

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque

--	--	--	--	--	--	--	--	--	--

THESE EN COTUTELLE

pour obtenir le grade de

DOCTEUR DE L'INPG ET DE L'UNIVERSITE SHARIF

Spécialité : «Signal, Image, Parole, Télécoms (SIPT)»

préparée au Laboratoire des Images et des Signaux (LIS)
dans le cadre de l'École Doctorale «Électronique, Électrotechnique, Automatique,
Télécommunications, Signal (EEATS)»
et à l'université technologique SHARIF

présentée et soutenue publiquement

par

Massoud BABAIEZADEH MALMIRI

le 20 septembre 2002

Titre :

On blind source separation in convolutive and nonlinear mixtures

Directeurs de thèse :

Christian JUTTEN
Kambiz NAYEBI

JURY

M. Dinh-Tuan PHAM	, Président
M. Jean-François CARDOSO	, Rapporteur
M. Pierre COMON	, Rapporteur
M. Christian JUTTEN	, Directeur de thèse
M. Kambiz NAYEBI	, Directeur de thèse
Mme. Masoomeh NASIRI	, Examinatrice

On blind source separation in convolutive and nonlinear
mixtures

Massoud BABAIE-ZADEH

To my family

Acknowledgements

First of all, I offer my special thanks to my thesis supervisor, Pr. Christian JUTTEN, who has guided me through every step of the thesis with a high attention; and I am proud of working with him during these years. I have to confess that his interest in research, and his attention to his students is exceptional and for me, very motivational; he is one of the best professors I have ever had.

Then, I acknowledge the members of the examination board: Dr. Kambiz NAYEBI, my advisor at Sharif university of technology (Tehran, Iran), Dr. Pierre COMON and Dr. Jean-François CARDOSO for accepting to be the reviewers of the thesis and their careful readings, and Dr. Dinh-Tuan PHAM and Dr. Masoomah NASIRI.

Finally, I have to thank all the members of the Laboratoire des Images et des Signaux (LIS) at Grenoble, especially Mari-Noëlle RODET, Danuta EMONET, Michel CREPIN-JOURDAN, Shahram HOSSEINI, Jordi SOLÉ i CASALS and Mohammad-Ali KHALIGHI.

Contents

1	Introduction	1
2	State of the art	7
2.1	Introduction	7
2.2	Linear instantaneous mixtures	7
2.2.1	Separability and indeterminacies	8
2.2.2	Independence criterion	8
2.2.3	Geometrical source separation algorithm	12
2.3	Convulsive mixtures	12
2.3.1	Algorithms in the time domain	13
2.3.2	Algorithms in the frequency domain	17
2.4	Nonlinear mixtures	18
2.5	Conclusion and discussion	20
3	Separability of Nonlinear Mixtures	23
3.1	Introduction	23
3.2	General nonlinear mixtures	23
3.3	Smooth transformations	24
3.4	Post Non-Linear (PNL) mixtures	27
3.5	Convulsive PNL (CPNL) mixtures	31
3.6	Conclusion	32
4	Independence Criterion	33
4.1	Introduction	33
4.2	Mutual information definition	33
4.3	Independence in the convulsive context	34
4.4	Mutual Information and gradient based algorithms	35
4.5	Multi-variate score functions	37

4.5.1	Definitions	37
4.5.2	Properties	38
4.6	Differential of the Mutual Information	42
4.7	Estimating multi-variate score functions	43
4.7.1	Estimating JSF	43
4.7.2	Estimating SFD	45
4.8	Mutual Information minimization	49
4.8.1	General nonlinear mixtures	49
4.8.2	Gradient approach	52
4.8.3	Projection approach	53
4.9	Application to linear instantaneous mixtures	53
4.9.1	Gradient approach	53
4.9.2	Projection approach	56
4.10	conclusion	58
5	Convolutive Mixtures	59
5.1	Introduction	59
5.2	Preliminary issues	59
5.3	Gradient approach	62
5.3.1	Calculating gradients	62
5.3.2	Separating algorithm	63
5.3.3	Experimental results	63
5.4	Projection approach	67
5.4.1	Calculating the optimal mapping	67
5.4.2	Separating algorithm	71
5.4.3	Experimental results	71
5.5	A special case: Post-Convolutive mixtures	72
5.5.1	Preliminary issues	72
5.5.2	Estimating equations	73
5.5.3	Separating algorithm	75
5.5.4	Experiments	75
5.6	Conclusion	77
6	Post Non-Linear Mixtures	79
6.1	Introduction	79
6.2	Geometric approach for separating PNL mixtures	80
6.2.1	Preliminary issues	80

6.2.2	Estimating the borders	81
6.2.3	Compensating for the nonlinearities	82
6.2.4	Separating the linear mixture	87
6.2.5	The algorithm	88
6.2.6	Experimental results	88
6.3	Gradient approach	90
6.3.1	Estimating equations	91
6.3.2	Separating algorithm	93
6.3.3	Experimental results	95
6.4	Projection approach	96
6.4.1	Finding the optimal system	96
6.4.2	Separating algorithm	97
6.4.3	Experimental results	98
6.5	Conclusion	101
7	CPNL Mixtures	105
7.1	Introduction	105
7.2	Gradient approach	106
7.2.1	Estimating equations	106
7.2.2	Separating algorithm	108
7.2.3	Experimental results	108
7.3	Projection approach	112
7.3.1	Finding the optimal system	113
7.3.2	Separation algorithm	114
7.3.3	Experimental results	114
7.4	Conclusion	116
8	Conclusion and Perspectives	117
A	Spline Functions	121
A.1	Definitions	121
A.2	Some interesting properties of splines	123
A.3	End-conditions in approximation with cubic splines	125
A.4	Multi-dimensional smoothing splines	126
B	Proofs	127
B.1	Theorem 4.1	127
B.2	Theorem 5.1	129

C	Scalar Score Functions	131
D	Kernel Estimators	133
D.1	Scalar density kernel estimation	133
D.2	Multivariate density kernel estimation	134
E	Some simple lemmas	137

Chapter 1

Introduction

Blind Source Separation (BSS) or Independent Component Analysis (ICA) is a relatively new subject in signal processing which has been started in the mid 80's by the work of Ans, Héroult and Jutten [7, 45, 44] when they were working on a biological problem (see [51] for historical notes on BSS). The problem consists in retrieving unobserved independent mixed signals from mixtures of them, assuming there is information neither about the original source signals, nor about the mixing system (hence the term *Blind*¹). This problem has a lot of applications in different areas including feature extraction, brain imaging, telecommunications, speech enhancement, *etc* [68].

Suppose we have N observed signals $x_1(t), \dots, x_N(t)$ which are assumed to be the mixtures of N independent source signals $s_1(t), \dots, s_N(t)$. Note that here the number of sources is assumed to be equal to the number of observations. Then $\mathbf{s}(t) \triangleq (s_1(t), \dots, s_N(t))^T$ and $\mathbf{x}(t) \triangleq (x_1(t), \dots, x_N(t))^T$ are called the *source vector* and the *observation vector*, respectively. Hence, $\mathbf{x}(t) = \mathcal{F}(\mathbf{s}(t))$, where \mathcal{F} is the unknown *mixing system*. In the general form, \mathcal{F} may be nonlinear or may have memory. The goal of BSS is to construct a *separating system* \mathcal{G} (Fig. 1.1) in order to isolate in each component of the *output vector* $\mathbf{y}(t) \triangleq \mathcal{G}(\mathbf{x}(t))$ the image of one source. In other words, each component of the output vector $\mathbf{y}(t)$ must depend on only one component of $\mathbf{s}(t)$. In general case, this relation may be nonlinear or have memory. To summarize, the goal is to obtain:

$$y_i(t) = h_i(s_{\sigma(i)}(t)), \quad i = 1, \dots, N \quad (1.1)$$

where σ is a permutation, and h_i stands for any invertible mapping.

Since the sole information about the sources is their statistical independence, one can try to construct the separating system \mathcal{G} in such a way that the output vector has independent

¹Strictly speaking, a totally blind solution is not possible because we need some assumptions about the general form of the mixing system (linear instantaneous, convolutive, ...).

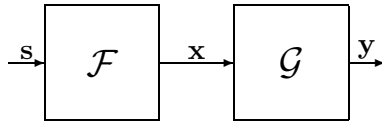


Figure 1.1: Mixing and separating systems.

components. The important question is now: Does the independence of the components of \mathbf{y} (ICA) imply necessarily the separation of the sources (BSS) in the sense of (1.1)?

The answer of this question, which regards to the *separability* of a mixing model, is not positive in the general case. This problem will be considered in more details in the chapter 3.

However, in the context of factor analysis, Darmois [32] and Skitovich [92] have proved the following theorem for linear mappings \mathcal{F} (refer to [52] for better accessibility):

Theorem 1.1 (Darmois-Skitovich) *Let x_1, \dots, x_N be some independent random variables, and define the random variables y_1 and y_2 by:*

$$\begin{cases} y_1 \triangleq a_1x_1 + a_2x_2 + \dots + a_Nx_N \\ y_2 \triangleq b_1x_1 + b_2x_2 + \dots + b_Nx_N \end{cases} \quad (1.2)$$

If y_1 and y_2 are independent, all x_k 's for which $a_k b_k \neq 0$ are Gaussian.

This theorem shows that nongaussian sources cannot be mixed linearly (and instantaneously) and generate independent outputs. Consequently, if the mixing-separating system is linear and instantaneous (memoryless), the independence of the outputs insures the separation of the sources. In other words, *linear instantaneous* mixtures, are separable. By a linear instantaneous mixture we mean a mixture of the form $\mathbf{x} = \mathbf{A}\mathbf{s}$, where \mathbf{A} is called the *mixing matrix*. Then, a *separating matrix* must be estimated to generate independent component outputs $\mathbf{y} = \mathbf{B}\mathbf{x}$. However, there is two *indeterminacies* in estimating the sources in a linear instantaneous mixture: a permutation and a change of scale. This can be seen from the fact that the permutation of the sources or changing their energies has no effect on their independence.

In many applications, an instantaneous (memoryless) mixing model may be unsuitable. One can then consider *linear convolutive* mixtures. For these mixtures, the mixing matrix is composed of linear filters:

$$\mathbf{x}(t) = [\mathbf{A}(z)] \mathbf{s}(t) \triangleq \sum_k \mathbf{A}_k \mathbf{s}(t - k) \quad (1.3)$$

Naturally, the separating system, too, is chosen as a filter $\mathbf{B}(z)$ to generate $\mathbf{y}(t) = [\mathbf{B}(z)] \mathbf{x}(t)$. It has been shown [108] that these mixtures, too, are separable. However, in this case, the

scale indeterminacy of the linear instantaneous mixtures is enlarged to a filtering indeterminacy. In other words, although the output independence results in the separation of the sources, the resulting outputs are not the original sources, but filtered versions of them. This indeterminacy is more severe than the scale indeterminacy of linear instantaneous mixtures. However, as it has been shown in [91], the effect of each source on each sensor (that is, what each sensor sees when all the other sources are zero) can be calculated after this separation (this fact will be more explained in the next chapter).

The equivalency of ICA and BSS problems disappears as soon as we leave the linear domain. We mention here an example which is derived from [98]:

Example. Suppose that s_1 is a Rayleigh distributed random variable with the Probability Density Function (PDF) $p_{s_1}(s_1) = s_1 \exp(-s_1^2/2)$ and s_2 is a random variable, independent of s_1 , with a uniform distribution on $[0, 2\pi)$. Consider now the nonlinear mapping:

$$\begin{cases} y_1 = s_1 \cos(s_2) \\ y_2 = s_1 \sin(s_2) \end{cases} \quad (1.4)$$

Then the Jacobian of this transformation is:

$$\mathbf{J}(s_1, s_2) = \begin{bmatrix} \cos(s_2) & -s_1 \sin(s_2) \\ \sin(s_2) & s_1 \cos(s_2) \end{bmatrix} \quad (1.5)$$

and:

$$p_{y_1 y_2}(y_1, y_2) = \frac{p_{s_1 s_2}(s_1, s_2)}{|\mathbf{J}(s_1, s_2)|} = \frac{1}{2\pi} \exp\left(-\frac{y_1^2 + y_2^2}{2}\right) = \left(\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{y_1^2}{2}\right)\right) \left(\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{y_2^2}{2}\right)\right) \quad (1.6)$$

Consequently y_1 and y_2 will be two *independent* Gaussian random variables with zero means and unit variances.

This example shows that nonlinear systems can provide independent outputs without separating the sources. Hence *in the general case, nonlinear mixtures are not separable*.

Although nonlinear mixtures are not separable in the most general case, we still can find some practically important subclasses of them which are separable. An example is the (instantaneous) *Post Non-Linear* (PNL) mixtures which is introduced by Taleb and Jutten [98]. In the PNL mixtures, an instantaneous linear mixture is followed by component-wise and invertible nonlinearities (see Fig. 1.2):

$$e_i = f_i\left(\sum_j a_{ij} s_j\right), \quad i = 1, \dots, N \quad (1.7)$$

where e_i 's are the observations. This type of mixing, corresponds to the (realistic enough) case where the mixture is itself linear, but the sensors have nonlinear memoryless effects, such as saturation. For separating these mixtures, one can use the mirror structure of

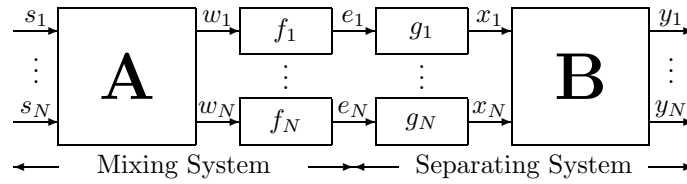


Figure 1.2: The mixing-separating system for PNL mixtures.

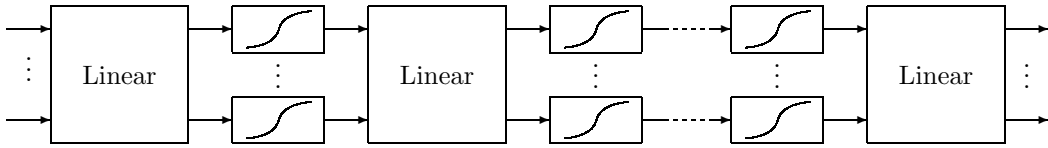


Figure 1.3: The mixing-separating system for a cascade of PNL mixtures.

the mixing system: compensating first for the nonlinear effects of the sensors, and then separating the resulting linear mixtures. Taleb and Jutten [98] have proved that, for this mixing-separating structure, the output independence implies the separation of the sources². Moreover, provided that the signals are really mixed before nonlinearities (*i.e.* if there is at least two non-zero elements in each row or each column of \mathbf{A}), then the indeterminacies are the same as linear instantaneous mixtures: a scale and a permutation indeterminacy.

An extension of PNL mixtures, is *Convolutional Post Non-Linear* (CPNL) mixtures, in which a linear convolutive mixture is followed by component-wise nonlinearities (the same system as in Fig. 1.2, but where \mathbf{A} and \mathbf{B} are convolutive). For independent identically distributed (iid) sources, the separability of this model can be deduced from the separability of PNL mixtures (see section 3.5).

Another extension of PNL mixtures is a cascade of PNL mixing blocks (see Fig. 1.3). This model appears for example when the signals are amplified by several amplifying stages during their propagations, or when the mixing system can be well modeled by a constrained³ Multi-Layer Perceptron (MLP). By now, the separability of this model is still an open question (see also [99]).

In this thesis, the extension of PNL mixtures to CPNL mixtures is addressed. The materials of the thesis are organized as follows:

- Chapter 2 contains a brief state-of-the-art of the blind source separation problem, which consists of spotlights on a few important results.

²Another proof will be given in section 3.4 for bounded sources.

³For separability using a mirror structure, it is intuitively required that each linear block is a square invertible matrix.

- In chapter 3, the problem of separability in nonlinear mixtures is considered. In this chapter, we present a new proof for the separability of PNL mixtures (for bounded sources), and we point out the separability of CPNL model. We also show that in general nonlinear mixtures, the *smoothness* of the nonlinear transformation does not insure the separability. In other words, for separating nonlinear mixtures, *having a structural constraint seems to be essential*.
- The independence criterion which is used (the mutual information) is considered in details in Chapter 4. We obtain an expression for its *stochastic gradient*, and we consider the estimation of this gradient, too. We also show that the mutual information has *no local minimum* (see theorem 4.2). Moreover, in this chapter we introduce two general approaches for separating the sources using the gradient of the mutual information: gradient and projection approaches. As an example, we use these approaches for separating linear instantaneous mixtures. In this chapter (see section 4.8.1), one can find a method for general nonlinear ICA (which is not equivalent to source separation).
- Separating convolutive mixtures by means of the gradient and the projection approaches is considered in Chapter 5. Moreover, a special kind of the convolutive mixtures, called *Post Convolutive* mixtures, is considered in this chapter. We will show that for this special convolutive mixture, contrary to general convolutive mixtures, the instantaneous independence of the outputs is sufficient for separating the sources.
- Chapter 6 is devoted to PNL mixtures. In this chapter, in addition to gradient and projection approaches, we present a geometric approach for separating the PNL mixtures of bounded sources. As a remark, the geometric approach shows the possibility of nonlinear compensation before separating the sources.
- The CPNL mixtures is considered in chapter 7. Two algorithms, based on gradient and projection approaches, are proposed.
- Finally, chapter 8 contains the conclusion and perspectives for future works.

Chapter 2

State of the art

2.1 Introduction

The problem of Blind Source Separation (BSS) has been first introduced by Ans, Héroult and Jutten [7, 45, 44] for linear instantaneous mixtures. Then, many researchers have been attracted by the subject, and many other works appeared. For example, see [26, 14, 21, 61, 54, 96, 69, 74, 6, 98, 57, 79] which are some of the most important papers on linear instantaneous BSS, and [68] which is a recently published book on the subject. A good overview of the problem can be found in [18].

The early works on the BSS and ICA problems concerned linear instantaneous mixtures, and by now, a lot of algorithms are available for separating them. Then, as an extension to the instantaneous mixtures, the convolutive mixtures have been considered by some researchers since early 90s [50, 100, 62, 23, 108, 58, 91].

Nonlinear mixtures have been much less considered in the literature [15, 75, 107, 98, 5], and till now a very few results are available. One reason is of course the mathematical difficulty of nonlinear systems, but another important reason is the fact that nonlinear mixtures are not separable (the separability problem in nonlinear mixtures is considered in more details in chapter 3).

2.2 Linear instantaneous mixtures

The simplest BSS model is the linear instantaneous model, in which, the N observed signals $x_1(t), \dots, x_N(t)$ are assumed to be linear instantaneous mixtures of N zero-mean and statistically independent source signals $s_1(t), \dots, s_N(t)$:

$$x_i(t) = \sum_{j=1}^N a_{ij} s_j(t), \quad j = 1, \dots, N \quad (2.1)$$

where a_{ij} are unknown constants. It must be noted that here the number of sources is assumed to be equal to the number of observations. This model is compactly represented by the matrix equation $\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t)$, or simply $\mathbf{x} = \mathbf{A}\mathbf{s}$, where $\mathbf{A} \triangleq [a_{ij}]$ denotes the unknown *mixing matrix*, $\mathbf{s} \triangleq (s_1, \dots, s_N)^T$ is an $N \times 1$ column vector collecting the source signals (usually called the *source vector*), and similarly $\mathbf{x} \triangleq (x_1, \dots, x_N)^T$ is the *observation vector*. The (linear instantaneous) BSS problem consists in finding an $N \times N$ *separating matrix*, such that the *output vector* $\mathbf{y} = \mathbf{B}\mathbf{x}$ is an estimate of the source vector \mathbf{s} .

2.2.1 Separability and indeterminacies

The sole information about the source signals is their independence. If the components of \mathbf{y} are statistically independent, are they necessarily equal to the sources, *i.e.* $\mathbf{B} = \mathbf{A}^{-1}$? We can easily find some linear transformations which preserve the independence. For example, permuting the sources, or changing their energies does not change their independence. On the other hand, if s_1 and s_2 are both zero mean unit variance Gaussian sources, and \mathbf{A} is any unitary matrix (rotation transformation), it can be easily verified that y_1 and y_2 will be independent, too. However, it has been shown [26] that these cases are the only cases for which a linear transformation preserves the independence. More precisely, Comon has shown in [26] that “if there is at most one Gaussian source, then the independence of the components of \mathbf{y} (and even pairwise independence) implies $\mathbf{B}\mathbf{A} = \mathbf{P}\mathbf{D}$, where \mathbf{P} and \mathbf{D} represent a permutation and a diagonal matrix, respectively”. In other words, the linear instantaneous mixtures are *separable*, up to some trivial indeterminacies (a permutation and a change of scale), provided that there is at most one Gaussian source.

2.2.2 Independence criterion

From decorrelation to HOS

It is well known that the independence is much more than decorrelation. In fact, transforming the observation vector to decorrelated outputs, which is usually called Principal Component Analysis (PCA), is not sufficient for separating the sources (ICA).

