été présenté pour calculer la projection d'une transformation quel conque $\mathbf{e} \mapsto \mathbf{y}$ sur des mélanges PNLs (Fig. 7), on obtient l'algorithme de la figure 11 pour calculer cette projection sur des mélanges CPNLs.

L'algorithme de projection pour la séparation des mélanges CPNL est présenté à la figure 12.

7 Conclusion

Dans ce résumé de la version complète de la thèse, on a d'abord présenté un théorème concernant la différentielle de l'information mutuelle. Puis, deux approches différentes ont été proposées pour utiliser ce théorème en minimisant l'information mutuelle des sorties d'un système séparateur. Ensuite, on a utilisé ces approches pour construire des algorithmes pour séparer les mélanges linéaires, convolutifs, PNLs et CPNLs.

Références

[1] P. Comon, "Independent component analysis, a new concept ?," *Signal Processing*, vol. 36, no. 3, pp. 287–314, 1994.

[2] H.L. Nguyen Thi and C. Jutten, "Blind sources separation for convolutive mixtures," *Signal Processing*, vol. 45, pp. 209–229, 1995.

[3] C. Simon, *Séparation aveugle des sources en mélange convolutif*, Ph.D. thesis, l'université de Marne la Vallée, Novembre 1999, (in French).

[4] M. Babaie-Zadeh, C. Jutten, and K. Nayebi, "Separating convolutive mixtures by mutual information minimization," in *Proceedings of IWANN'2001*, Granada, Spain, Juin 2001, pp. 834–842.

[5] D. Yellin and E. Weinstein, "Criteria for multichannel signal separation," *IEEE Trans. Signal Processing*, pp. 2158–2168, August 1994.

- Initialization: $\mathbf{y} = \mathbf{x} = \mathbf{e}$.
- Loop:
  1. Choose a random m from the set $\{-M, \ldots, M\}$.
  2. Estimate $\boldsymbol{\beta}_{\mathbf{y}^{(m)}}(\mathbf{y})$, the SFD of $\mathbf{y}^{(m)}$.
  3. Modify the outputs by $\mathbf{y}^{(m)} \leftarrow \mathbf{y}^{(m)} - \mu \boldsymbol{\beta}_{\mathbf{y}^{(m)}}(\mathbf{y}^{(m)})$.
  4. For $k = 1, \ldots, K$, do:
     (a) Find the best $\mathbf{B}(z)$ which maps $\mathbf{x}(n)$ to $\mathbf{y}(n)$, using (44).
     (b) Compute $\mathbf{x}(n) = [\text{inv}(\mathbf{B}(z))]\, \mathbf{y}(n)$ using the recursive equation (2).
     (c) For $i = 1, 2$, change the order of $x_i(k)$ such that the function $x_i = g_i(e_i)$ be ascending.
  5. For $i = 1, 2$, remove the DC of x_i and normalize its energy.
  6. For $i = 1, 2$, let g_i be the smoothing spline which fits on (e_i, x_i).
  7. Let $\mathbf{y}(n) = [\mathbf{B}(z)]\, \mathbf{g}\big(\mathbf{e}(n)\big)$.
  8. For $i = 1, 2$, remove the DC of y_i and normalize its energy.
- Repeat until convergence

Figure 12: L'algorithme de projection pour séparer des mélanges CPNLs.

[6] A. Taleb and C. Jutten, "Source separation in post nonlinear mixtures," *IEEE Transactions on Signal Processing*, vol. 47, no. 10, pp. 2807–2820, 1999.

[7] M. Babaie-Zadeh, C. Jutten, and K. Nayebi, "A geometric approach for separating Post Non-Linear mixtures," in *EUSIPCO*, Toulouse, France, September 2002, vol. II, pp. 11–14.

[8] M. Babaie-Zadeh, *On blind source separation in convolutive and nonlinear mixtures*, Ph.D. thesis, INP Grenoble, 2002.

[9] M. Babaie-Zadeh, C. Jutten, and K. Nayebi, "Blind separating Convolutive Post-Nonlinear mixtures," in *ICA2001*, San Diego, California, December 2001, pp. 138–143.

[10] M. Babaie-Zadeh, C. Jutten, and K. Nayebi, "Compensating frequency response of the sensors in blind separation of the sources," in *International Symposium on Telecommunications (IST2001)*, Tehran, Iran, September 2001.

[11] M. Babaie-Zadeh, C. Jutten, and K. Nayebi, "Compensation des réponse en fréquence des capteurs en séparation aveugle de sources," in *GRETSI'2001*, Toulouse, France, September 2001, pp. 399–402, (in French).

[12] M. Babaie-Zadeh, C. Jutten, and K. Nayebi, "Using joint score functions in separating post non-linear mixtures," *Scientia-Iranica*, 2002, accepted.

[13] M. Babaie-Zadeh, C. Jutten, and K. Nayebi, "Differential of mutual information function," *IEEE Signal Processing Letters*, 2002, submitted.

[14] D. T. Pham, "Estimation de la fonction score conditionnelle et l'entropie conditionnelle," Tech. Rep., 2002, (in French).

[15] J.-F. Cardoso and B. Laheld, "Equivariant adaptive source separation," *IEEE Trans. on SP*, vol. 44, no. 12, pp. 3017–3030, December 1996.

[16] S. I. Amari, "Natural gradient works efficiently in learning," *Neural Computation*, vol. 10, pp. 251–276, 1998.