Another way to state the insufficiency of the output decorrelation for solving the BSS problem is as follows. For estimating the $N \times N$ matrix \mathbf{A} , taking into account N scale indeterminacies, we must determine $N^2 - N = N(N - 1)$ unknown coefficients. The decorrelation constraints $E\{y_i y_j\} = 0$ for all pairs $1 \leq i \neq j \leq N$, gives us $N(N - 1)/2$ equations, which is not sufficient for determining \mathbf{A} : the second order independence (decorrelation) of the outputs only does “*half of the ICA job*”. This fact shows that for finding the other required equations in a fully blind context, higher order independence constraints must be

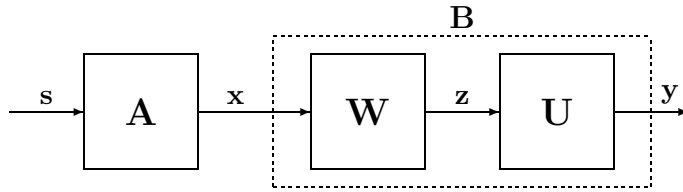


Figure 2.1: Decorrelation leaves an unknown rotation.

used. It also shows why the Gaussian sources cannot be separated: they have no higher (than 2) order statistics.

Finally, it is interesting to note that the second order independence, reduces the ICA problem to a rotation. In fact, suppose that the energies of the outputs are normalized and consider the factorization $\mathbf{B} = \mathbf{U}\mathbf{W}$ of the mixing matrix, where \mathbf{W} is the (spatial) whitening matrix of the observations. In other words, for $\mathbf{z} = \mathbf{W}\mathbf{x}$, we have¹ $E\{\mathbf{z}\mathbf{z}^T\} = \mathbf{I}$. Now, since the outputs are independent, then from $E\{\mathbf{y}\mathbf{y}^T\} = \mathbf{U}E\{\mathbf{z}\mathbf{z}^T\}\mathbf{U}^T = \mathbf{I}$, we deduce $\mathbf{U}\mathbf{U}^T = \mathbf{I}$, that is, \mathbf{U} is a unitary (rotation) matrix. Consequently, the ICA transformation can be seen as the cascade of a whitening and a rotation transformation (see Fig. 2.1).

The idea of the first work on BSS (HJ algorithm [44]) is that if, for two odd nonlinear functions f and g , $f(y_i)$ and $g(y_j)$ ($i \neq j$) are decorrelated, then from the Taylor expansion of f and g , we deduce that all the cross-moments of y_i and y_j are zero, and hence these variables are independent. Then, the algorithm is based on an adaptive manner for canceling $E\{f(y_i)g(y_j)\}$. Although this claim is not true in the general case, and it has been shown that there exists source distributions and nonlinear functions for which the algorithm converges to spurious solutions [94, 59].

Looking for the other necessary independence constraints, some algorithms [50, 56, 22, 71, 61, 19] have used Higher Order Statistics (HOS) methods and mainly canceling 4th order cross-cumulants [73] of the outputs (since 3rd order cumulants vanish for symmetric distributions, they are not usually used in BSS).

Contrast functions

The concept of *contrast functions* for source separation has been first presented by Comon [26], inspired from the contrast functions for deconvolution [37]. A contrast function for source separation (or simply a contrast) is a real valued function of the distribution of a random vector which is minimized when the source separation achieved. In other words, a contrast

¹One way for finding this matrix is to use the Cholesky decomposition of the covariance matrix of the observations. In other words, \mathbf{W} is an upper triangular matrix which satisfies $\mathbf{R}_x = \mathbf{W}\mathbf{W}^T$, where $\mathbf{R}_x \triangleq E\{\mathbf{x}\mathbf{x}^T\}$ denotes the covariance matrix of \mathbf{x} .

function ϕ satisfies $\phi\{\mathbf{Cs}\} \geq \phi\{\mathbf{s}\}$ for any independent component random vector \mathbf{s} , and the equality holds if and only if $\mathbf{C} = \mathbf{PD}$, where \mathbf{P} and \mathbf{D} are a permutation and a diagonal matrix, respectively. Consequently, a source separation algorithm can be based on the minimization of a contrast function of the outputs.

As an example, it has been shown [26, 18] that the sum of the 4th order cross-cumulants of the components is a contrast function (provided that the prewhitening has been done). Other examples can be found in [26, 18, 68].

Mutual Information

Another independence criterion or contrast function is the mutual information of the outputs. The mutual information of the random variables y_1, \dots, y_N is defined as the Kullback-Leibler divergence² [27] of $p_{\mathbf{y}}(\mathbf{y})$ and $\prod_i p_{y_i}(y_i)$:

$$I(\mathbf{y}) \triangleq \int_{\mathbf{y}} p_{\mathbf{y}}(\mathbf{y}) \ln \frac{p_{\mathbf{y}}(\mathbf{y})}{\prod_i p_{y_i}(y_i)} d\mathbf{y} \quad (2.2)$$

where $\mathbf{y} \triangleq (y_1, \dots, y_N)^T$ and p denotes the Probability Density Function (PDF). Mutual information can also be expressed by the equation:

$$I(\mathbf{y}) = \sum_{i=1}^N H(y_i) - H(\mathbf{y}) \quad (2.3)$$

where $H(\cdot) \triangleq -E\{\ln p_{(\cdot)}(\cdot)\}$ denotes the Shannon's entropy. From the well known properties of the Kullback-Leibler divergence [27], we know that $I(\mathbf{y})$ is always non-negative, and is zero if and only if $p_{\mathbf{y}}(\mathbf{y}) = \prod_i p_{y_i}(y_i)$, that is, if y_1, \dots, y_N are independent. Consequently, $I(\mathbf{y})$ is a contrast function for source separation, and separation algorithms can be designed based on its minimization.

Mutual information and Maximum Likelihood

Source separation based on the minimization of the mutual information of the outputs has another nice property which makes this approach very attractive: *it is asymptotically a Maximum Likelihood (ML) estimation of the sources* [18, 95]. Consequently, more and more recent works are based on this criterion [96, 78, 5]. This is also the criterion we have used throughout this thesis.

²The Kullback-Leibler divergence between two probability density functions $f(x)$ and $g(x)$ is defined as $\int_{-\infty}^{+\infty} f(x) \ln \frac{f(x)}{g(x)} dx$. It is well known that this quantity is always non-negative, and is zero if and only if f and g are identical. Although it is not a distance (it is not symmetric), it can be seen as a measure of the closeness of f and g .

BSS and mutual information

The mutual information depends on the densities of the random variables, which must be estimated from the data. One method for the estimation of the PDF of a random variable, consists in writing an expansion like Edgeworth or Gram-Charlier [55]. This approach leads again to cumulant-based contrast functions [26]. Another approach based on the mutual information, is to calculate its gradient with respect to the separating matrix, and to estimate this gradient from the data [96]. This approach points out the relevance of the score function (the log-derivative of the density) of a random variable.

BSS and Non-Gaussianity

Another important class of source separation algorithms is based on the nongaussianity of the outputs [35, 69, 60, 67]. To state the idea, suppose that the whitening $\mathbf{z} = \mathbf{W}\mathbf{x}$ has been done, and hence the unitary (rotation) matrix \mathbf{U} must be estimated to achieve independent outputs. Now, from $\mathbf{y} = \mathbf{U}\mathbf{z}$ we have $p_{\mathbf{y}}(\mathbf{y}) = p_{\mathbf{z}}(\mathbf{z})/|\det \mathbf{U}| = p_{\mathbf{z}}(\mathbf{z})$. Consequently, $H(\mathbf{y}) = H(\mathbf{z})$ and $I(\mathbf{y}) = \sum_i H(y_i) - H(\mathbf{z})$. Since $H(\mathbf{z})$ does not depend on \mathbf{U} , minimizing $I(\mathbf{y})$ with respect to \mathbf{U} is equivalent to minimizing the sum of the marginal entropies. Moreover, $-H(y_i)$ can be seen as the Kullback-Leibler divergence between the density of y_i and a zero-mean unit-variance Gaussian density (up to a constant term). This leads us to this conclusion that \mathbf{U} must be estimated to produce the outputs as nongaussian as possible. This fact has a nice intuitive interpretation: from the central limit theorem [77] we know that the mixing tends to gaussianize the observations, and hence the separating system should go to the opposite direction. A well-known algorithm based on the nongaussianity of the outputs is FastICA [67] which uses *negentropy*³ as a measure of nongaussianity.

Semi-blind approaches

Up to now, no time structure has been assumed about the sources (like time correlation or nonstationarity). However, usual practical signals are not series of independent and identically distributed (iid) samples, and this fact can be used in source separation (see also [20]). If we know that the sources have time correlations (*i.e.* they are colored signals which is equivalent to dropping the first ‘i’ in iid), then in addition to $E\{y_i y_j\} = 0$, one can use the decorrelation at different time lags:

$$E\{y_i(t) y_j(t - \tau)\} = 0 \quad , \text{ for all } i, j, \tau \quad (2.4)$$

³The negentropy of a random vector (or random variable) is defined as the difference between its entropy and the entropy of a Gaussian random vector with the same covariance matrix.

This idea, which requires the existence of some time structure, results in second order separation algorithms [101, 68] (note that each $\tau \neq 0$ adds a set of $N(N - 1)$ equations). In addition to their simplicity, an advantage of second order algorithms is that they can be applied for Gaussian sources, too.

On the other hand, if it is assumed that the sources are non-stationary (dropping the second ‘i’ of iid), then we can divide the signals into short windows and consider the covariances in each one:

$$E_{t \in T_k} \{y_i(t)y_j(t)\} \quad (2.5)$$

where $T_k = (kT, (k + 1)T]$. Then, the “joint diagonalization” of the covariance matrices at different segments can separate the sources [65, 82].

2.2.3 Geometrical source separation algorithm

Another method for source separation is the geometric source separation algorithm [83, 64]. Since, we have generalized this method to Post Non-Linear (PNL) mixtures (see section 6.2), we state it with some more details.

This approach, which holds essentially for two sources and two sensors, is based on a geometrical interpretation of the independence of two random variables. To state the idea more clearly, suppose that the marginal PDF’s of the sources s_1 and s_2 are non-zero only within the intervals $M_1 \leq s_1 \leq M_2$ and $N_1 \leq s_2 \leq N_2$. Then, from the independence of s_1 and s_2 , we have $p_{s_1 s_2}(s_1, s_2) = p_{s_1}(s_1)p_{s_2}(s_2)$, and hence the support of $p_{s_1 s_2}(s_1, s_2)$ will be the rectangular region $\{(s_1, s_2) \mid M_1 \leq s_1 \leq M_2, N_1 \leq s_2 \leq N_2\}$. In other words, the scatter plot of the source samples forms a rectangular region in the (s_1, s_2) plane. The linear mapping $\mathbf{x} = \mathbf{A}\mathbf{s}$ transforms this region into a parallelogram region. Figure 2.2 shows the scatter plot of the sources and the observations for two uniform random sources. Now, without loss of generality (due to the scale indeterminacy), one can write:

$$\mathbf{A} = \begin{bmatrix} 1 & a \\ b & 1 \end{bmatrix} \quad (2.6)$$

then it can be seen that the slopes of the borders of the scatter plot of the observations will be b and $1/a$. Hence, if we estimate the slopes of the borders of this parallelogram, we can find the mixing matrix, and then easily separate the sources.

2.3 Convolutional mixtures

In linear instantaneous mixtures, it has been implicitly assumed that the difference between the effect of one source on two different sensors is only a scale factor. However, in many

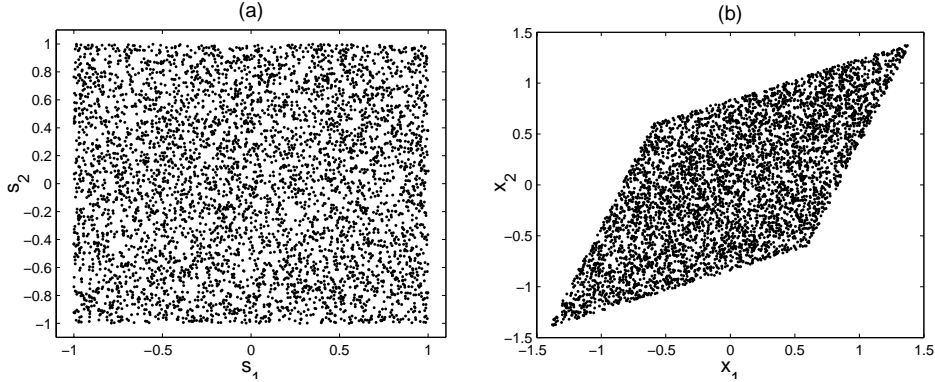


Figure 2.2: The joint distributions of \mathbf{s} and \mathbf{x} for a linear instantaneous mixture.

applications this is not the case, because one has to take into account the propagation in the medium, from one source to different sensors. Consequently, each source passes through different transfer functions for arriving to the different sensors.

The existing algorithms for solving this problem can be mainly divided in two different categories: the algorithms in the time domain (*eg.* [25, 50, 100, 23, 24, 41, 58, 108, 62, 42]) and the algorithms in the frequency domain (*eg.* [106, 17]). Recently, a third approach based on time-frequency representation has also been investigated [87, 86]. They are based on the implicit source separation in the time-frequency plane, provided that the signals are sparse in time and frequency. It seems to be very efficient, especially for speech signals.

2.3.1 Algorithms in the time domain

The model

In solving a convolutive BSS problem, we are dealing with the mixing model:

$$x_i(t) = \sum_{j=1}^N a_{ij}(t) * s_j(t) \quad (2.7)$$

where $a_{ij}(t)$ denotes the impulse response from the j -th source to the i -th sensor, and ‘*’ is the convolution operator.

In discrete form, the model can be written as:

$$x_i(n) = \sum_{j=1}^N [A_{ij}(z)] s_j(n) \quad (2.8)$$

where $A_{ij}(z) = \sum_k a_{ij}(k) z^{-k}$ is the transfer function between the j -th source and the i -th

sensor, and $[A_{ij}(z)]s_j(n)$ stands for the filtering operation:

$$[A_{ij}(z)]s_j(n) \triangleq \sum_{k=-\infty}^{+\infty} a_{ij}(k)s_j(n-k) \quad (2.9)$$

We can also rewrite these equations in the matrix form:

$$\mathbf{x}(n) = [\mathbf{A}(z)]\mathbf{s}(n) \triangleq \sum_k \mathbf{A}(k)\mathbf{s}(n-k) \quad (2.10)$$

where $\mathbf{A}(k)$ is a matrix with entries $a_{ij}(k)$.

The mixtures is then separated by a separating matrix $\mathbf{B}(z) = [B_{ij}(z)]$ composed on linear filters:

$$\mathbf{y}(n) = [\mathbf{B}(z)]\mathbf{x}(n) \quad (2.11)$$

Separability and indeterminacies

The convolutive mixing model is more complicated than the linear instantaneous mixing model, and less considered in the literature (moreover, it has mainly addressed for the simple case of two sources and two sensors). It has been proved [108] that the convolutive mixtures are *separable*, that is, in a convolutive mixing-separating system, the independence of the outputs insures the separation of the sources. But, the indeterminacies in this case are not as trivial as in the instantaneous case: the permutation indeterminacy still exists, but the scaling one now becomes a filtering indeterminacy. This can be seen from the fact that if $s_1(n)$ and $s_2(n)$ are independent, then $[H_1(z)]s_1(n)$ and $[H_2(z)]s_2(n)$ are independent, too, for any invertible filters H_1 and H_2 . This indeterminacy may be unacceptable, since it can strongly distort the sources. However, as it has been proposed by Simon [91], after the separation of the sources, the effect of each source on each sensor can be estimated. To clarify the idea, suppose that after the separation (*i.e.* after the independence of the outputs), $y_i(n)$ is a filtered version of $s_i(n)$. Now, let $H_{ji}(z)$ be the filter which minimizes:

$$E \left\{ (x_j(n) - [H_{ji}(z)]y_i(n))^2 \right\} \quad (2.12)$$

then, from the independence of the sources we can say that $[H_{ji}(z)]y_i(n) = [A_{ji}(z)]s_i(n)$. In other words, we can estimate the effect of the i -th source on j -th sensor, which is what the j -th sensor sees when all the sources but the i -th one are zero.

Time domain algorithms can be divided into two main categories: second order approaches [41, 58, 104], and higher order approaches [50, 108, 100, 23, 24]. One can also mention algorithms [1, 62, 42, 36] that convert first the convolutive mixture to an instantaneous mixture based on a second order approach, and this instantaneous mixture would be separated with higher order approaches.

Second order approaches

These approaches can be seen as the generalization of the Widrow's Adaptive Noise Cancellation (ANC) principle [105]. His adaptive noise canceling system is a special convolutional mixture, in which we already know that one of the sources (called *signal*) has no influence on one of the observations (called *noise reference*).

Weinstein *et. al.* [104] have used a second order approach for the case of 2 sources, and 2 sensors. In their approach, it is assumed that $A_{11}(z) = A_{22}(z) = B_{11}(z) = B_{22}(z) = 1$, which is not restrictive, because of the filtering indeterminacy. Then, they use the fact that the decorrelation of the outputs is equivalent to cancel the cross Power Spectral Density (PSD) of the outputs [77], that is, $S_{y_1 y_2}(\omega) \equiv 0$. Then, it is obvious that at each frequency, we have one equation, but two unknowns ($B_{12}(\omega)$ and $B_{21}(\omega)$), which is not sufficient for determining them. Hence, Weinstein *et. al.* assume that the sources are *nonstationary*. In fact, they divide the signals into different segments, and they assume that in each segment the sources are stationary, but their PSD is not the same in different segments. This assumption permits them to construct the other required equations and separate the sources.

An adaptive second order algorithm has been developed by Van Gerven and Van Compernelle [41]. This algorithm is based on an adaptive manner for canceling $E\{y_1(t)y_2(t-\tau)\}$ for $\tau = 0$ and $\tau = \deg(B_{12})$, and $E\{y_2(t)y_1(t-\tau)\}$ for $\tau = 0$ and $\tau = \deg(B_{21})$. Consequently, for insuring the uniqueness of the solution, they need to assume that the diagonal elements of $\mathbf{A}(z)$ and $\mathbf{B}(z)$ are 1, and that A_{12} , A_{21} , B_{12} and B_{21} are Finite Impulse Response (FIR), which is somewhat restrictive. They also assume that $\det(\mathbf{A}(z))$ is minimum phase (which is required for having a causal inverse).

Lindgren and Broman [58] have proved that, under some assumptions, the decorrelation of the outputs has a unique answer, and they developed an algorithm for decorrelating the outputs. In their approach, it is first assumed that the sources are purely nondeterministic, that is, they can be stated as the (linear) filtered versions of some white noises. They also assume that $A_{11}(z) = A_{22}(z) = B_{11}(z) = B_{22}(z) = 1$ and that A_{12} , A_{21} , B_{12} and B_{21} are FIR. They also need that $\mathbf{A}(z)$ be minimum phase, and causal, and in each $B_{ij}(z)$ ($i \neq j$), there is at least two non-zero coefficients. They also need that $\deg(B_{ij}(z)) \geq \deg(A_{ij}(z))$.

Essentially second order approaches

These algorithms use a second order approach to convert the convolutional mixture to an instantaneous mixture, which will be separated by a higher order approach.

Abed-Meraim *et. al.* [1], Gorokhov and Loubaton [42] and Mansour *et. al.* [63] have used subspace methods for separating the sources. These algorithms require the knowledge of the

exact degree of the mixing filters. Moreover, the number of sensors must be *strictly* greater than the number of the sources. Another difficulty of this approach is the existence of very large matrices, which results in high computational cost.

Delfosse and Loubaton [35, 36] have developed a linear prediction approach. In this method, it is assumed that the sources can be stated as the outputs of some linear systems which are excited by unit-variance white noises. If we denote these white noises by $\boldsymbol{\nu}(t) = (\nu_1(t), \dots, \nu_N(t))^T$, it can be seen that $\boldsymbol{\nu}(t)$ is a normalized innovation process⁴ of $\mathbf{x}(t)$. Hence, a normalized innovation process of $\mathbf{x}(n)$ is first estimated with a linear prediction approach, and it is converted to another independent component innovation process by means of a linear instantaneous ICA method. The resulting innovation, is the innovation processes of the sources, which can be used to determine the effect of each source on each sensor.

Higher order approaches

It seems that the first work on the convolutive mixtures has been proposed by Jutten *et. al.* [50], as an attempt for extending the HJ algorithm to convolutive mixtures. In this algorithm, an adaptive manner has been proposed for eliminating $E\{f(y_1(n))g(y_2(n-m))\}$, where f and g are two odd nonlinearities. N. Charkani *et. al.* [23, 24] have used the same separation criterion, but with a different separating system.

In [100], Nguyen and Jutten proposed an algorithm for separating the convolutive mixtures based on the cancellation of the 4th order cross cumulant $\text{Cum}_{22}(y_1(n), y_2(n-k))$.

Yellin and Weinstein [108] proposed an algorithm based on the cancellation of the cross bi-spectra or tri-spectra of the outputs. The polyspectrum of the joint random processes $x_1(t), \dots, x_k(t)$ associated with the set of indices $k_0, k_1, \dots, k_m \in \{1, 2, \dots, k\}$ is defined by:

$$\begin{aligned} P_{x_{k_0} x_{k_1} \dots x_{k_m}}(\omega_1, \dots, \omega_m) &\triangleq \sum_{\tau_1} \dots \sum_{\tau_m} \text{Cum}(x_{k_0}(t), x_{k_1}(t+\tau_1), \dots, x_{k_m}(t+\tau_m)) e^{-j \sum_{i=1}^m \omega_i \tau_i} \\ &= \mathcal{F} \{ \text{Cum}(x_{k_0}(t), x_{k_1}(t+\tau_1), \dots, x_{k_m}(t+\tau_m)) \} \end{aligned} \quad (2.13)$$

⁴The *innovation process* of a vectorial stochastic process $\mathbf{x}(n)$, is the process $\mathbf{i}(n)$ defined by $\mathbf{i}(n) = [\mathbf{P}(z)]\mathbf{x}(n)$, where $\mathbf{P}(z) = 1 + \sum_{k \geq 1} \mathbf{P}_k z^{-k}$ is the filter which minimizes:

$$\mathbf{P}(z) = \underset{\mathbf{Q}}{\text{argmin}} E \left\{ \left\| \mathbf{x}(n) + \sum_{k=1}^{\infty} \mathbf{Q}_k \mathbf{x}(n-k) \right\|^2 \right\}$$

Now, if $\mathbf{R}_i \triangleq E\{\mathbf{i}(n)\mathbf{i}^T(n)\}$ is the covariance matrix of \mathbf{i} , and \mathbf{L} is a matrix satisfying $\mathbf{R}_i = \mathbf{L}\mathbf{L}^T$, then it can be seen that the covariance matrix of $\boldsymbol{\nu}(n) \triangleq \mathbf{L}^{-1}\mathbf{i}(n)$ is \mathbf{I} . Then, $\boldsymbol{\nu}(n)$ is called the *normalized innovation process* of $\mathbf{x}(n)$. It can be easily verified that, for any unitary matrix \mathbf{U} , $\mathbf{U}\boldsymbol{\nu}(n)$ is also an innovation process of $\mathbf{x}(n)$.

They prove that, under some mild assumptions, the joint cancellation of the bi-spectra $P_{y_1^* y_1 y_2}(\omega_1, \omega_2)$ and $P_{y_2^* y_2 y_1}(\omega_1, \omega_2)$ implies the separation of the sources. This requires that $P_{y_i^* y_i y_i}(\omega_1, \omega_2)$ is not identically zero, which is not satisfied if y_i has a symmetric density. For symmetric densities, they propose to use the joint cancellation of the tri-spectra $P_{y_i^* y_i y_j^*}(\omega_1, \omega_2, \omega_2)$ ($i \neq j$) for separating the sources.

2.3.2 Algorithms in the frequency domain

In frequency domain approaches [106, 17], the mixing system is first written in the frequency domain:

$$\mathbf{X}(f) = \mathbf{A}(f)\mathbf{S}(f) \quad (2.14)$$

which is like an instantaneous mixture (but with a varying mixing matrix). There are two difficulties for using the instantaneous ICA techniques for separating this instantaneous model:

- If we divide the frequency band into different frequency bins, and if we assume that in each bin $\mathbf{A}(f)$ is approximately constant (but complex-valued), then for the i -th frequency bin we can write:

$$\mathbf{X}^{(i)}(f) = \mathbf{A}^{(i)}\mathbf{S}^{(i)}(f) \quad (2.15)$$

which corresponds to an instantaneous mixture, that can be separated by using an instantaneous BSS technique. This gives an estimation of the sources $\hat{\mathbf{S}}_k^{(i)}(f)$ ($k = 1, \dots, N$) in that bin. However, we know that in the instantaneous BSS methods, the sources are estimated with a permutation and a scale indeterminacies. Consequently, a manner must be found for (a) associating an estimated source in a frequency bin to one of the estimated sources in the neighbor frequency bin, and (b) adjusting its energy.

- From the central limit theorem we know that the Fourier transform of a random source tends to be Gaussian. But the instantaneous BSS techniques based on high order statistics are not efficient for the distributions near Gaussian: one has to prefer second order algorithms using extra information like time correlation or nonstationarity of sources.

Servière *et. al.* [16, 17, 29] have considered this approach for the case that the sources come from rotating machines. They have shown that the Fourier transform of these sources does not become Gaussian. Then in each frequency bin a method based on PCA followed by canceling 4-th order cross-cumulants has been used for separating the instantaneous mixture in each frequency bin. Finally, for solving the permutation problem, they introduce a method based on the coherence between the sources estimated in adjacent frequency bins.

2.4 Nonlinear mixtures

An (instantaneous) nonlinear mixing system, in its most general form, is a mapping $\mathbf{x} = \mathcal{F}(\mathbf{s})$, where \mathbf{x} and \mathbf{s} denote the observation and source vectors, respectively. One has then to find another mapping $\mathbf{y} = \mathcal{G}(\mathbf{x})$ such that the components of \mathbf{y} be independent (ICA). However, in nonlinear mappings, independence does not imply source separation (BSS). Hence, nonlinear mixtures are not separable, and the independence assumption is not strong enough to lead to source estimation. In other words, for nonlinear mixtures, a totally blind solution is not possible and one must have some extra information *e.g.* about the structure of the mixing system. In the next chapter, we will consider the separability problem of nonlinear mixtures with more details.

The problem of nonlinear mixtures traces back to [49], where C. Jutten used soft nonlinear mixtures in order to assess the robustness and the performance of the HJ algorithm.

Later, Burel [15] proposed a neural network based solution. In his algorithm, it is assumed that the mixing structure is known but depends on unknown parameters. Then, despite the restricted model, the algorithm tries to find independent outputs by means of a very complicated independence criterion.

Pajunen *et al.* [75] used the Kohonen's Self-Organizing Maps (SOM) for separating the nonlinear mixtures. The SOM [43] is a well known mapping method that, in an unsupervised manner, learns a nonlinear mapping from the data into a (usually) two-dimensional grid. It is based on quantizing the data with an array of neurons. The SOM can be used to map the data so that it would be uniformly distributed on the rectangular grid. Then they use this approach to find a mapping that transforms the observations to a uniformly distributed grid data. Moreover, this approach counts on the smoothness of the transformation provided by a SOM⁵. One difficulty of this approach is the tendency to create uniformly distributed outputs. To solve this problem, in [76], Generative Topographic Mappings (GTP) has been used as an alternative to SOM's [68]. In this approach, the sources can have any distribution, but it must be *previously known*.

Deco and Brauer [34] also addressed the problem, considering a volume preservation condition on the nonlinear transforms. Yang *et al.* [107] have also studied the problem for special kind mixtures, with the assumption that the inverse nonlinearities can be approximated with a two layer perceptron.

Valpola *et al.* [102, 103] proposed an ensemble learning approach. In their approach, the mapping from \mathbf{s} to \mathbf{x} is modeled by a Multi-Layer Perceptron (MLP) neural network, and

⁵In section 3.3 we will present an example of a *smooth* nonlinear mapping which maps two *uniform* independent random variables to two other *uniform* independent random variables. Consequently, the SOM can do nothing for separating it.

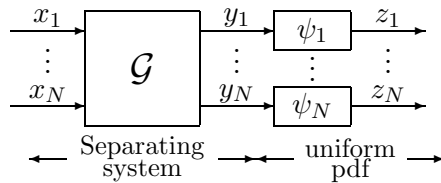


Figure 2.3: The idea used by Almeida.

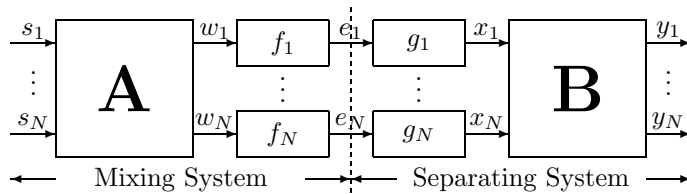


Figure 2.4: The mixing-separating system for PNL mixtures.

the learning procedure is based on the unsupervised Bayesian ensemble learning.

Another method (used for both linear and smooth nonlinear mixtures), which is based on the outputs' mutual information, has been proposed by Almeida [5]. This approach can be seen as an extension of the INFOMAX method of Bell and Sejnowsky [14]. In this approach, the independence criterion is the mutual information of the outputs. However, for its minimization, they first apply invertible componentwise nonlinearities on each output (see Fig. 2.3). It can be shown that the mutual information remains unchanged by componentwise invertible nonlinearities, and hence $I(\mathbf{y}) = I(\mathbf{z}) = \sum_i H(z_i) - H(\mathbf{z})$. Almeida uses the Cumulative Probability Functions (CPF) of y_i 's as the nonlinearities and hence each z_i is always uniform and $H(z_i)$ is constant. Consequently, minimizing $I(\mathbf{y})$ is equivalent to maximizing $H(\mathbf{z})$ (in the original INFOMAX algorithm, ψ_i 's are some fixed functions). Then, he uses a neural network approach for maximizing $H(\mathbf{z})$ and generating independent outputs.

Most of the above works, even if they are based on a good intuition, lacked theoretical justification. In fact, in most of the cases, the goal has been to find independent component outputs, which, in general nonlinear mixtures, is not equivalent to the separation of the sources. In some works, this is justified by forcing the transformation to be *smooth* (with \mathcal{G} modeled by neural networks for instance), with the hope that a smooth transformation cannot both preserve the independence and mix the sources. However, as we will see in the next chapter, this is not the case (see section 3.3). Consequently, for separating nonlinear mixtures, adding some knowledge about the structure of the mixing system seems to be essential.

The Post Non-Linear (PNL) mixtures that have been presented in the previous chapter, have been first introduced by A. Taleb and C. Jutten [98, 97, 95], as a realistic nonlinear mixture which is separable. Figure 2.4 shows the mixing-separating system. These mixtures have also been considered by some other researchers [2, 3, 109]. For separating these mixtures, Taleb and Jutten have used the mutual information of the outputs as the separation criterion, and then developed a gradient based approach for adjusting the parameters of the separating system. In [98] they used MLP's for modeling the compensating nonlinearities, but in [97] they proposed a non-parametric approach for estimating the nonlinearities. This non-parametric approach has been developed more clearly in [2].

2.5 Conclusion and discussion

In this chapter we briefly reviewed the blind source separation methods for linear (instantaneous and convolutive) and instantaneous nonlinear mixtures.

With a look at the existing methods for separating the convolutive and the PNL mixtures, we notice that the extension to Convolutive Post Non-Linear (CPNL) mixtures is not obvious. The difficulty comes from the fact that the criteria used for separating the convolutive and the PNL mixtures are different. In PNL mixtures, the criterion used is the mutual information of the outputs. Till now, this criterion has never been applied for separating convolutive mixtures. Moreover, the extension of methods used in convolutive mixtures to PNL mixtures is not possible or is very difficult. For example, it does not seem that the 4-th order cumulant methods is suitable for nonlinear mixtures, because although the fourth order independence is sufficient for separating linear mixtures, its sufficiency in nonlinear mixtures is not obvious. In fact, in [98] the authors have shown the inefficiency of the estimation of a PDF with the Gram-Charlier expansion for separation of PNL mixtures, which is equivalent to using 4th order cumulants.

Consequently, we will first develop a method for separating the convolutive mixtures based on minimizing the mutual information of the outputs, *i.e.* the same criterion used for separating PNL mixtures. This method is presented in the chapter 5.

Moreover, the methods of separating instantaneous PNL mixtures cannot be easily extended to the convolutive case. In fact, as we will see in the chapter 4, the method of Taleb and Jutten, relies highly on the instantaneous relation $\mathbf{y} = \mathbf{B}\mathbf{x}$ (see Fig. 3.4) and $p_{\mathbf{y}}(\mathbf{y}) = p_{\mathbf{x}}(\mathbf{x})/|\det \mathbf{B}|$, for calculating the gradients of the mutual information of the outputs with respect to the nonlinearities. To overcome this problem, we compute an expression for the differential of the mutual information in the general case (see Theorem 4.1). Using this differential, we first develop another method for the separation of the PNL mixtures

based on minimization of the mutual information of the outputs (it is the subject of the chapter 6).

Finally, the combination of the different approaches for separating convolutive and PNL mixtures, results in an algorithm for separating CPNL mixtures, which is presented in the chapter 7.

Chapter 3

Separability of Nonlinear Mixtures

3.1 Introduction

In this chapter, the problem of *separability* is considered. A model is called separable, if the independence of the outputs implies the separation of the sources, that is, if ICA and BSS are equivalent for that model.

To be more precise, consider a model $\mathbf{y}(t) = \mathcal{H}(\mathbf{s}(t))$. This model is called separable, if the statistical independence of the components of \mathbf{y} insures that \mathcal{H} is the trivial transformation:

$$y_i(t) = h_i(s_{\sigma(i)}(t)) , \quad i = 1, \dots, N \quad (3.1)$$

where σ denotes a permutation. In other words, the separability of a model, enables us to separate the sources only using the independence of the outputs.

As we saw in chapter 1, in the general case, such a property does not hold. Especially, for any random vector it is always possible to find nonlinear systems which transform it to a random vector with independent components [70].

It must be emphasized that source separation in the sense of (3.1), is not itself equivalent to retrieve the original sources. This is because of the remaining distortion introduced by the systems h_i , which can strongly modify the source signals.

3.2 General nonlinear mixtures

In linear instantaneous domain, the transformation which maps a non-Gaussian random vector with independent components to another random vector with independent components must be trivial (which is here a mixture of permutation and scaling), that is, it is unique up to some trivial transformation. This property is a direct result of the Darmois-Skitovich theorem, which has been mentioned in Chapter 1. However, in nonlinear mappings, such a

transformation is by no means unique, and hence the nonlinear mixtures are not separable. A simple example has been mentioned in chapter 1.

This result has been established in early 50's by Darmois [31] (in factor analysis domain), where he used a simple constructive method for decomposing any random vector as a non-trivial mapping of independent variables.

The uniqueness and existence of the transformations which preserve the independence has also been addressed by Hyvarinen and Pajunen [70]. As it has been shown in their paper and is strongly inspired from the Darmois' work, it is always possible to construct a transformation which maps any random vector to a random vector with uniform distribution on the hypercube $[0, 1]^N$ (which clearly has independent components). Moreover, they have presented a Gram-Schmidt like recursion for actually constructing such a transformation.

These results show the existence of non-trivial transformations \mathcal{H} which still “mix” the variables while preserving their statistical independence. This is a negative result, because it shows that, for general nonlinear systems, without constraints on the transformation model, source separation is simply “impossible” by only using the statistical independence.

In the conclusion of [31], Darmois clearly states: “*These properties [...] clarify the general problem of factor analysis by showing the large indeterminacies it presents as soon as one leaves the field, already very wide, of linear diagrams*”.

3.3 Smooth transformations

At the first glance, one may think that the nontrivial mappings which preserve the independence are very complicated mappings. Hence, there are some works on nonlinear BSS (e.g. [75, 5]) which constraint the nonlinear mapping to be a smooth transformation, with the hope to avoid the nontrivial transformations with this regularization.

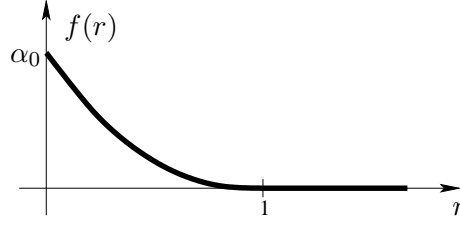
However, this is not true. The following example shows that even smooth transformations are able to mix independent sources and preserve their statistical independence. Consequently, in nonlinear source separation, we prefer to add structural constraints which are more restrictive than constraining the mapping to be smooth.

Example. Let s_1 and s_2 be two independent random variables. Now consider a rotation in such a way that the angle of rotation depends on $\|\mathbf{s}\|$, where $\mathbf{s} = (s_1, s_2)^T$:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}, \alpha = f\left(\sqrt{s_1^2 + s_2^2}\right) \quad (3.2)$$

where f is a differentiable function. The Jacobian matrix of this transformation is:

$$\mathbf{J} \triangleq \begin{bmatrix} \frac{\partial y_1}{\partial s_1} & \frac{\partial y_1}{\partial s_2} \\ \frac{\partial y_2}{\partial s_1} & \frac{\partial y_2}{\partial s_2} \end{bmatrix} \quad (3.3)$$

Figure 3.1: The graph of f defined in (3.12).

It can be easily seen that:

$$\frac{\partial y_1}{\partial s_1} = \left(1 - s_2 \frac{\partial \alpha}{\partial s_1}\right) \cos \alpha - s_1 \frac{\partial \alpha}{\partial s_1} \sin \alpha \quad (3.4)$$

$$\frac{\partial y_1}{\partial s_2} = -\left(1 + s_1 \frac{\partial \alpha}{\partial s_2}\right) \sin \alpha - s_2 \frac{\partial \alpha}{\partial s_2} \cos \alpha \quad (3.5)$$

$$\frac{\partial y_2}{\partial s_1} = \left(1 - s_2 \frac{\partial \alpha}{\partial s_1}\right) \sin \alpha + s_1 \frac{\partial \alpha}{\partial s_1} \cos \alpha \quad (3.6)$$

$$\frac{\partial y_2}{\partial s_2} = \left(1 + s_1 \frac{\partial \alpha}{\partial s_2}\right) \cos \alpha - s_2 \frac{\partial \alpha}{\partial s_2} \sin \alpha \quad (3.7)$$

and hence:

$$\mathbf{J} = \begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix} \begin{bmatrix} 1 - s_2 \frac{\partial \alpha}{\partial s_1} & -s_2 \frac{\partial \alpha}{\partial s_2} \\ s_1 \frac{\partial \alpha}{\partial s_1} & 1 + s_1 \frac{\partial \alpha}{\partial s_2} \end{bmatrix} \quad (3.8)$$

From this relation, we will have:

$$\det(\mathbf{J}) = 1 + s_1 \frac{\partial \alpha}{\partial s_2} - s_2 \frac{\partial \alpha}{\partial s_1} \quad (3.9)$$

However, it can be seen that:

$$s_1 \frac{\partial \alpha}{\partial s_2} = s_2 \frac{\partial \alpha}{\partial s_1} = \frac{s_1 s_2}{\sqrt{s_1^2 + s_2^2}} f' \left(\sqrt{s_1^2 + s_2^2} \right) \quad (3.10)$$

consequently $\det(\mathbf{J}) = 1$. Finally, we have:

$$\boxed{p_{y_1 y_2}(y_1, y_2) = p_{s_1 s_2}(s_1, s_2)} \quad (3.11)$$

Consider now for example:

$$r \in \mathbb{R}^+ \Rightarrow f(r) \triangleq \begin{cases} \alpha_0 (1 - r)^n & ; \text{if } 0 \leq r \leq 1 \\ 0 & ; \text{if } r \geq 1 \end{cases} \quad (3.12)$$

The graph of this function is sketched in Fig. 3.1. This is a smooth function (it is $n - 1$ times differentiable everywhere). Moreover, with f defined as (3.12), the transformation (3.2) maps the region $-1 \leq s_1, s_2 \leq 1$ to $-1 \leq y_1, y_2 \leq 1$. Consequently, from (3.11), we conclude that this mapping transforms two independent random variables with uniform distributions on $(-1, 1)$ to two other independent random variables with the same distributions. That is, in spite of its smoothness, this transformation is not trivial but preserves the independence. Figure 3.2 shows the resulting mapping for $n = 2$ and $\alpha_0 = \pi/2$ and Fig. 3.3 shows the input and output joint distributions for two independent random variables with uniform distributions on $(-1, 1)^1$.

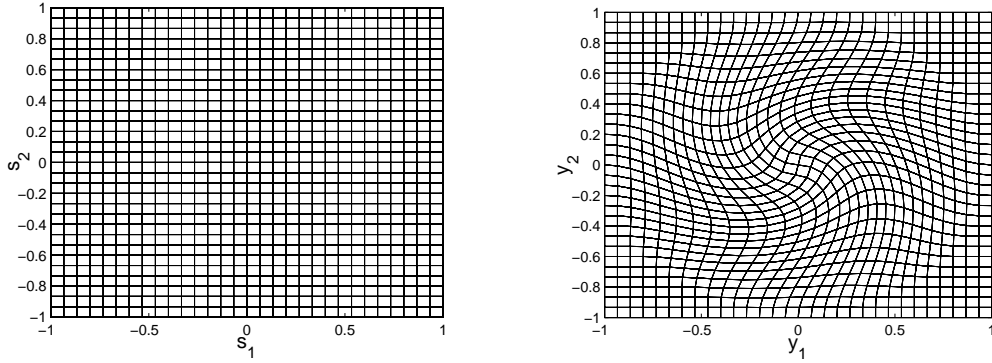


Figure 3.2: A smooth mapping.

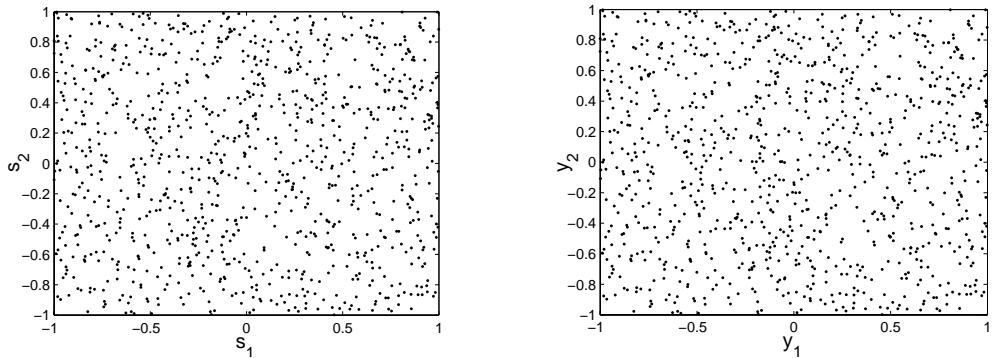


Figure 3.3: The input-output joint distributions for the mapping of Fig. 3.2.

The above example shows that, a regularization constraint (looking for a smooth mapping) is not sufficient for separating nonlinear mixtures. Then, we suggest to use structural constraints for avoiding non-trivial transformations. In other words, a totally *Blind* solution is not possible, and we need to have, or assume, a certain structure for the mixing system, and consequently for the separating system, too.

Kagan *et. al.* [53] have been extended the Darmois-Skitovich theorem (chapter 1) to a subclass of nonlinear mappings, and its usage in BSS has been considered by Eriksson and Koivunen [38]. Here, we will consider two other important subclass of nonlinear mappings, for which a separability result can be stated.

¹Notice that any random variable x , can be transformed to a uniform random variable on $[0, 1]$ with the mapping $y = F_x(x)$, where F_x is the Cumulative Probability Function (CPF) of x . Hence, we can construct many other independence preserving transformation based on this example.

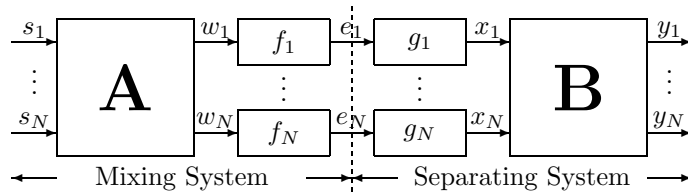


Figure 3.4: The mixing-separating system for PNL mixtures.

3.4 Post Non-Linear (PNL) mixtures

One important subclass of separable nonlinear mixtures, is the so called Post Non-Linear (PNL) mixtures (we will design separation algorithms for these mixtures in chapter 6). As it has been explained in chapter 1, a PNL mixture consists of a linear mixture followed by componentwise nonlinearities, which correspond to nonlinear effects of the sensors. Fig. 3.4 shows the mixing-separating system for these mixtures. As it can be seen in this figure, the separating system first tries to compensate for the sensor nonlinearities and then separate the resulting linear mixture.

The separability of this mixing-separating system has been already pointed out [98, 95]. Here, we propose another proof for the separability of these mixtures for bounded sources. For sake of simplicity, we first state the proof for the case of two sources and two sensors, and then we point out how it can be extended to more general cases. This proof will also result in a method for separating PNL mixtures (section 6.2).

The proof is based on a geometric interpretation of the statistical independence. In fact, if the sources s_1 and s_2 are independent (with a unimodal bounded distribution), then the joint distribution of (s_1, s_2) forms a rectangular region in the (s_1, s_2) plane. In fact, note that $p_{s_1 s_2}(s_1, s_2) = p_{s_1}(s_1)p_{s_2}(s_2)$. Hence, if $p_{s_1}(s_1)$ and $p_{s_2}(s_2)$ have the supports $M_1 \leq s_1 \leq M_2$ and $N_1 \leq s_2 \leq N_2$, respectively, then the support of $p_{s_1 s_2}(s_1, s_2)$ will be the rectangular region $\{(s_1, s_2) \mid M_1 \leq s_1 \leq M_2, N_1 \leq s_2 \leq N_2\}$. After applying the linear transformation \mathbf{A} , this rectangular region will be converted to a parallelogram region in (w_1, w_2) plane (see Fig. 3.5). As we have seen in section 2.2.3, the geometric source separation method for linear mixtures, is based on the slope estimation of the parallelogram borders, which determines the mixing matrix.

For PNL mixtures, the output independence, means that the joint distribution of (y_1, y_2) is a rectangular region, and hence the joint distribution of (x_1, x_2) is another parallelogram. Consequently, if we prove that the only componentwise transformation which maps a parallelogram to a parallelogram, is a linear transformation, then the separability of PNL mixtures will be proved (*i.e.* it proves that output independence results in the linearity of the functions

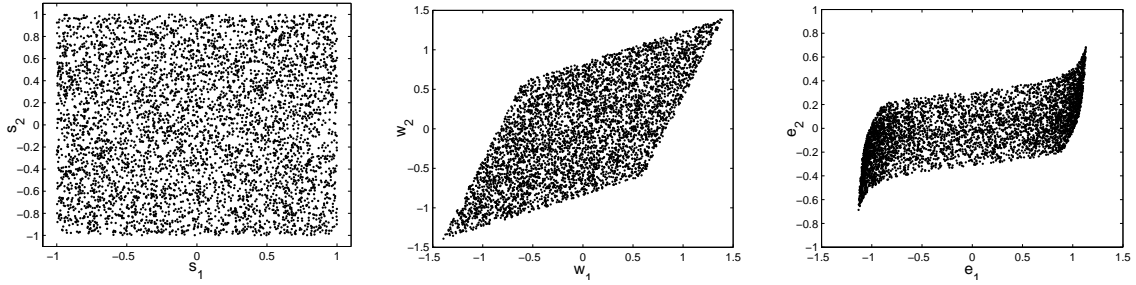


Figure 3.5: The joint distributions in s , w and e planes for PNL mixtures.

$g_i \circ f_i$, and consequently in separating the sources up to the classical indeterminacies of linear instantaneous mixtures).

We first present the following theorem, which states such a property for the ‘borders’ of a parallelogram:

Theorem 3.1 *Consider the transformation:*

$$\begin{cases} x_1 = h_1(w_1) \\ x_2 = h_2(w_2) \end{cases} \quad (3.13)$$

where h_1 and h_2 are analytic functions². If the borders of a parallelogram in the (w_1, w_2) plane are transformed to the borders of a parallelogram in the (x_1, x_2) plane, and the borders of these parallelograms are not parallel to the coordinate axes, then, there exist constants a_1 , a_2 , b_1 and b_2 such that:

$$\begin{cases} h_1(x) = a_1x + b_1 \\ h_2(x) = a_2x + b_2 \end{cases} \quad (3.14)$$

Before proving the theorem, we mention a few remarks.

Remark 1: The above theorem is stated for the borders of the parallelograms, not the whole region itself. Hence, to use it for proving the separability of PNL mixtures, we must prove that the borders of a parallelogram in (w_1, w_2) plane will be transformed to the borders of the parallelogram in (x_1, x_2) plane. This can be done with the assumption that h_1 and h_2 are *monotonous* (which is equivalent to assume that they are invertible). In fact, with (3.13), $w_2 = \text{cte}$ is mapped into $x_2 = \text{cte}$. Moreover, as it has been shown in Fig. 3.6, if h_1 and h_2 are monotonous, then the point order on this line remains unchanged (h_1 increasing) or reversed (h_1 decreasing). Therefore, the borders of any transformed region in (x_1, x_2) are the mappings of the borders of the corresponding region in (w_1, w_2) . Hence, the above theorem also proves that the only componentwise mapping (with monotonous nonlinearities) which transforms a parallelogram region to a parallelogram region is a generalized linear mapping.

²A function is called analytic on an interval, if it can be expressed with a Taylor series on that interval.

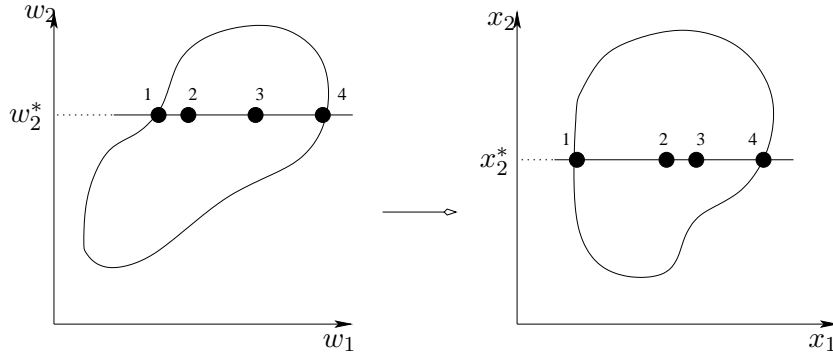


Figure 3.6: Under a componentwise mapping with monotone nonlinearities, the borders of a region will be mapped to the borders of its transformation.

Remark 2: Note that the theorem holds only when the borders of the parallelogram are not parallel to the coordinate axes. In fact, any componentwise transformation will transform a rectangle with the borders parallel to coordinate axes to another rectangle. This condition, is another statement of the fact that in PNL mixtures, we need that the sources are really mixed before nonlinearities, that is, there must be at least two non-zero elements in each row or each column of the mixing matrix \mathbf{A} .

Remark 3: The theorem shows that, as for linear mixtures with bounded sources, the borders of the joint plot are sufficient for separating the sources.

Remark 4: The existence of the constants b_1 and b_2 , emphasizes on a DC³ indeterminacy in separating PNL mixtures. This indeterminacy exists also in linear mixtures but is generally skipped by assuming zero-mean sources.

Remark 5: The theorem suggests a geometrical method for separating PNL mixtures: first, estimate the borders of the scatter plot of (e_1, e_2) , then find the unique (up to a scale and a DC indeterminacies, as suggested by (3.14)) componentwise transformation which maps these borders to a parallelogram, and finally separate the resulting linear mixture. We will develop this method in details in section 6.2. However, although the proof is mathematically applicable to all independent and bounded sources, the separation method is practically restricted to the sources that permit a good estimation of the borders of the scatter plot using a finite number of observations.

The proof of the theorem requires the following lemmas:

Lemma 3.1 *Let f be an analytic function on the interval $D_f \subseteq \mathbb{R}$ such that $0 \in D_f$. Suppose that for all x in a neighborhood of 0 we have:*

$$f(x) = c_1 f(c_2 x) \quad (3.15)$$

³We use the term DC (Direct Current) of a waveform $x(n)$, for its mean $E\{x(n)\}$.

where $c_1 \neq 1$ and $c_2 \neq 1$.

1. If $\exists n \in \mathbb{N}$ such that $c_1 c_2^n = 1$, then $\exists a \in \mathbb{R}$ such that $f(x) = a x^n$, $\forall x \in D_f$,
2. if $\forall n \in \mathbb{N}, c_1 c_2^n \neq 1$, then $f(x) = 0$, $\forall x \in D_f$.

Proof. From (3.15), we deduce $f^{(n)}(x) = c_1 c_2^n f^{(n)}(c_2 x)$ where $f^{(n)}(x)$ denotes the n -th order derivative of f . By letting $x = 0$, we conclude that, $\forall n$ such that $c_1 c_2^n \neq 1$, we have $f^{(n)}(0) = 0$. Consequently, in the Taylor expansion of f around $x = 0$:

$$f(x) = f(0) + \frac{1}{1!} f'(0) x + \frac{1}{2!} f''(0) x^2 + \cdots + \frac{1}{n!} f^{(n)}(0) x^n + \cdots, \quad \forall x \in D_f \quad (3.16)$$

the coefficients of all the terms, except probably the n -th which satisfies $c_1 c_2^n \neq 1$, are zero. Hence, if there is such an integer, then $f(x) = a x^n$; otherwise, $f(x) = 0$. \blacktriangle

Lemma 3.2 Consider the componentwise transformation in (3.13) with invertible nonlinearities. Suppose that the transformation of two intersecting lines $w_2 = c_1 w_1$ and $w_2 = c_2 w_1$ which are not parallel to coordinate axes, is two intersecting lines $x_2 = d_1 x_1$ and $x_2 = d_2 x_1$. Then, there exists real constants a_1 and a_2 and an integer n such that:

$$\begin{cases} x_1 = h_1(w_1) = a_1 w_1^n \\ x_2 = h_2(w_2) = a_2 w_2^n \end{cases} \quad (3.17)$$

Proof. Since the transformation is one to one, the transformed lines $x_2 = d_1 x_1$ and $x_2 = d_2 x_1$ cannot be parallel to the coordinate axes, too. That is, d_1 and d_2 are neither zero nor infinite. Now, from the assumptions of the lemma, we have

$$\begin{cases} h_2(c_1 w) = d_1 h_1(w) \\ h_2(c_2 w) = d_2 h_1(w) \end{cases} \quad (3.18)$$

Combining these equations, with a little algebra, leads to:

$$h_2(w) = \frac{d_1}{d_2} h_2\left(\frac{c_2}{c_1} w\right) \quad (3.19)$$

The conditions of the lemma 3.1 are now satisfied (note that $c_1 \neq c_2$ and $d_1 \neq d_2$), and we conclude that $h_2(w) = a_2 w^n$. Then, from this result and (3.18), we also conclude $h_1(w) = a_1 w^n$. \blacktriangle

Proof of theorem 1. In the proof, we assume, without loss of generality (the general case is trivial by changing the variables), that $h_1(0) = h_2(0) = 0$ and the origin is a corner of the parallelogram.

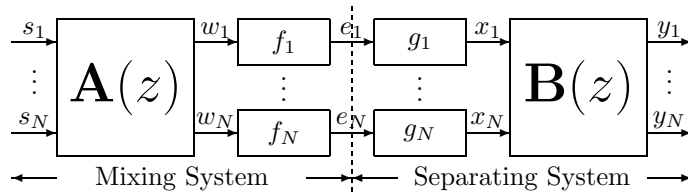


Figure 3.7: The mixing-separating system for CPNL mixtures.

Consider two borders of the parallelogram crossing the origin. Since the mappings of $w_2 = c_1 w_1$ and $w_2 = c_2 w_1$ (the borders in the (w_1, w_2) plane) are $x_2 = d_1 x_1$ and $x_2 = d_2 x_1$ (the borders in the (x_1, x_2) plane), the conditions of the lemma 3.2 are satisfied, and the transformation must be in the form of (3.17). Now, if $n \neq 1$, then the mappings of the other borders, which are not crossing the origin, cannot be straight lines. Hence, $n = 1$, *i.e.* h_1 and h_2 are both linear. \blacktriangle

Generalization to more than two sources and two sensors: For the case of more than two sources and two sensors, we can repeat the above proof by considering two pairs w_i and w_j . Here the support of $p_{w_i w_j}(w_i, w_j)$ (and hence the scatter plot of (w_i, w_j)) is not a parallelogram⁴, because of the effects of the other sources. However, in this case too, if we assume that in each row of \mathbf{A} there is at least two non zero elements, then the scatter plot of (w_i, w_j) contains straight line borders which are not parallel to coordinate axes. Consequently, by considering an intersecting pair of these borders, we conclude that the transformation must be in the form of (3.17). But n must be 1, because the transformation of the other borders (which do not pass through the intersection of the two first ones) are still straight lines.

3.5 Convolutional PNL (CPNL) mixtures

As it has been mentioned in chapter 1, one extension to the PNL mixtures is the Convolutional Post Non-Linear (CPNL) mixtures. CPNL mixtures consist of a convolutional mixing system followed by componentwise nonlinearities, as shown in Fig. 3.7. Like PNL mixtures, the nonlinearities correspond to nonlinear effects of the sensors. The design of separation algorithms of these mixtures will be considered in details in chapter 7.

The separating system is composed of componentwise nonlinear blocks, g_i , such that $x_i = g_i(e_i)$ and of a linear separating filter, $\mathbf{B}(z)$, such that the estimated sources are $\mathbf{y}(n) = [\mathbf{B}(z)] \mathbf{x}(n)$. For iid sources and FIR mixing filters, the separability of CPNL mixtures

⁴It is a polygon, with $2N$ edges (N is the number of sources) which are parallel two by two.

can be directly deduced from the separability of instantaneous PNL mixtures. In fact, denoting $\mathbf{A}(z) = \sum_n \mathbf{A}_n z^{-n}$, and:

$$\mathbf{s} = (\dots, \mathbf{s}^T(n-1), \mathbf{s}^T(n), \mathbf{s}^T(n+1), \dots)^T \quad (3.20)$$

$$\mathbf{e} = (\dots, \mathbf{e}^T(n-1), \mathbf{e}^T(n), \mathbf{e}^T(n+1), \dots)^T \quad (3.21)$$

we have:

$$\mathbf{e} = \mathbf{f}(\bar{\mathbf{A}}\mathbf{s}) \quad (3.22)$$

where \mathbf{f} acts componentwise, and:

$$\bar{\mathbf{A}} = \begin{bmatrix} \dots & \dots & \dots & \dots & \dots \\ \dots & \mathbf{A}_{n+1} & \mathbf{A}_n & \mathbf{A}_{n-1} & \dots \\ \dots & \mathbf{A}_{n+2} & \mathbf{A}_{n+1} & \mathbf{A}_n & \dots \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix} \quad (3.23)$$

The iid nature of the sources insures the spatial independence of \mathbf{s} . Then, the CPNL mixtures can be viewed as a particular PNL mixtures. For FIR mixing matrix $\mathbf{A}(z)$, (3.22) corresponds to a finite dimension PNL mixture and the separability holds. For more general filter (IIR) matrix, (3.22) is an infinite dimension PNL mixture, and the separability can be conjectured.

3.6 Conclusion

In this chapter, we considered the separability problem in nonlinear mixtures. First, by using an example, we showed that smooth nonlinear systems may preserve the statistical independence with still mixing the sources. This property shows that, in separating nonlinear mixtures, using some structural constraints is essential.

Then, we pointed out the separability of two special nonlinear mixtures, that is, PNL and CPNL mixtures. In fact, we have shown, that for these mixtures the independence of the outputs insures the separation of the sources, provided that the separating system be the mirror structure of the mixing system.

Chapter 4

Independence Criterion

4.1 Introduction

In this chapter, we deal with the mutual information, as the independence criterion we have used for separating different mixing models. First of all, we emphasize that, in convolutive context, “independence” is in the sense of independence of stochastic processes, while in instantaneous context, it is in the sense of independence of random variables. This fact will result in more complicated independence criterion in convolutive mixtures.

Then, we consider the mutual information more deeply, and we will find an expression for its differential, which requires multivariate score functions for random vectors. After considering the properties of these score functions, we discuss their empirical estimation. After that, we will present two different approaches for using the differential of the mutual information in source separation: the first one consists in calculating the gradient of the outputs’ mutual information with respect to the components of the separating system, and the second one is a “projection method”. Finally, as an example, we use these approaches for designing new separation algorithms for linear instantaneous mixtures.

4.2 Mutual information definition

Recall from the chapter 2 that the mutual information of random variables x_1, x_2, \dots, x_N is defined by the Kullback-Leibler divergence between $p_{\mathbf{x}}(\mathbf{x})$ and $\prod_{i=1}^N p_{x_i}(x_i)$:

$$I(\mathbf{x}) = \text{KL} \left(p_{\mathbf{x}}(\mathbf{x}) \parallel \prod_{i=1}^N p_{x_i}(x_i) \right) = \int_{\mathbf{x}} p_{\mathbf{x}}(\mathbf{x}) \ln \frac{p_{\mathbf{x}}(\mathbf{x})}{\prod_i p_{x_i}(x_i)} d\mathbf{x} = \sum_i H(x_i) - H(\mathbf{x}) \quad (4.1)$$

where $\mathbf{x} = (x_1, \dots, x_N)^T$, $p_{\mathbf{x}}$ and p_{x_i} are the PDFs of \mathbf{x} and x_i , respectively, and H denotes the Shannon’s entropy. From the properties of the Kullback-Leibler divergence, we deduce

that the mutual information is always non-negative, and is zero if and only if $p_{\mathbf{x}}(\mathbf{x}) = \prod_{i=1}^N p_{x_i}(x_i)$, *i.e.* when x_1, \dots, x_N are independent. This property of the mutual information makes it an interesting independence criterion for blind source separation, and it has been already used frequently in BSS and ICA (*e.g.* [26, 96, 18, 98, 78, 5]). In fact, for separating the sources, we must minimize the mutual information of the outputs. Moreover, as it has been mentioned in chapter 2, source separation based on minimization of the mutual information is asymptotically a Maximum Likelihood (ML) estimation of the sources [18, 95].

4.3 Independence in the convolutive context

As we have seen in the previous chapters, linear mixtures (instantaneous or convolutive) are separable, that is, the *independence* of the outputs insures the separation of the sources, up to a few indeterminacies [26, 108]. However, the meaning of the *independence* is not the same in convolutive and instantaneous contexts. This can be better stated by the following example.

Example. Let the sources $s_1(n)$ and $s_2(n)$ be iid. Consider now the convolutive mixing-separating system $\mathbf{y}(n) = [\mathbf{B}(z)\mathbf{A}(z)]\mathbf{s}(n)$, where:

$$\mathbf{B}(z)\mathbf{A}(z) = \begin{bmatrix} 1 & z^{-1} \\ 0 & 1 \end{bmatrix} \quad (4.2)$$

hence:

$$\begin{cases} y_1(n) = s_1(n) + s_2(n-1) \\ y_2(n) = s_2(n) \end{cases} \quad (4.3)$$

It can be seen that $y_1(n)$ and $y_2(n)$ are independent for all n , but the sources have not been separated.

The problem comes from the fact that in the convolutive context, $y_1(n)$ and $y_2(n)$ have to be independent in the sense of stochastic processes¹ [77], which requires the independence of the random variables $y_1(n)$ and $y_2(n-m)$ for all n and all m^2 . Therefore, although $I(y_1(n), y_2(n))$, where $I(\cdot, \cdot)$ denotes the mutual information, is a good independence criterion

¹Two stochastic processes $y_1(t)$ and $y_2(t)$ are independent if and only if for any set $t_1, \dots, t_n, t'_1, \dots, t'_m$ the group $\{y_1(t_1), \dots, y_1(t_n)\}$ is independent from the group $\{y_2(t'_1), \dots, y_2(t'_m)\}$.

²This fact reveals an intrinsic difficulty of convolutive mixtures: in instantaneous mixtures we are dealing with random variables, and each sample of signals is treated as an observation of a random variable. That is, with N data points, we have N observations of some joint random variables, and from them, we would like to estimate the parameters of the mixing system. However, in convolutive mixtures, we are dealing with stochastic processes, and with N data points, we have observed N points of a “sole” observation of a stochastic process.

for instantaneous mixtures, it cannot be applied for convolutive mixtures. Instead, we can use the criterion³:

$$J = \sum_m I(y_1(n), y_2(n-m)) \quad (4.4)$$

The limits of the above summation, depends on the degree of the filters corresponding to the whole mixing-separating system. For Infinite Impulse Response (IIR) filters, the range of m is from $-\infty$ to $+\infty$. If we knew the maximum degree of the whole mixing-separating system, say M , the limits of the above summation could be selected in the range $-M$ and M . But, in practice, we have not this information. However, when we use FIR separating filters, it seems that we do not need to use more independent terms in the criterion than the number of separating parameters, and hence M can be chosen equal to twice the maximum degree of separating filters.

The criterion (4.4) is computationally expensive. However, in gradient based algorithms (which are our approach for source separation), we can use a simple trick: *at each iteration choose randomly a different value for m , and use $I(y_1(n), y_2(n-m))$ as the current separation criterion*. This is, in fact, a stochastic implementation of the criterion (4.4), but with a highly smaller computational load. Also, it can be said intuitively that with this trick, the speed of convergence (in the sense of the number of iterations) will not be highly affected, because most real signals are not iid, and hence the information in the different terms of (4.4) are correlated.

4.4 Mutual Information and gradient based algorithms

Suppose that, in a source separation algorithm, the mutual information of the outputs $I(\mathbf{y})$ has been chosen as the independence criterion. Now if we use the steepest descent gradient algorithm for its minimization, we do not need to estimate $I(\mathbf{y})$ itself, but only its derivative with respect to the parameters of the separating system.

As it is evident from (4.1), the mutual information depends on the multivariate joint PDF $p_{\mathbf{x}}(\mathbf{x})$, and hence it is natural to expect that its derivative with respect to the parameters of the separating system depends on the partial derivatives of $p_{\mathbf{x}}(\mathbf{x})$. Unfortunately, the joint density is not easy to estimate. However, the previously known methods escape from this problem by a simple trick. For example, in linear instantaneous mixtures, the separating matrix \mathbf{B} must be estimated in such a way that the mutual information of $\mathbf{y} = \mathbf{B}\mathbf{x}$ be minimized. The steepest descent algorithm $\mathbf{B} \leftarrow \mathbf{B} - \mu \frac{\partial I(\mathbf{y})}{\partial \mathbf{B}}$, requires the estimation of

³Another approach for using the mutual information based criteria in the convolutive context is considered by D.-T. Pham [78, 80].

$\frac{\partial I(\mathbf{y})}{\partial \mathbf{B}}$. From $\mathbf{y} = \mathbf{B}\mathbf{x}$, we have:

$$p_{\mathbf{y}}(\mathbf{y}) = \frac{p_{\mathbf{x}}(\mathbf{x})}{|\det \mathbf{B}|} \quad (4.5)$$

Taking the logarithm of both sides of the above equation, gives us:

$$\ln p_{\mathbf{y}}(\mathbf{y}) = \ln p_{\mathbf{x}}(\mathbf{x}) - \ln |\det \mathbf{B}| \quad (4.6)$$

The key point here is that in (4.6), the output joint PDF is expressed as the sum of a fixed term (which does not depend on the separating system) and a term which does not depend on any multi-variate joint PDF. This property is a direct result of the fact that the relation (4.5) is *multiplicative*.

Finally, by combining (4.6) with $H(\cdot) = -E \{\ln p_{(\cdot)}(\cdot)\}$ and $I(\mathbf{y}) = \sum_i H(y_i) - H(\mathbf{y})$, we obtain:

$$I(\mathbf{y}) = \sum_i H(y_i) - H(\mathbf{x}) - \ln |\det \mathbf{B}| \quad (4.7)$$

From (4.7), the desired gradient $\frac{\partial I(\mathbf{y})}{\partial \mathbf{B}}$ can be easily computed. Since $H(\mathbf{x})$ is constant with respect to \mathbf{B} , it disappears in the differentiation and hence $\frac{\partial I(\mathbf{y})}{\partial \mathbf{B}}$ only depends on the derivatives of the marginal PDFs. In fact:

$$\frac{\partial}{\partial \mathbf{B}} I(\mathbf{y}) = E \{ \boldsymbol{\psi}_{\mathbf{y}}(\mathbf{y}) \mathbf{x}^T \} - \mathbf{B}^{-T} \quad (4.8)$$

where $\boldsymbol{\psi}_{\mathbf{y}}(\mathbf{y}) = (\psi_{y_1}(y_1), \dots, \psi_{y_N}(y_N))^T$ is the vector of score functions⁴ of the components of \mathbf{y} (a review on the score function of a random variable is presented in the Appendix C).

A similar trick is used in PNL mixtures for escaping from the multivariate PDFs. In fact, for these mixtures too, the relation between $p_{\mathbf{e}}(\mathbf{e})$ and $p_{\mathbf{y}}(\mathbf{y})$ is *multiplicative*:

$$p_{\mathbf{y}}(\mathbf{y}) = \frac{p_{\mathbf{e}}(\mathbf{e})}{|(\det \mathbf{B}) \prod_i g'(e_i)|} \quad (4.9)$$

where \mathbf{e} is the observation vector, and g_i 's are the nonlinear compensating functions. Consequently, we can use the logarithm to separate the multivariate part from the separating parameters, and obtain finally the equation:

$$I(\mathbf{y}) = \sum_i H(y_i) - H(\mathbf{e}) - \ln |\det \mathbf{B}| - \sum_i E \{ \ln |g'_i(e_i)| \} \quad (4.10)$$

However, these tricks cannot be applied in the convolutive case, because the relation between the joint PDF's of the observations and the joint PDF's of the outputs is no longer *multiplicative*. For example, in a linear convolutive mixture and using FIR separating filters with the maximum order p , we have:

$$\mathbf{y}(n) = \mathbf{B}_0 \mathbf{x}(n) + \mathbf{B}_1 \mathbf{x}(n-1) + \dots + \mathbf{B}_p \mathbf{x}(n-p) \quad (4.11)$$

⁴Later, we will call it the Marginal Score Function (MSF) of the random *vector* \mathbf{y} .

and hence, the relation between $p_{\mathbf{x}}(\mathbf{x})$ and $p_{\mathbf{y}}(\mathbf{y})$ is not multiplicative. Therefore, if we want to use mutual information as the separation criterion in convolutive mixtures, we are obliged to deal with multi dimensional distributions. However, in this case too, we are mainly interested in estimating the *gradient* of the mutual information, and not the mutual information itself. This gradient will be calculated in the following sections, but first we need to define multivariate score functions.

Another problem when using gradient based methods concerns the possibility of the existence of local minimum. We will prove in the following sections, that the mutual information, has no “local minimum” by itself (see Theorem 4.2, page 42).

4.5 Multi-variate score functions

In this section, we extend the concept of score function of random variables to random vectors. These score functions are required for expressing the “gradient” of the mutual information in the next section. After defining these score functions, we consider some of their properties, which are necessary for better understanding their importance and finding methods for estimating them.

4.5.1 Definitions

First of all, recall the definition of the score function of a random variable:

Definition 4.1 (Score Function) *The score function of a scalar random variable x is the opposite of the log derivative of its density, i.e. :*

$$\psi_x(x) \triangleq -\frac{d}{dx} \ln p_x(x) = -\frac{p'_x(x)}{p_x(x)} \quad (4.12)$$

where p_x denotes the PDF of x .

In conjunction with this definition, we define two different types of score functions for a random vector $\mathbf{x} = (x_1, \dots, x_N)^T$:

Definition 4.2 (MSF) *The Marginal Score Function (MSF) of \mathbf{x} is the vector of score functions of its components. That is:*

$$\boldsymbol{\psi}_{\mathbf{x}}(\mathbf{x}) \triangleq (\psi_1(x_1), \dots, \psi_N(x_N))^T \quad (4.13)$$

where:

$$\psi_i(x_i) \triangleq -\frac{d}{dx_i} \ln p_{x_i}(x_i) = -\frac{p'_{x_i}(x_i)}{p_{x_i}(x_i)} \quad (4.14)$$

Definition 4.3 (JSF) *The Joint Score Function (JSF) of \mathbf{x} is the gradient of “ $-\ln p_{\mathbf{x}}(\mathbf{x})$ ”, that is:*

$$\boldsymbol{\varphi}_{\mathbf{x}}(\mathbf{x}) \triangleq (\varphi_1(\mathbf{x}), \dots, \varphi_N(\mathbf{x}))^T \quad (4.15)$$

where:

$$\varphi_i(\mathbf{x}) \triangleq -\frac{\partial}{\partial x_i} \ln p_{\mathbf{x}}(\mathbf{x}) = -\frac{\frac{\partial}{\partial x_i} p_{\mathbf{x}}(\mathbf{x})}{p_{\mathbf{x}}(\mathbf{x})} \quad (4.16)$$

In the following sections, we will see that the difference between these two score functions contains a lot of information about the independence of the components of a random vector. Thus, it worths enough to give a formal name to this difference:

Definition 4.4 (SFD) *The Score Function Difference (SFD) of \mathbf{x} is the difference between its MSF and JSF, that is:*

$$\boldsymbol{\beta}_{\mathbf{x}}(\mathbf{x}) \triangleq \boldsymbol{\psi}_{\mathbf{x}}(\mathbf{x}) - \boldsymbol{\varphi}_{\mathbf{x}}(\mathbf{x}) \quad (4.17)$$

4.5.2 Properties

The first property presented here points out that SFD contains information about the independence of the components of a random vector:

Property 4.1 *The components of a random vector $\mathbf{x} = (x_1, \dots, x_N)^T$ are independent if and only if $\boldsymbol{\beta}_{\mathbf{x}}(\mathbf{x}) \equiv \mathbf{0}$, that is:*

$$\boldsymbol{\varphi}_{\mathbf{x}}(\mathbf{x}) = \boldsymbol{\psi}_{\mathbf{x}}(\mathbf{x}) \quad (4.18)$$

Proof. Here, for the sake of simplicity, we prove this property only for two dimensional case. Its generalization to higher dimensions is obvious.

If the components of \mathbf{x} are independent, then (4.18) can be easily obtained. Conversely, suppose that (4.18) holds. Then we have $\frac{\partial}{\partial x_1} \ln p_{\mathbf{x}}(x_1, x_2) = \frac{\partial}{\partial x_1} \ln p_{x_1}(x_1)$. Integrating both sides of this equation with respect to x_1 , leads to:

$$\ln p_{\mathbf{x}}(x_1, x_2) = \ln p_{x_1}(x_1) + g(x_2) \Rightarrow p_{\mathbf{x}}(x_1, x_2) = p_{x_1}(x_1)h(x_2) \quad (4.19)$$

By integrating both sides of this equation with respect to x_1 from $-\infty$ to $+\infty$, we have $h(x_2) = p_{x_2}(x_2)$, which proves the property. \blacktriangle

Property 4.2 *For a random vector $\mathbf{x} = (x_1, \dots, x_N)^T$ we have:*

$$\beta_i(\mathbf{x}) = \frac{\partial}{\partial x_i} \ln p(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_N | x_i) \quad (4.20)$$

where $\beta_i(\mathbf{x})$ denotes the i -th component of the SFD of \mathbf{x} .

Proof. Without loss of generality, let $i = 1$. Then we can write:

$$\begin{aligned}
\beta_1(\mathbf{x}) &= \psi_1(x_1) - \varphi_1(\mathbf{x}) \\
&= \frac{\partial}{\partial x_1} \ln p_{\mathbf{x}}(\mathbf{x}) - \frac{\partial}{\partial x_1} \ln p_{x_1}(x_1) \\
&= \frac{\partial}{\partial x_1} \ln \frac{p_{\mathbf{x}}(x_1, \dots, x_N)}{p_{x_1}(x_1)} \\
&= \frac{\partial}{\partial x_1} \ln p_{x_2, \dots, x_N}(x_2, \dots, x_N | x_1)
\end{aligned} \tag{4.21}$$

▲

It is interesting to note the relationship between this property and Property 4.1. For example, for two dimensional case, we can write:

$$\beta_1(x_1, x_2) = \frac{\partial}{\partial x_1} \ln p(x_2 | x_1) \tag{4.22}$$

$$\beta_2(x_1, x_2) = \frac{\partial}{\partial x_2} \ln p(x_1 | x_2) \tag{4.23}$$

In other words, $\beta_1(x_1, x_2) = 0$ if $p(x_2|x_1)$ does not depend on x_1 , *i.e.* when x_1 and x_2 are independent. For the N -dimensional case too, $\beta_i(\mathbf{x}) = 0$ if “ x_i and the other components of \mathbf{x} are independent”, that is, if $p(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_N | x_i) = p(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_N)$, or $p(\mathbf{x}) = p(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_N)p(x_i)$.

The following property is, in fact, a generalization of the similar property for the score function of a scalar random variable (refer to Appendix C, Property C.1).

Property 4.3 *Let \mathbf{x} be a random vector with a density $p_{\mathbf{x}}$ and a JSF $\varphi_{\mathbf{x}}$. Moreover, let $f(\mathbf{x})$ be a multivariate function with continuous partial derivatives and:*

$$\lim_{x_i \rightarrow \pm\infty} \int_{x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_N} f(\mathbf{x}) p_{\mathbf{x}}(\mathbf{x}) dx_1 \cdots dx_{i-1} dx_{i+1} \cdots dx_N = 0 \tag{4.24}$$

Then we have:

$$E \{ f(\mathbf{x}) \varphi_i(\mathbf{x}) \} = E \left\{ \frac{\partial f}{\partial x_i}(\mathbf{x}) \right\} \tag{4.25}$$

Note that the condition (4.24) is not too restrictive for usual sources, because for most physical signals $p_{\mathbf{x}}(\mathbf{x})$ decreases rapidly when $\|\mathbf{x}\|$ goes to infinity. In fact, most real signals are “bounded”, and for them (4.24) holds.

Proof. Without loss of generality, let $i = 1$. We write:

$$E \{ f(\mathbf{x}) \varphi_1(\mathbf{x}) \} = \int f(\mathbf{x}) \varphi_1(\mathbf{x}) p_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} \tag{4.26}$$

$$= - \int_{x_2, \dots, x_N} \int_{x_1} f(\mathbf{x}) \frac{\partial p_{\mathbf{x}}(\mathbf{x})}{\partial x_1} dx_1 dx_2 \cdots dx_N \tag{4.27}$$

Now, by applying integration by parts for the inner integral and using (4.24) the desired relation will be obtained. ▲

Corollary 1 For bounded random vector \mathbf{x} :

$$E \{ \varphi_i(\mathbf{x}) x_j \} = \begin{cases} 1 & ; \text{if } i = j \\ 0 & ; \text{if } i \neq j \end{cases} \quad (4.28)$$

or equivalently:

$$E \{ \boldsymbol{\varphi}_{\mathbf{x}}(\mathbf{x}) \mathbf{x}^T \} = \mathbf{I} \quad (4.29)$$

where \mathbf{I} denotes the identity matrix.

Corollary 2 Suppose we would like to estimate $\varphi_i(\mathbf{x})$ by a parametric function $f(\mathbf{x}; \mathbf{w})$, where $\mathbf{w} = (w_1, \dots, w_K)^T$ is the parameters vector. Then:

$$\underset{\mathbf{w}}{\operatorname{argmin}} E \left\{ (\varphi_i(\mathbf{x}) - f(\mathbf{x}; \mathbf{w}))^2 \right\} = \underset{\mathbf{w}}{\operatorname{argmin}} \left\{ E \{ f^2(\mathbf{x}; \mathbf{w}) \} - 2E \left\{ \frac{\partial f}{\partial x_i}(\mathbf{x}, \mathbf{w}) \right\} \right\} \quad (4.30)$$

This corollary shows a nice property of JSF: even without knowledge about $\varphi_i(\mathbf{x})$, we can design a minimum mean square error (MMSE) estimator for it.

Property 4.4 For a random vector $\mathbf{x} = (x_1, \dots, x_N)^T$ we have:

$$\psi_i(x) = E \{ \varphi_i(\mathbf{x}) \mid x_i = x \} \quad (4.31)$$

where φ_i and ψ_i denote the i -th component of JSF and MSF of \mathbf{x} , respectively.

Proof. Without loss of generality let $i = 1$. Then we write:

$$\begin{aligned} E \{ \varphi_1(\mathbf{x}) \mid x_1 \} &= \int_{x_2, \dots, x_N} \varphi_1(\mathbf{x}) p(x_2, \dots, x_N \mid x_1) dx_2 \cdots dx_N \\ &= - \int_{x_2, \dots, x_N} \frac{\frac{\partial}{\partial x_1} p_{\mathbf{x}}(\mathbf{x})}{p(\mathbf{x})} \cdot \frac{p(\mathbf{x})}{p_{x_1}(x_1)} dx_2 \cdots dx_N \\ &= - \frac{1}{p_{x_1}(x_1)} \cdot \frac{\partial}{\partial x_1} \int_{x_2, \dots, x_N} p(\mathbf{x}) dx_2 \cdots dx_N \\ &= - \frac{1}{p_{x_1}(x_1)} \cdot \frac{\partial}{\partial x_1} p_{x_1}(x_1) \\ &= \psi_1(x_1) \end{aligned} \quad (4.32)$$

which proves the property. ▲

The above property needs more explanation. Consider φ_i as a function⁵ of x_i , denoted by $\varphi_i(x_i)$. If x_i is independent from the other variables, then $\varphi_i(x_i) = \psi_i(x_i)$. However, if the other variables depend on x_i , $\varphi_i(x_i)$ is no longer equal to $\psi_i(x_i)$, but the above property

⁵Strictly speaking, it is a “relation” not a “function”, because for each value of x_i we have several values for φ_i .

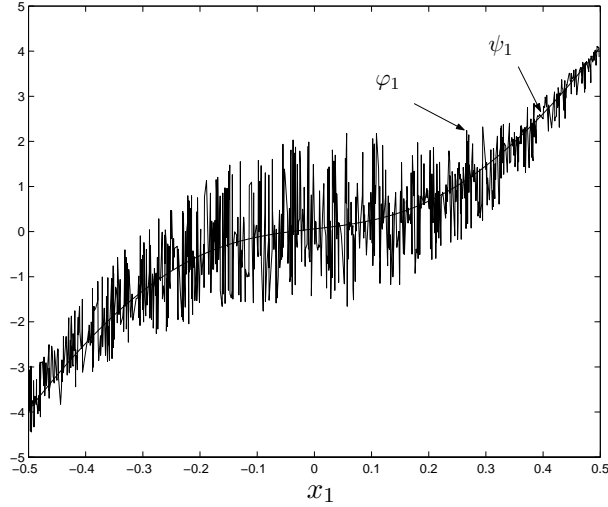


Figure 4.1: φ_1 and ψ_1 versus x_1 in the presence of statistical dependence.

says that its “mean” will be still equal to $\psi_i(x_i)$. In other words, the statistical dependence, can introduce some fluctuations in $\varphi_i(x_i)$, but only around its constant “mean”. We will later use this property for estimating the SFD.

Example. Let s_1 and s_2 be two independent random variables with uniform distributions on the interval $(-0.5, 0.5)$. We define random variables x_1 and x_2 by:

$$\begin{cases} x_1 = s_1 \\ x_2 = s_2 + ks_1 \end{cases} \quad (4.33)$$

For $k = 0$, x_1 and x_2 are independent and we have $\varphi_1(x_1) = \psi_1(x_1)$. Now, if we vary k , $\psi_1(x_1)$ remains unchanged (because it does not depend on k) but $\varphi_1(x_1)$ will be varied. However, Property 4.4 states that its “mean” remains unchanged. Figure 4.1 shows the estimated⁶ $\varphi_1(x)$ and $\psi_1(x)$ for $k = 0.5$.

Therefore, we can conclude that: *The SFD is, in fact, a measure of the variations of JSF (around its smoothed value).*

Property 4.5 Let $\mathbf{y} = \mathbf{B}\mathbf{x}$, where \mathbf{x} and \mathbf{y} are random vectors and \mathbf{B} is a non-singular square matrix. Then:

$$\varphi_{\mathbf{y}}(\mathbf{y}) = \mathbf{B}^{-T} \varphi_{\mathbf{x}}(\mathbf{x}) \quad (4.34)$$

Proof. From $\mathbf{y} = \mathbf{B}\mathbf{x}$ we have:

$$p_{\mathbf{y}}(\mathbf{y}) = \frac{p_{\mathbf{x}}(\mathbf{x})}{|\det \mathbf{B}|} \Rightarrow \ln p_{\mathbf{x}}(\mathbf{x}) = \ln p_{\mathbf{y}}(\mathbf{y}) + \ln |\det \mathbf{B}| \quad (4.35)$$

⁶These curves have been obtained by using *kernel estimators* (see appendix D).

Therefore, for $i = 1, \dots, N$ we can write:

$$\begin{aligned}
\varphi_{\mathbf{x},i}(\mathbf{x}) &= -\frac{\partial}{\partial x_i} \ln p_{\mathbf{x}}(\mathbf{x}) \\
&= -\frac{\partial}{\partial x_i} \ln p_{\mathbf{y}}(\mathbf{y}) \\
&= -\sum_k \frac{\partial}{\partial y_k} \ln p_{\mathbf{y}}(\mathbf{y}) \cdot \frac{\partial y_k}{\partial x_i} \\
&= \sum_k b_{ki} \varphi_{\mathbf{y},k}(\mathbf{y})
\end{aligned} \tag{4.36}$$

where $\varphi_{\mathbf{x},i}(\mathbf{x})$ and $\varphi_{\mathbf{y},i}(\mathbf{y})$ denote the i -th components of the JSFs of \mathbf{x} and \mathbf{y} , respectively. From the above relation, we have $\varphi_{\mathbf{x}}(\mathbf{x}) = \mathbf{B}^T \varphi_{\mathbf{y}}(\mathbf{y})$, which proves the property. \blacktriangle

4.6 Differential of the Mutual Information

We can now state the theorem concerning the variation of the mutual information of a random vector resulting from a small variation in its argument. Since its proof is somewhat long and needs two other lemmas, it is presented in the appendix B.

Theorem 4.1 (Differential of the mutual information) *Let \mathbf{x} be a bounded random vector, and let Δ be a ‘small’ random vector with the same dimension, then:*

$$I(\mathbf{x} + \Delta) - I(\mathbf{x}) = E \{ \Delta^T \beta_{\mathbf{x}}(\mathbf{x}) \} + o(\Delta) \tag{4.37}$$

where $\beta_{\mathbf{x}}$ is the SFD of \mathbf{x} , and $o(\Delta)$ denotes higher order terms in Δ .

Recall that for a multivariate (differentiable) function $f(\mathbf{x})$ we have:

$$f(\mathbf{x} + \Delta) - f(\mathbf{x}) = \Delta^T \cdot (\nabla f(\mathbf{x})) + o(\Delta) \tag{4.38}$$

Comparing this equation with (4.37), we can call SFD the “stochastic gradient” of the mutual information.

Recall now the property 4.1, which states that SFD is zero if and only if $I(\mathbf{x})$ is minimum. For a usual multivariate function $f(\mathbf{x})$, if $\nabla f(\mathbf{x}) = \mathbf{0}$ is equivalent to global minimization of f , we can conclude that the function f has no local minimum. In this case, too, we can state the following theorem, which, in fact, points out that “the mutual information has no local minimum”.

Theorem 4.2 *Let \mathbf{x}_0 be a random vector with a continuously differentiable PDF. If for any “small” random vector Δ , $I(\mathbf{x}_0) \leq I(\mathbf{x}_0 + \Delta)$ holds, then $I(\mathbf{x}_0) = 0$.*

Proof. Suppose that $I(\mathbf{x}_0) \neq 0$. Then, we conclude that the components of \mathbf{x}_0 are not independent, and hence $\beta_{\mathbf{x}_0}(\mathbf{x}_0)$ cannot be zero. Moreover, the continuity of the function $\beta_{\mathbf{x}_0}(\mathbf{x})$ shows that $E \left\{ \|\beta_{\mathbf{x}_0}(\mathbf{x}_0)\|^2 \right\}$ cannot be zero, too. Now, let $\Delta = -\mu\beta_{\mathbf{x}_0}(\mathbf{x}_0)$, where μ is a small positive constant which insures that Δ is a “small” random vector. From Theorem 4.1 we can write (up to first order terms):

$$I(\mathbf{x}_0 + \Delta) - I(\mathbf{x}_0) = -\mu E \left\{ \|\beta_{\mathbf{x}_0}(\mathbf{x}_0)\|^2 \right\} < 0 \quad (4.39)$$

hence $I(\mathbf{x}_0 + \Delta) < I(\mathbf{x}_0)$, which is a contradiction. Therefore, we must have $I(\mathbf{x}_0) = 0$, that is, I is in its global minimum. \blacktriangle



Note: In the above theorem, it is not necessary that the PDF of \mathbf{x}_0 is continuously differentiable on all its support. In fact, it is sufficient that if the function $\beta_{\mathbf{x}_0}(\mathbf{x})$ is not identically zero, we are able to conclude that $E \left\{ \|\beta_{\mathbf{x}_0}(\mathbf{x}_0)\|^2 \right\}$ is not zero, too. This property certainly holds for all “usual” random vectors.



Note: Although the above theorem guaranties that the mutual information has no local minimum, it tells nothing about the local minimum of a parametric separating system with respect to the parameter space. In other words, for the nonlinear separating system $\mathbf{y} = g(\mathbf{x}; \mathbf{w})$, the function $h(\mathbf{w}) = I(\mathbf{y})$ may have some local minima with respect to the parameter vector \mathbf{w} . The same problem exists, for example, in MMSE estimation of any nonlinear model: although the function $(\cdot)^2$ has no local minimum, the function $h(\mathbf{w}) = E \left\{ (\mathbf{y} - g(\mathbf{x}; \mathbf{w}))^2 \right\}$ may contain local minima (as a function of \mathbf{w}). The local minimum, can also be introduced by the estimation method of SFD.

4.7 Estimating multi-variate score functions

In this section, we consider the problem of estimating multi-variate score functions and SFD. Since MSF is nothing but the vector of usual scalar score functions, it can be estimated by one of the known methods of estimating scalar score functions, and need not to be considered here.

4.7.1 Estimating JSF

The methods presented here for estimating JSF are, in fact, the generalizations of the corresponding methods for estimating uni-variate score functions.

Kernel Estimators

We define a multi-variate *kernel* function $k(\mathbf{x}) = k(x_1, \dots, x_N)$ as the PDF of a zero mean random vector. That is, $k(\mathbf{x})$ is a kernel function if and only if: (a) $\forall \mathbf{x} \in \mathbb{R}^N, k(\mathbf{x}) \geq 0$, (b) $\int_{\mathbb{R}^N} k(\mathbf{x}) d\mathbf{x} = 1$ and (c) $\int_{\mathbb{R}^N} \mathbf{x} k(\mathbf{x}) d\mathbf{x} = \mathbf{0}$. The *bandwidth* of the kernel in the direction i is defined as the variance of the i -th component of the corresponding random vector: $h_i \triangleq \int_{\mathbb{R}^N} x_i^2 k(\mathbf{x}) d\mathbf{x}$.

Suppose now that \mathbf{x} is a random vector, from which the samples $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T\}$ have been observed. Then the kernel estimation of the PDF of \mathbf{x} is⁷ [66, 90]:

$$\hat{p}_{\mathbf{x}}(\mathbf{x}) \triangleq \frac{1}{T} \sum_{t=1}^T k(\mathbf{x} - \mathbf{x}_t) \quad (4.40)$$

And from there, the kernel estimation of the i -th component of JSF will be:

$$\hat{\varphi}_i(\mathbf{x}) \triangleq -\frac{\frac{\partial}{\partial x_i} \hat{p}_{\mathbf{x}}(\mathbf{x})}{\hat{p}_{\mathbf{x}}(\mathbf{x})} = -\frac{\sum_{t=1}^T \frac{\partial k}{\partial x_i}(\mathbf{x} - \mathbf{x}_t)}{\sum_{t=1}^T k(\mathbf{x} - \mathbf{x}_t)} \quad (4.41)$$

Remark 1. The bandwidth parameter in the above equations determines the degree of smoothness of the estimator: the larger bandwidth, the smoother estimated PDF. If it is chosen too small, the estimated PDF will be too fluctuating, and if it is chosen too large, the estimated PDF will be a rough shape of the kernel itself. Optimal choice can be done by cross-validation. However, a rule of thumb is done in the appendix D (see equation (D.6)).

Remark 2. S. Achard has proposed [2] (for the scalar case) that the bandwidth used for estimating the derivative of PDF must be 1.4 times greater than the value used for estimating the PDF itself. This is for partly compensating for the differentiation noise, which is accomplished by forcing more smoothing in the estimation of the derivative of the PDF.

Remark 3. The bandwidth of the kernel can be different in each direction, which depends on the spread of data in that direction. However, as suggested by Fukunaga [40, 90] and is explained in the appendix D, after a whitening process on data, we can use the isotropic kernels (kernels with equal bandwidths in all directions). To state this idea more clearly, let $\mathbf{R}_{\mathbf{x}}$ denote the covariance matrix of \mathbf{x} , and \mathbf{T} be its Cholesky decomposition, *i.e.* \mathbf{T} is an upper triangular matrix which satisfies $\mathbf{R}_{\mathbf{x}} = \mathbf{T}\mathbf{T}^T$. If we now define random variable $\mathbf{y} = \mathbf{T}^{-1}\mathbf{x}$, then $\mathbf{R}_{\mathbf{y}} = \mathbf{I}$, that is, the variance of \mathbf{y} is the same in different directions, and moreover the variables y_i are uncorrelated. Hence, it is natural to use isotropic kernels for estimating PDF and JSF of \mathbf{y} , and then we can use the equations (see the property 4.5):

$$\hat{p}_{\mathbf{x}}(\mathbf{x}) = \frac{\hat{p}_{\mathbf{y}}(\mathbf{y})}{|\det \mathbf{T}|} \quad (4.42)$$

⁷Kernel estimators are considered with more details in the appendix D.

$$\hat{\varphi}_{\mathbf{x}}(\mathbf{x}) = \mathbf{T}^{-T} \hat{\varphi}_{\mathbf{y}}(\mathbf{y}) \quad (4.43)$$

for estimating PDF and JSF of \mathbf{x} .

Minimum Mean Square Error (MMSE) estimation

Property 4.3 can be used for designing Minimum Mean Square Error (MMSE) estimates of $\varphi_i(\mathbf{x})$ (see also Corollary 2 of the property). Here we consider a parametric model which is linear with respect to the parameters.

Suppose we would like to estimate $\varphi_i(\mathbf{x})$ as a linear combination of multi-variate functions $\{k_1(\mathbf{x}), \dots, k_L(\mathbf{x})\}$, that is:

$$\hat{\varphi}_i(\mathbf{x}) = \sum_{j=1}^L w_j k_j(\mathbf{x}) = \mathbf{k}^T(\mathbf{x}) \mathbf{w} \quad (4.44)$$

where $\mathbf{k}(\mathbf{x}) \triangleq (k_1(\mathbf{x}), \dots, k_L(\mathbf{x}))^T$, and $\mathbf{w} \triangleq (w_1, \dots, w_L)^T$. \mathbf{w} can be computed by minimizing the error:

$$\mathcal{E} = E \left\{ (\varphi_i(\mathbf{x}) - \hat{\varphi}_i(\mathbf{x}))^2 \right\} \quad (4.45)$$

From the orthogonality principle [77], we have:

$$E \left\{ \mathbf{k}(\mathbf{x}) (\varphi_i(\mathbf{x}) - \hat{\varphi}_i(\mathbf{x})) \right\} = 0 \quad (4.46)$$

which, by using the property 4.3, becomes:

$$E \left\{ \mathbf{k}(\mathbf{x}) \mathbf{k}^T(\mathbf{x}) \right\} \mathbf{w} = E \left\{ \frac{\partial \mathbf{k}}{\partial x_i}(\mathbf{x}) \right\} \quad (4.47)$$

This equation easily determines \mathbf{w} .

4.7.2 Estimating SFD

Since SFD is the gradient of the mutual information, we are mainly interested in estimating SFD, instead of JSF. Therefore, in this section, we consider some methods for SFD estimation.

Independent estimations of JSF and MSF

One method for estimating SFD, is to estimate independently JSF and MSF, and then to compute their difference. That is:

$$\hat{\beta}_{\mathbf{x}}(\mathbf{x}) = \hat{\psi}_{\mathbf{x}}(\mathbf{x}) - \hat{\varphi}_{\mathbf{x}}(\mathbf{x}) \quad (4.48)$$

We may apply kernel or MMSE estimation of the joint and marginal score functions. In the latter case, it must be noted that although the estimations of JSF and MSF are both optimal (in the MMSE sense), the estimation of SFD is not. In other words, the independent estimations of $\varphi_i(\mathbf{x})$ and $\psi_i(x_i)$ which minimize $E\{(\varphi_i(\mathbf{x}) - \hat{\varphi}_i(\mathbf{x}))^2\}$ and $E\{(\psi_i(x_i) - \hat{\psi}_i(x_i))^2\}$, respectively, do not necessarily minimize $E\{(\beta_i(\mathbf{x}) - \hat{\beta}_i(\mathbf{x}))^2\}$, where $\beta_i(\mathbf{x}) \triangleq \psi_i(x_i) - \varphi_i(\mathbf{x})$.

Example. (Polynomial estimation of SFD) We have applied successfully the following estimator for separating two linear mixtures (instantaneous or convolutive) of two sources. We estimate $\varphi_i(x_1, x_2)$ by the polynomial:

$$\hat{\varphi}_i(x_1, x_2) = \sum_{j=1}^7 w_{ij} k_j(x_1, x_2) \quad (4.49)$$

where:

$$\begin{aligned} k_1(x_1, x_2) &= 1, & k_2(x_1, x_2) &= x_1, & k_3(x_1, x_2) &= x_1^2, & k_4(x_1, x_2) &= x_1^3 \\ k_5(x_1, x_2) &= x_2, & k_6(x_1, x_2) &= x_2^2, & k_7(x_1, x_2) &= x_2^3 \end{aligned} \quad (4.50)$$

Consequently:

$$\mathbf{k}(x_1, x_2) = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & x_2 & x_2^2 & x_2^3 \end{bmatrix}^T \quad (4.51)$$

$$\frac{\partial}{\partial x_1} \mathbf{k}(x_1, x_2) = \begin{bmatrix} 0 & 1 & 2x_1 & 3x_1^2 & 0 & 0 & 0 \end{bmatrix}^T \quad (4.52)$$

$$\frac{\partial}{\partial x_2} \mathbf{k}(x_1, x_2) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 2x_2 & 3x_2^2 \end{bmatrix}^T \quad (4.53)$$

Then from (4.47) we have:

$$E \{ \mathbf{k}(x_1, x_2) \mathbf{k}(x_1, x_2)^T \} \mathbf{w}_1 = E \left\{ \frac{\partial}{\partial x_1} \mathbf{k}(x_1, x_2) \right\} \quad (4.54)$$

$$E \{ \mathbf{k}(x_1, x_2) \mathbf{k}(x_1, x_2)^T \} \mathbf{w}_2 = E \left\{ \frac{\partial}{\partial x_2} \mathbf{k}(x_1, x_2) \right\} \quad (4.55)$$

where $\mathbf{w}_1 \triangleq (w_{11}, w_{12}, \dots, w_{17})^T$ and $\mathbf{w}_2 \triangleq (w_{21}, w_{22}, \dots, w_{27})^T$. These equations determine \mathbf{w}_1 and \mathbf{w}_2 , which determine $\hat{\varphi}_1(x_1, x_2)$ and $\hat{\varphi}_2(x_1, x_2)$ from (4.49).

Similarly, $\psi_i(x_i)$ is estimated by:

$$\hat{\psi}_i(x_i) = w_1^{(i)} + w_2^{(i)} x_i + w_3^{(i)} x_i^2 + w_4^{(i)} x_i^3 \quad (4.56)$$

and the optimal values of the coefficients are obtained from:

$$E \left\{ \begin{bmatrix} 1 & x_i & x_i^2 & x_i^3 \end{bmatrix}^T \begin{bmatrix} 1 & x_i & x_i^2 & x_i^3 \end{bmatrix} \right\} \mathbf{w}^{(i)} = E \left\{ \begin{bmatrix} 0 & 1 & 2x_i & 3x_i^2 \end{bmatrix}^T \right\} \quad (4.57)$$

where $\mathbf{w}^{(i)} \triangleq (w_1^{(i)}, \dots, w_4^{(i)})^T$ for $i = 1, 2$.

Finally, the SFD is estimated by:

$$\hat{\beta}_1(x_1, x_2) = \hat{\psi}_1(x_1) - \hat{\varphi}_1(x_1, x_2) \quad (4.58)$$

$$\hat{\beta}_2(x_1, x_2) = \hat{\psi}_2(x_2) - \hat{\varphi}_2(x_1, x_2) \quad (4.59)$$

Smoothing JSF

The difficulty of the previous method, is that the estimation errors of JSF and MSF are independent. Recall that a gradient based algorithm for minimizing the mutual information stops when SFD (the difference between MSF and JSF) vanishes. But when the MSF and JSF estimation errors are independent, SFD does not vanish exactly for independent variables. In linear mixtures, the limited degree of freedom of the separating system overcomes this problem and the above estimator works well. On the contrary, nonlinear mixtures require more accurate estimation of the SFD.

Then we suggest to estimate MSF from the estimated JSF. From the property 4.4 we know that MSF is the smoothed version of JSF. Therefore, we can estimate MSF as the smoothed version of the estimated JSF. With this trick, the estimation errors in JSF and MSF are no longer independent, and they partially cancel each other when calculating SFD. In other words, in this method, we estimate SFD as the variations of JSF around its mean (see Fig. 4.1), and hence the separation algorithm tries to minimize these variations.

Practically, following the property 4.4, $\psi_i(x_i)$ is a regression from x_i to $\varphi_i(\mathbf{x})$, that we can compute for instance using smoothing splines (see Appendix A). The final estimation procedure can be summarized in the following steps:

1. From the observed values $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T\}$ estimate $\hat{\varphi}_{i,t} = \hat{\varphi}_i(\mathbf{x}_t), t = 1, \dots, T$ (e.g. by kernel estimators).
2. Compute the smoothing spline (or another regressor) which fits on the data $(x_{i,t}, \hat{\varphi}_{i,t}), t = 1, \dots, T$. This spline will be the estimated MSF $\hat{\psi}_i(x_i)$.
3. Estimate SFD by $\hat{\beta}_{\mathbf{x}}(\mathbf{x}) = \hat{\psi}_{\mathbf{x}}(\mathbf{x}) - \hat{\varphi}_{\mathbf{x}}(\mathbf{x})$.

Histogram estimation method

Another method for estimating SFD (in two dimensional case) is based on a simple histogram estimation of the PDF of \mathbf{x} . The histogram is not a very accurate estimator for PDF, but since we estimate SFD directly, we do not need a very good estimation of PDF (for more details, see Section 4.9.1). In fact, despite of its simplicity, this estimator works very well for instantaneous linear mixtures.

In this method, we first use a histogram for estimating the joint PDF of \mathbf{x} . For two-dimensional vectors, let $N(n_1, n_2)$ denote the number of observations in the bin (n_1, n_2) , then the histogram estimation of $p_{\mathbf{x}}$ is:

$$p(n_1, n_2) = \frac{N(n_1, n_2)}{T} \quad (4.60)$$

where T is the number of observations. Then, we obtain a histogram estimation of p_{x_1} :

$$p_1(n_1) = \sum_{n_2} p(n_1, n_2) \quad (4.61)$$


And from there, we will have an estimation of $p(x_2|x_1)$:

$$p(n_2|n_1) = \frac{p(n_1, n_2)}{p_1(n_1)} = \frac{N(n_1, n_2)}{N(n_1)} \quad (4.62)$$

Finally, having in mind (4.22), we obtain a histogram estimation of $\beta_1(x_1, x_2)$:

$$\beta_1(n_1, n_2) = \frac{p(n_2|n_1) - p(n_2|n_1 - 1)}{p(n_2|n_1)} \quad (4.63)$$

$\beta_2(n_1, n_2)$ will be estimated in a similar manner. Note that the value $\beta(n_1, n_2)$ will be assigned to all the points of the bin (n_1, n_2) .

 **Note:** $p(n_2|n_1)$ is not defined in the bins where $p_1(n_1) = 0$, but this is not a problem, because there is no point in these bins, and then we set $p(n_2|n_1) = 0$. However, in the left most bins (the bins with smallest n_1), for computing $\beta_1(n_1, n_2)$ from (4.63), we need the value of $p(n_2|n_1 - 1)$ which is not defined. In our simulations, we have used 0 for these values, too.

Pham's method

Recently D. T. Pham has proposed [81] a method for estimating the “conditional score function”, which appears in separating temporally correlated sources. The conditional score function of the random vector $\mathbf{x} = (x_1, \dots, x_N)^T$ is defined by:

$$\psi_{x_N|x_{N-1}\dots x_1}(x_N|x_{N-1}, \dots, x_1) \triangleq -\nabla \ln p_{x_N|x_{N-1}\dots x_1}(x_N|x_{N-1}, \dots, x_1) \quad (4.64)$$

where $p_{x_N|x_{N-1}\dots x_1}(x_N|x_{N-1}, \dots, x_1)$ is the conditional density of x_N given x_1, \dots, x_{N-1} are known. From the property 4.2, we know that this is closely related to the SFD of \mathbf{x} .

The Pham's method starts with a whitening stage for obtaining non correlated random variables. This is like what is mentioned in the remark 3 of the kernel estimation method of JSF. He computes also the influence of the whitening on the estimation of the score functions. This influence will be later compensated using an additive term. Afterwards, the joint entropies of whitened data are estimated using a discrete Riemann sum and third order cardinal spline kernels. This discrete Riemann sum, along with the chosen kernel, results in a fast algorithm, specially when the number of data points is relatively high. The conditional entropies, defined as (\mathbf{y} is the whitened data obtained from \mathbf{x}):

$$H(y_N|y_{N-1}, \dots, y_1) \triangleq -E[\log p_{y_N|y_{N-1}\dots y_1}(y_N|y_{N-1}, \dots, y_1)] \quad (4.65)$$

are computed by estimating the joint entropies:

$$H(y_N|y_{N-1}, \dots, y_1) = H(y_N, y_{N-1}, \dots, y_1) - H(y_{N-1}, \dots, y_1) \quad (4.66)$$

The estimator $\hat{H}(y_N|y_{N-1}, \dots, y_1)$ is a function of the observations $\mathbf{y}(1), \dots, \mathbf{y}(T)$, where T stands for the length of the data block. In other words, $\hat{H}(y_N|y_{N-1}, \dots, y_1)$ is stated by an expression depending on $y_1(1), y_1(2), \dots, y_1(T), y_2(1), \dots, y_2(T), \dots, y_N(1), y_N(2), \dots, y_N(T)$. Finally, the l -th component of the conditional score function at a sample point $\mathbf{y}(n)$ is computed as:

$$\hat{\psi}_{y_N|y_{N-1}\dots y_1}^{(l)}(y_N|y_{N-1}, \dots, y_1) \Big|_{\mathbf{y}=\mathbf{y}(t)} = T \frac{\partial \hat{H}(y_N|y_{N-1}, \dots, y_1)}{\partial y_l(t)} \quad (4.67)$$

The idea behind the above formula comes from the fact that if $\tilde{\mathbf{y}} = \mathbf{y} + \boldsymbol{\delta}$, where $\boldsymbol{\delta}$ is a “small” random vector, then:

$$H(\tilde{y}_N|\tilde{y}_{N-1}, \dots, \tilde{y}_1) - H(y_N|y_{N-1}, \dots, y_1) = E \left\{ \boldsymbol{\delta}^T \cdot \boldsymbol{\psi}_{y_N|y_{N-1}\dots y_1}(y_N|y_{N-1}, \dots, y_1) \right\} \quad (4.68)$$

and with (4.67), the above relation holds if the expectation operation (E) is replaced by the empirical averaging. Note also that the conditional score function is estimated only in the points $\mathbf{y}(t)$, but this is sufficient in our application.

Pham also obtains an expression for an optimal choice of the kernel bandwidths. Finally, the conditional score function of \mathbf{x} is calculated from the conditional score functions of \mathbf{y} .

4.8 Mutual Information minimization

In this section, we will show how the main theorem of this chapter (Theorem 4.1) can be used for Independent Component Analysis and source separation.

4.8.1 General nonlinear mixtures

From the proof of Theorem 4.2, we conclude that the algorithm:

$$\mathbf{y} \leftarrow \mathbf{y} - \mu \boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y}) \quad (4.69)$$

will decrease $I(\mathbf{y})$ at each iteration and that the algorithm will stop only when all the components of \mathbf{y} become independent. This gives us the algorithm of Fig. 4.2 for smoothly transforming any “observation” vector to “independent components”.

Note that the normalization of the output energies is required for canceling the scale indeterminacy and specially for avoiding that the algorithm converges to $\mathbf{y} = \mathbf{0}$. The second

- Initialization: $\mathbf{y} = \mathbf{x}$.
 - Loop:
 1. $\mathbf{y} \leftarrow \mathbf{y} - \mu\beta_{\mathbf{y}}(\mathbf{y})$.
 2. For $i = 1, \dots, N$, replace y_i with its smoothed version (obtained from transformation $\mathbf{x} \rightarrow y_i$).
 3. Remove the DC of each output, and normalize its energy to 1.
 - Repeat until convergence.

Figure 4.2: General nonlinear and non-parametric algorithm for Independent Component Analysis.

step (smoothing) is needed because physical systems are usually smooth⁸, and hence the final transformation ($\mathbf{x} \rightarrow \mathbf{y}$) must be smooth. In other words, if two samples of $\mathbf{x}(n)$ are close to each other, their corresponding \mathbf{y} 's must be close, too. This also partially compensates for the measurement errors (noise). The smoothing can be done, for example, by using multi-dimensional smoothing splines (see Appendix A), that is, we first calculate the N -dimensional smoothing spline $\text{sp}(\mathbf{x})$ for the regression from \mathbf{x} to y_i , and then replace y_i with $\text{sp}(\mathbf{x})$. Another approach for achieving this smoothing will be considered in Section 4.8.3.



Note: As it has been mentioned in Chapter 3, in general nonlinear mixtures, ICA and BSS are not equivalent, that is, the independence does not insure the separation in absence of some structural constraints. Therefore, although the above algorithm is an ICA algorithm, without structural constraints, it cannot be considered as a source separation algorithm.



Experiment 4.1. In this experiment, we use two sources with uniform distributions on the interval $(-\sqrt{3}/2, +\sqrt{3}/2)$. The mixing system is:

$$\begin{cases} x_1 = s_1 + 0.5 s_2 + 0.5 s_1 s_2 \\ x_2 = 0.6 s_1 + s_2 + 0.3 s_1 s_2 \end{cases} \quad (4.70)$$

We have used a block of 500 observations, $\mu = 0.1$, two-dimensional smoothing splines for the smoothing process, and the Pham's estimator for estimating SFD. The smoothing parameter

⁸Note that this is not in contradiction with the result of insufficiency of smooth mappings for source separation. In fact, here we are giving an algorithm that only finds a smooth mapping which transforms the observation to independent outputs: it does not necessarily separate the sources. Later, in section 4.8.3, we replace the smoothing step of this algorithm by structural constraints to obtain the projection approach for source separation.

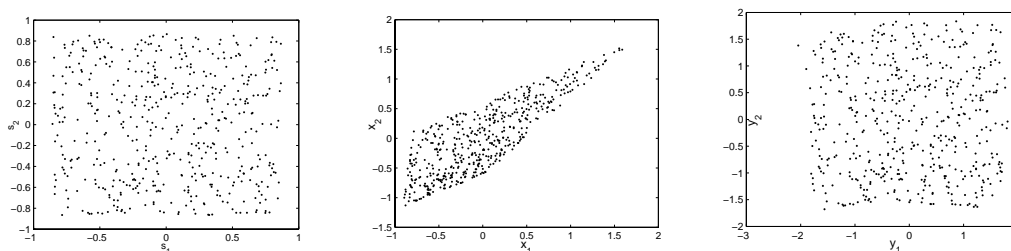
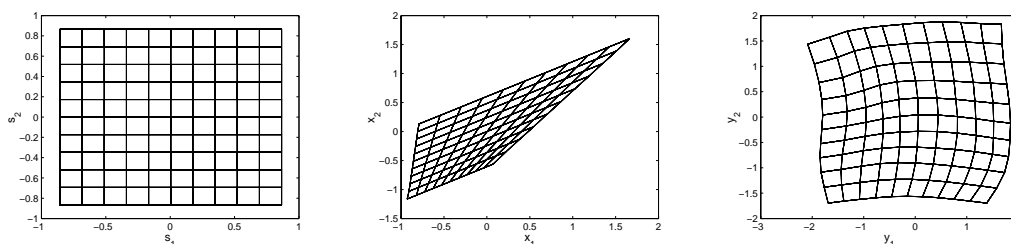
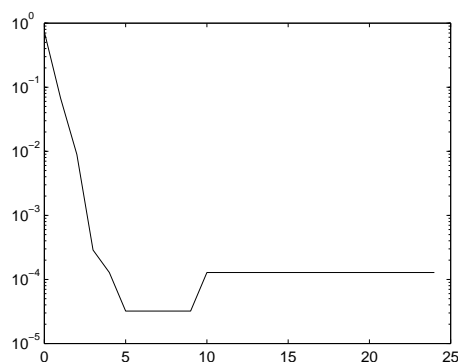


Figure 4.3: The distribution of the sources, observations and outputs (experiment 4.1).

Figure 4.4: The transformations from s -plane to x and y planes (experiment 4.1).

for the multi-dimensional smoothing splines is $\lambda = 0.001$ as defined in equation (A.8) of appendix A. Figure 4.3 shows the joint distributions of sources, observations and the outputs after 25 iterations. Also, Fig. 4.4 shows the mappings from s -plane to x -plane and y -plane. From the distribution plots, it can be seen that the outputs are essentially independent. Moreover, as it is evident from the transformation plot, in this experiment the outputs are approximately separated, too! But this cannot be generalized to any cases.

Figure 4.5 shows the logarithmic plot of the estimated $I(\mathbf{y})$ with respect to the number of iterations. For estimating I the algorithm of [30] has been used. It can be seen that in this method, the convergence achieved after a few iterations.

Figure 4.5: $I(\mathbf{y})$ versus iteration (experiment 4.1).

4.8.2 Gradient approach

Theorem 4.1 can be used for computing the gradient of $I(\mathbf{y})$ with respect to the elements of the separating system. These elements can be scalars, matrices or functions (as in PNL mixtures). Then, this gradient can be used to apply a steepest descent gradient algorithm on the elements, and separating the sources. Here, as an example, we state the details of this calculation for the linear case.

For linear instantaneous mixtures, we have $\mathbf{y} = \mathbf{B}\mathbf{x}$, and minimizing $I(\mathbf{y})$ requires its gradient with respect to \mathbf{B} . Let $\hat{\mathbf{B}} = \mathbf{B} + \mathcal{E}$, where \mathcal{E} is a “small” matrix. Then, we have $\hat{\mathbf{y}} \triangleq \hat{\mathbf{B}}\mathbf{x} = \mathbf{y} + \mathcal{E}\mathbf{x}$. Hence, from Theorem 4.1, and up to first order terms, we can write:

$$I(\hat{\mathbf{y}}) - I(\mathbf{y}) = E \{ \beta_{\mathbf{y}}(\mathbf{y})^T \mathcal{E} \mathbf{x} \} = \langle \mathcal{E}, E \{ \beta_{\mathbf{y}}(\mathbf{y}) \mathbf{x}^T \} \rangle \quad (4.71)$$

where $\langle \cdot, \cdot \rangle$ stands for the scalar product of two matrices (see appendix E, lemma E.1). From the above equation, we conclude:

$$\boxed{\frac{\partial}{\partial \mathbf{B}} I(\mathbf{y}) = E \{ \beta_{\mathbf{y}}(\mathbf{y}) \mathbf{x}^T \}} \quad (4.72)$$

Finally, for estimating \mathbf{B} , we apply the steepest descent algorithm:

$$\mathbf{B} \leftarrow \mathbf{B} - \mu E \{ \beta_{\mathbf{y}}(\mathbf{y}) \mathbf{x}^T \} \quad (4.73)$$

Note that the development of (4.72) does not rely on the multiplicativity of (4.5)⁹. Therefore, we can use exactly the same method for calculating this gradient for the convolutive separating system in (4.11). For doing it, let $\hat{\mathbf{B}}_k = \mathbf{B}_k + \mathcal{E}$. Then $\hat{\mathbf{y}}(n) = \mathbf{y}(n) + \mathcal{E}\mathbf{x}(n-k)$, and hence:

$$I(\hat{\mathbf{y}}(n)) - I(\mathbf{y}(n)) = E \{ \beta_{\mathbf{y}}(\mathbf{y}(n))^T \mathcal{E} \mathbf{x}(n-k) \} = \langle \mathcal{E}, E \{ \beta_{\mathbf{y}}(\mathbf{y}(n)) \mathbf{x}(n-k)^T \} \rangle \quad (4.74)$$

Finally¹⁰:

$$\frac{\partial}{\partial \mathbf{B}_k} I(\mathbf{y}(n)) = E \{ \beta_{\mathbf{y}}(\mathbf{y}(n)) \mathbf{x}(n-k)^T \} \quad (4.75)$$



Note: This technique is, in fact, highly general. For calculating the gradient of the mutual information of the outputs (or their shifted versions) with respect to some parameter of the separating system (which can be a scalar, a matrix, or even a function —as in PNL mixtures), first assume a small variation in this parameter, then calculate its effect on the output, and finally apply the theorem 4.1.

⁹although this multiplicativity can be used to prove the equivalence of (4.72) and (4.8).

¹⁰Remember from Section 4.3 that in convolutive mixtures, minimizing $I(\mathbf{y}(n))$ does not sufficient for separating the sources.

A simple experiment with this method (for separating linear instantaneous mixtures) is presented in Section 4.9.

4.8.3 Projection approach

The idea of this method comes from the idea of general nonlinear ICA algorithm of Section 4.8.1 and equation (4.69). The problem of this method is that it has no structural constraint, that is, although it will generate independent components, the resulting transformations do not necessarily belong to the desired class. One trick to overcome this problem is to project the resulting transformation onto the desired class at each iteration and after modifying \mathbf{y} .

For more explanations, suppose that we are looking for a parametric mapping¹¹ $\mathbf{y} = g(\mathbf{x}; \boldsymbol{\theta})$. After each modification of \mathbf{y} (using (4.69)), we determine the parameters which minimize the difference between \mathbf{y} and $g(\mathbf{x}; \boldsymbol{\theta})$ (e.g. in minimum mean square error sense $E\{\|\mathbf{y} - g(\mathbf{x}; \boldsymbol{\theta})\|^2\}$) and then replace \mathbf{y} by $g(\mathbf{x}; \boldsymbol{\theta})$. Then, this technique minimizes $I(\mathbf{y})$ with the constraint that the mapping belongs to the desired class.

This technique can be seen as another approach for the smoothing step in the algorithm of Fig. 4.2 (step 2). However, in this case, the order of steps 2 and 3 of that algorithm must be reversed, otherwise the output energies does not depend on the values of the parameters, and hence the algorithm of parameter estimation will not converge.

4.9 Application to linear instantaneous mixtures

In this section, we show how our approaches can be applied for separating linear instantaneous mixtures.

4.9.1 Gradient approach

For linear instantaneous mixtures, the separating system is $\mathbf{y} = \mathbf{B}\mathbf{x}$, and we already have calculated the gradient of $I(\mathbf{y})$ with respect to \mathbf{B} (see equation (4.72)). However, in linear instantaneous mixtures, instead of a simple gradient algorithm, it is better to use the equivariant algorithm [21, 6], for obtaining a performance independent of the mixture hardness. In equivariant algorithms, instead of $\frac{\partial I}{\partial \mathbf{B}}$, we use the natural (or relative) gradient:

$$\nabla_{\mathbf{B}} I \triangleq \frac{\partial I}{\partial \mathbf{B}} \mathbf{B}^T = E \{ \boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y}) \mathbf{y}^T \} \quad (4.76)$$

¹¹This equation must not be seen as an instantaneous relation to include convolutive mixtures, too. Even (as in PNL mixtures) some parameters can be functions, and not scalars. Hence this notation is only for stating the idea of the method.


- Initialization: $\mathbf{B} = \mathbf{I}$.
- Loop:
 1. $\mathbf{y} = \mathbf{B}\mathbf{x}$.
 2. Estimate $\beta_{\mathbf{y}}(\mathbf{y})$ (e.g. by histogram method).
 3. $\nabla_{\mathbf{B}}I = E\{\beta_{\mathbf{y}}(\mathbf{y})\mathbf{y}^T\}$
 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 4.6: Separation algorithm for linear instantaneous mixtures

and the updating algorithm for \mathbf{B} is:

$$\mathbf{B} \leftarrow (\mathbf{I} - \mu\nabla_{\mathbf{B}}I)\mathbf{B} \quad (4.77)$$

where \mathbf{I} denotes the identity matrix. Figure 4.6 shows the final algorithm for separating linear instantaneous mixtures.

 **Experiment 4.2.** In this experiment, we use two uniform random sources with zero means and unit variances. The mixing matrix is:

$$\mathbf{A} = \begin{bmatrix} 1 & 0.7 \\ 0.5 & 1 \end{bmatrix} \quad (4.78)$$

and the parameters of the separating algorithm are: (a) Histogram estimation of SFD, by using a 10×10 histogram, (b) 500 point data block, and (c) $\mu = 0.1$. The experiment has been repeated 100 times (for 100 different realization of the sources). For measuring the quality of separation, we use output Signal to Noise Ratio (SNR), defined by (assuming there is no permutation):

$$\text{SNR}_i = 10 \log_{10} \frac{E\{s_i^2\}}{E\{(s_i - y_i)^2\}} \quad (4.79)$$

Figure 4.7 shows the averaged SNR's over 100 runs of the algorithm, versus iterations.

Run time: For giving an idea about the run-time of the experiments in separating different mixing systems considered in the thesis, we give the time required for 100 iterations of each algorithm on our computer, which is a 1GHz Pentium III with 256MB of memory and the Mandrake Linux 8.2 operating system. Our programs are written by a mixture of MATLAB (version 6.1) and C codes, and are not optimized for the speed (the C codes are mainly for kernel estimation

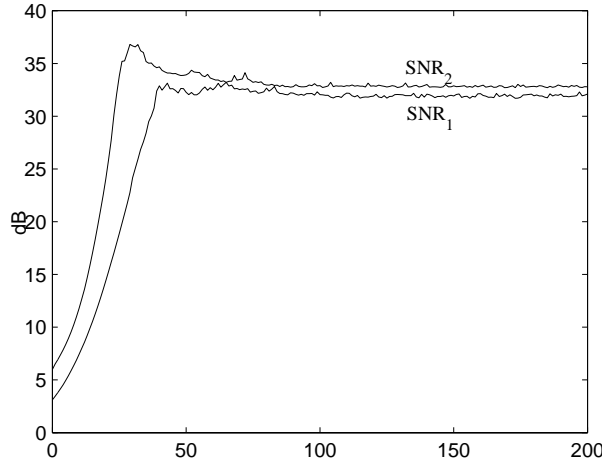


Figure 4.7: Averaged output SNR's versus iteration, in separating linear instantaneous mixtures by using gradient method and histogram estimation of SFD.

of SFD). Consequently, these times are not suitable for an exact comparison between the computational cost of the algorithms. However, they seem to be a useful information for giving an idea about the runtime of the algorithms.

In this experiment, the required time for 100 iterations of the algorithm for each realization of the sources is approximately 3.73 seconds.

As it can be seen in the above experiment, despite of the simplicity of the SFD estimator (a 10×10 histogram), a very good separation performance has been obtained. This fact can be explained as follows. First note that we can rewrite (4.76) as:

$$\begin{aligned}
 \nabla_{\mathbf{B}} I &= E \{ \boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y}) \mathbf{y}^T \} \\
 &= E \{ \boldsymbol{\psi}_{\mathbf{y}}(\mathbf{y}) \mathbf{y}^T \} - E \{ \boldsymbol{\varphi}_{\mathbf{y}}(\mathbf{y}) \mathbf{y}^T \} \\
 &= E \{ \boldsymbol{\psi}_{\mathbf{y}}(\mathbf{y}) \mathbf{y}^T \} - \mathbf{I}
 \end{aligned} \tag{4.80}$$

Which is the equation used in the previous methods. Practically, this equation is simpler than (4.76), since it only requires estimation of marginal score functions. However, the separation information is contained in the averaged SFD, as the gradient of the mutual information, and the separating algorithms will stop when it becomes zero. Moreover, SFD is the difference of two terms. In (4.80), one of these terms is theoretically computed. Hence, a good estimation of the other term is required for separating the sources, since the difference of the two terms must vanish for achieving the source separation. But in our method which is based on the direct use of the SFD, since we estimate it directly, a good estimation of \mathbf{B} can be achieved even with a coarse estimation of the SFD (*e.g.* a simple histogram).

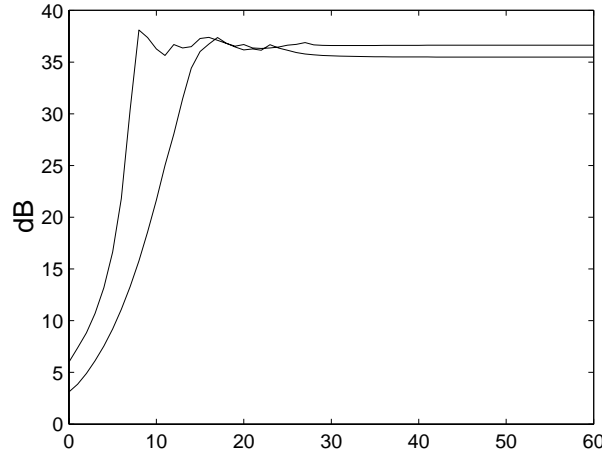



Figure 4.8: Averaged output SNR's versus iteration, in separating linear instantaneous mixtures by using gradient method and polynomial estimation of SFD.

 **Experiment 4.3.** We repeat the previous experiment with the polynomial estimator of SFD. All the parameters are the same as the previous experiment but the SFD estimation method. Figure 4.8 shows the averaged outputs SNR's versus iteration. It can be seen that we have obtained better results compared to histogram estimators (better separation with fewer iterations).

Run time: In this experiment, the required time for 100 iterations of the algorithm for each realization of the sources is approximately 0.96 seconds¹².

4.9.2 Projection approach

For applying the method of Section 4.8.3 on linear instantaneous mixtures, we must do the following tasks at each iteration: first modifying \mathbf{y} using (4.69), then calculating the matrix \mathbf{B} which minimizes $E \left\{ \|\mathbf{y} - \mathbf{B}\mathbf{x}\|^2 \right\}$, and finally replacing \mathbf{y} by $\mathbf{B}\mathbf{x}$. Hence, we need the following lemma:

Lemma 4.1 *The matrix \mathbf{B} which minimizes $E \left\{ \|\mathbf{y} - \mathbf{B}\mathbf{x}\|^2 \right\}$ is:*

$$\mathbf{B}_{opt} = E \left\{ \mathbf{y}\mathbf{x}^T \right\} \left(E \left\{ \mathbf{x}\mathbf{x}^T \right\} \right)^{-1} \quad (4.81)$$

¹²We emphasize again that these run times are only for giving an idea to the reader. The programs are not optimized for speed. Here it can be seen that the time needed for a polynomial estimation of SFD is less than its histogram estimation, although the histogram estimation has less computational cost. This comes from our MATLAB implementation, which is slow for the loops required for the histogram estimator, but is fast for the matrix manipulations required for the polynomial estimator.

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

Figure 4.9: Projection algorithm for separating linear instantaneous mixtures.

Proof. We write:


$$\begin{aligned}
 \mathcal{C} &\triangleq E\{\|\mathbf{y} - \mathbf{B}\mathbf{x}\|^2\} \\
 &= E\{(\mathbf{y} - \mathbf{B}\mathbf{x})^T(\mathbf{y} - \mathbf{B}\mathbf{x})\} \\
 &= E\{\mathbf{y}^T\mathbf{y}\} - E\{\mathbf{y}^T\mathbf{B}\mathbf{x}\} - E\{\mathbf{x}^T\mathbf{B}^T\mathbf{y}\} + E\{\mathbf{x}^T\mathbf{B}^T\mathbf{B}\mathbf{x}\} \\
 &= E\{\mathbf{y}^T\mathbf{y}\} - 2E\{\mathbf{y}^T\mathbf{B}\mathbf{x}\} + E\{\|\mathbf{B}\mathbf{x}\|^2\}
 \end{aligned} \tag{4.82}$$

Then, from the lemmas E.3 and E.4 (see Appendix E) we have:

$$\frac{\partial \mathcal{C}}{\partial \mathbf{B}} = -2E\{\mathbf{y}\mathbf{x}^T\} + 2\mathbf{B}E\{\mathbf{x}\mathbf{x}^T\} \tag{4.83}$$

Finally, setting $\partial \mathcal{C} / \partial \mathbf{B} = 0$ proves the lemma. ▲

From the above lemma, the projection algorithm for separating linear instantaneous mixtures will be obtained as shown in Fig. 4.9.

 **Experiment 4.4.** We repeat the experiment 4.3 using the projection approach. In this experiment, the sources are two uniform random signals with zero means and unit variances. The mixing matrix is:

$$\mathbf{A} = \begin{bmatrix} 1 & 0.7 \\ 0.5 & 1 \end{bmatrix} \tag{4.84}$$

and the parameters of the separating algorithm are: (a) Polynomial estimation of SFD, (b) 500 point data block, and (c) $\mu = 0.1$. Figure 4.10 shows the averaged SNR's taken over 100 runs of the algorithm versus iteration. It clearly shows the ability of the algorithm for separating the sources. Moreover, it can be seen that it converges faster than the gradient approach.

Run time: The time required for running 100 iterations of the experiment for each realization of the sources is about 1.62 seconds (note that we do not actually need to run 100 iterations).

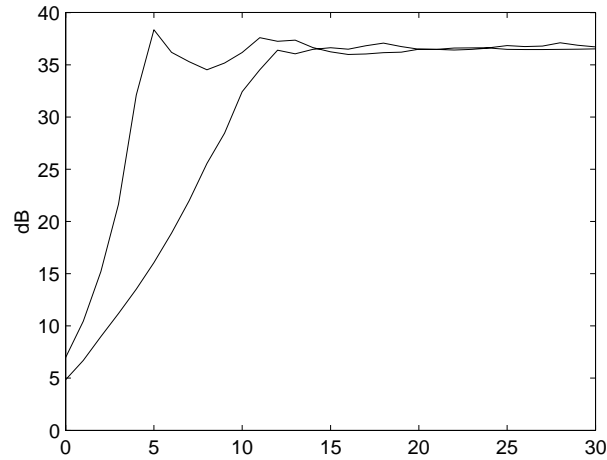


Figure 4.10: Averaged output SNR's versus iteration, in separating linear instantaneous mixtures by using the projection approach.

4.10 conclusion

In this chapter, we considered the mutual information more deeply, and we found its “differential”, which can be used for its minimization. We also showed that SFD is the stochastic “gradient” of the mutual information and that the mutual information has no local minimum. Then, we have considered a few methods for estimating SFD from data. After that, we proposed two general approaches for using SFD in source separation: (1) Using it for calculating the gradient of $I(\mathbf{y})$ with respect to the elements of the separating system, and (2) Projection methods. Finally, we designed practical algorithms for separating linear instantaneous mixtures and proposed some experiments. In the following chapters, we apply the gradient and projections approaches for separating other mixing types.

Chapter 5

Convolutional Mixtures

5.1 Introduction

In this chapter we deal with convolutional mixtures and mainly for the case of two sources and two sensors. First, we consider some general remarks about these mixtures, and then we use the gradient and projection approaches of the previous chapter for separating them. In each approach, after obtaining the estimation equations, the separation algorithm will be presented followed by some experimental results. Finally, we present a special type of convolutional mixtures, called Post-Convolutional mixtures which has a nice property: *instantaneous* independence is sufficient for separating it.

5.2 Preliminary issues

In convolutional mixtures, the observation vector $\mathbf{x}(n)$ is obtained from the sources $\mathbf{s}(n)$ through the mixing system:

$$\mathbf{x}(n) = [\mathbf{A}(z)] \mathbf{s}(n) \quad (5.1)$$

This model appears for example in modeling the “cocktail party problem” (separating mixed speech signals): the effects of one speech on different microphones do not differ only in a scale factor and there is usually different transfer functions and at least a time delay.

For separating these mixtures, we first need to assume that $\mathbf{A}(z)$ is invertible. This is equivalent to assuming that $\det(\mathbf{A}(z))$ is minimum phase [91]. Then, we apply the separating filter $\mathbf{B}(z)$ to obtain the output signals by:

$$\mathbf{y}(n) = [\mathbf{B}(z)] \mathbf{x}(n) \quad (5.2)$$

As it has been proved in [108], if $\mathbf{B}(z)$ is determined in such a way that the output signals become independent, then the sources are separated. However, the indeterminacies are more

serious for convolutional mixtures than for linear instantaneous mixtures: the sources will be determined up to a permutation and a *filtering indeterminacy*. This can be seen from the fact that if $s_1(n)$ and $s_2(n)$ are independent, then for any filters $H_1(z)$ and $H_2(z)$, the signals $[H_1(z)]s_1(n)$ and $[H_2(z)]s_2(n)$ will also be independent.


Here, we summarize some important remarks about convolutional mixtures.


Remark 1. (Independence Criterion) As it has been stated in the section 4.3, in convolutional mixtures we are dealing with random processes, not random variables. The output independence, too, must be seen as the independence of two random processes, not independence of random variables. In other words, the independence of $y_1(n)$ and $y_2(n)$ for all n does not insure the source separation, and we need that $y_1(n)$ and $y_2(n - m)$ be independent for all n and all m (see the example of the page 34). Therefore, $I(y_1(n), y_2(n))$ is not a separation criterion, and we will use the separation criterion:

$$J \triangleq \sum_{m=-M}^{+M} I(y_1(n), y_2(n - m)) \quad (5.3)$$

Theoretically, the limits of this summation must be $-\infty$ and $+\infty$. However, this is not practically possible, and as explained in the section 4.3, we use $M = 2p$, where p is the maximum degree of the separating filters. In fact, in many previously known methods for separating the convolutional mixtures, this limitation exists implicitly. For example, Charkani [23] and Nguyen and Jutten [100] have used approximate independence criteria of $y_1(n)$ and $y_2(n - m)$ for the values of m between 0 and M , where M is the maximum degree of the FIR filters which exist in their feedback models.

Since this criterion is very expensive, we use its stochastic version. In other words, at each iteration we take $I(y_1(n), y_2(n - m))$ as the current independence criterion, but with a different randomly chosen m from the set $\{-M, \dots, M\}$.

 **Note:** It may seem that it is harmful to choose a very large M in (5.3). However, in our statistical implementation of this criterion, if M is highly greater than the degree of the whole mixing-separating system, and if the sources are iid, then in most iterations the criterion has no information, and it slows down the convergence rate.

 **Note:** This criterion is for the case of two sources and two sensors. One can define a similar criterion for higher dimensions, too. For example, for the 3-dimensional case:

$$J \triangleq \sum_{m_1} \sum_{m_2} I(y_1(n), y_2(n - m_1), y_3(n - m_2)) \quad (5.4)$$

It is obvious that the computational load increases very rapidly, with increasing the dimensionality.

Moreover, it is necessary to estimate the joint PDF's of the outputs. Consequently, we have developed and tested the algorithms mainly for the case of two sources and two sensors.

Remark 2. (Filtering Indeterminacy) Because of the filtering indeterminacy which exists in separating convolutive mixtures, after achieving output independence, the outputs will be the filtered versions of the sources, instead of the original ones. However, as it has been explained in the chapter 2, the effect of each source on each sensor can be determined. In fact, after obtaining output independence, if we calculate the filters $H_{ij}(z)$ which minimize:

$$E \left\{ (x_i(n) - [H_{ij}(z)] y_j(n))^2 \right\} \quad (5.5)$$

then, the signal $[H_{ij}(z)] y_j(n)$ will be the effect of j -th source on the i -th sensor, that is, what the i -th sensor receives, when there is no other sources (here, we assumed that there is no permutation). It is obvious that this is sufficient for many real applications (*e.g.* in the cocktail party problem).

Remark 3. (FIR Separating Filters) Throughout this thesis, we will always use FIR separating filters. In digital signal processing, when one uses an FIR filter for estimating an inverse system, there must exist some good reasons for assuming that the system is itself all-pole, or one must be sure that the degree of the FIR filter is sufficiently large for *approximately* modeling the inverse system.

However, in separating convolutive mixtures, we can take the filtering indeterminacy as an advantage, and use it to apply FIR separating filters. In fact, if all the components of $\mathbf{A}(z)$ are rational functions in z , then the separating filter $\mathbf{B}(z)$ can be always chosen FIR. To clarifying the idea, consider the two-dimensional mixing system:

$$\mathbf{A}(z) = \begin{bmatrix} \frac{N_{11}(z)}{D_{11}(z)} & \frac{N_{12}(z)}{D_{12}(z)} \\ \frac{N_{21}(z)}{D_{21}(z)} & \frac{N_{22}(z)}{D_{22}(z)} \end{bmatrix} \quad (5.6)$$

Then, the following FIR system results in the independence of the outputs:

$$\mathbf{B}(z) = \begin{bmatrix} \frac{N_{22}(z)}{D_{22}(z)} & -\frac{N_{12}(z)}{D_{12}(z)} \\ -\frac{N_{21}(z)}{D_{21}(z)} & \frac{N_{11}(z)}{D_{11}(z)} \end{bmatrix} \prod_{i,j} D_{ij}(z) \quad (5.7)$$

Remark 4. (Separation versus system identification) In convolutive mixtures, separation of the sources does not necessarily identify the mixing system. Of course, theoretically the outputs become independent only when the overall system $\mathbf{C}(z) = \mathbf{B}(z)\mathbf{A}(z)$ is a diagonal matrix. However, in practice (as confirmed by our experiences, too) it is possible to achieve the separation, without obtaining a diagonal $\mathbf{C}(z)$.

Example. Suppose that the sources have a high temporal correlation, and let the overall mixing-separating system be:

$$\begin{cases} y_1(n) = s_1(n) + \alpha s_2(n) + \beta s_2(n-1) \\ y_2(n) = s_2(n) \end{cases} \quad (5.8)$$

Mathematically, y_1 and y_2 are independent if and only if $\alpha = \beta = 0$. However, since the correlation between $s_2(n)$ and $s_2(n-1)$ is large (for instance, when s_2 is low frequency with respect to the sampling frequency), $s_2(n) \simeq s_2(n-1)$, and hence if $\beta \simeq -\alpha$, then $y_1(n) \simeq s_1(n)$ and we have obtained good separation, without obtaining a diagonal $\mathbf{C}(z)$. In fact, having in mind that we are working with a limited number of samples, the algorithm cannot discriminate between the cases $\beta \simeq -\alpha$ and $\beta = \alpha = 0$.

5.3 Gradient approach

In this section, we apply the gradient approach for separating convolutional mixtures. We have published the algorithm presented in this section in [11]. Suppose we would like to separate the sources by means of FIR filters with the maximum degree p . Then the separating system will be:

$$\mathbf{y}(n) = \mathbf{B}_0 \mathbf{x}(n) + \mathbf{B}_1 \mathbf{x}(n-1) + \cdots + \mathbf{B}_p \mathbf{x}(n-p) \quad (5.9)$$

and we must estimate the matrices \mathbf{B}_k which generate independent outputs. Hence, we must first calculate the gradients of the separation criterion with respect to each \mathbf{B}_k , and then apply a steepest descent gradient algorithm on each \mathbf{B}_k .

5.3.1 Calculating gradients

In Section 4.8.2 we have shown how to calculate the gradient of $I(y_1(n), y_2(n))$ with respect to \mathbf{B}_k 's. However, it is not sufficient, because the criterion 5.3 requires the gradients of $I(y_1(n), y_2(n-m))$, for all m .

For calculating these gradients, we first define the following notation for any signal $\mathbf{y}(n) = (y_1(n), y_2(n))^T$:

$$\mathbf{y}^{(m)}(n) \triangleq \begin{bmatrix} y_1(n) \\ y_2(n-m) \end{bmatrix} \quad (5.10)$$

In other words, $\mathbf{y}^{(m)}(n)$ shows a shift operation on the second component of $\mathbf{y}(n)$ with respect to the first one.

With this notation, we must calculate the gradient of $I(\mathbf{y}^{(m)}(n))$. To calculate its gradient with respect to \mathbf{B}_k , let $\hat{\mathbf{B}}_i = \mathbf{B}_i$ for $i \neq k$ and $\hat{\mathbf{B}}_k = \mathbf{B}_k + \mathcal{E}$, where \mathcal{E} is a ‘‘small’’

matrix. Then, from (5.9) we conclude that $\hat{\mathbf{y}}(n) \triangleq [\hat{\mathbf{B}}(z)]\mathbf{x}(n) = \mathbf{y}(n) + \mathcal{E}\mathbf{x}(n - k)$. Denoting $\mathbf{\Delta}(n) = \mathcal{E}\mathbf{x}(n - k)$, we will have $\hat{\mathbf{y}}^{(m)}(n) = \mathbf{y}^{(m)}(n) + \mathbf{\Delta}^{(m)}(n)$, and hence from the Theorem 4.1:

$$\begin{aligned} I(\hat{\mathbf{y}}^{(m)}(n)) - I(\mathbf{y}^{(m)}(n)) &= E \left\{ \boldsymbol{\beta}_{\mathbf{y}^{(m)}}(n)^T \mathbf{\Delta}^{(m)}(n) \right\} \\ &= E \left\{ \beta_{\mathbf{y}^{(m)},1}(n) \Delta_1^{(m)}(n) \right\} + E \left\{ \beta_{\mathbf{y}^{(m)},2}(n) \Delta_2^{(m)}(n) \right\} \\ &= E \left\{ \beta_{\mathbf{y}^{(m)},1}(n) \Delta_1(n) \right\} + E \left\{ \beta_{\mathbf{y}^{(m)},2}(n) \Delta_2(n - m) \right\} \\ &= E \left\{ \beta_{\mathbf{y}^{(m)},1}(n) \Delta_1(n) \right\} + E \left\{ \beta_{\mathbf{y}^{(m)},2}(n + m) \Delta_2(n) \right\} \end{aligned} \quad (5.11)$$

where $\boldsymbol{\beta}_{\mathbf{y}^{(m)}}(n)$ stands for $\boldsymbol{\beta}_{\mathbf{y}^{(m)}}(\mathbf{y}^{(m)}(n))$. Note that in writing the above equations, we have assumed that the sources are stationary. Now, we define:

$$\boldsymbol{\beta}_m(n) \triangleq \boldsymbol{\beta}_{\mathbf{y}^{(m)}}^{(-m)}(n) = \begin{bmatrix} \beta_{\mathbf{y}^{(m)},1}(n) \\ \beta_{\mathbf{y}^{(m)},2}(n + m) \end{bmatrix} \quad (5.12)$$

In other words, for obtaining $\boldsymbol{\beta}_m(n)$, the second component of $\mathbf{y}(n)$ must be shifted m times, then after calculating its SFD, the second component of SFD must be shifted back m times:

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

By using this notation, we can write:

$$\begin{aligned} I(\hat{\mathbf{y}}^{(m)}(n)) - I(\mathbf{y}^{(m)}(n)) &= E \left\{ \boldsymbol{\beta}_m(n)^T \mathbf{\Delta}(n) \right\} \\ &= E \left\{ \boldsymbol{\beta}_m(n)^T \mathcal{E}\mathbf{x}(n - k) \right\} \\ &= \langle \mathcal{E}, E \left\{ \boldsymbol{\beta}_m(n) \mathbf{x}(n - k)^T \right\} \rangle \end{aligned} \quad (5.13)$$

and finally:

$$\boxed{\frac{\partial}{\partial \mathbf{B}_k} I(y_1(n), y_2(n - m)) = E \left\{ \boldsymbol{\beta}_m(n) \mathbf{x}(n - k)^T \right\}} \quad (5.14)$$

5.3.2 Separating algorithm

From (5.14) we can design a gradient based algorithm for separating the sources. Practically, to overcome the scale indeterminacy and preventing the algorithm from converging to $\mathbf{y} = \mathbf{0}$, we normalize the output energies at each iteration. The final algorithm is shown in Fig. 5.1.

5.3.3 Experimental results

Here, we present some experimental results. For measuring the separation quality, we use the output SNR's defined by:

$$\text{SNR}_i = 10 \log_{10} \frac{E \{ y_i^2 \}}{E \{ y_i^2 |_{s_i=0} \}} \quad (5.15)$$

- 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 $\beta^*(n)$, the SFD of $(y_1(n), y_2(n-m))^T$.
 3. Let $\beta_m(n) = (\beta_1^*(n), \beta_2^*(n+m))^T$.
 4. For $k = 0 \dots p$:

$$\mathbf{B}_k \leftarrow \mathbf{B}_k - \mu E \{ \beta_m(n) \mathbf{x}(n-k)^T \}$$
 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 row of $\mathbf{B}(z)$ by σ_i .
- Repeat until convergence.

Figure 5.1: Gradient algorithm for separating convolutional mixtures.

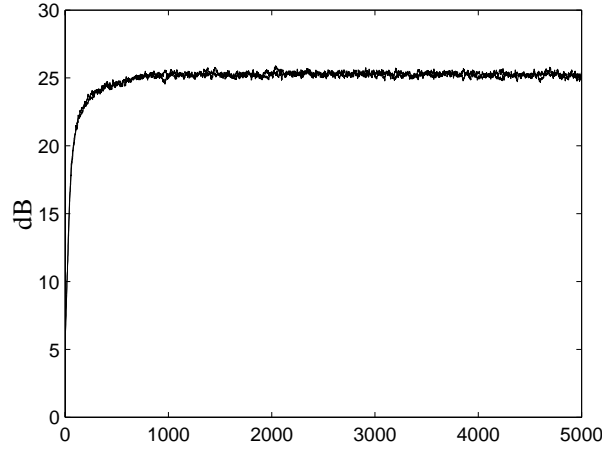



Figure 5.2: Averaged output SNR's versus iterations, for the experiment 5.1

where $y_i|_{s_i=0}$ stands for what is at the i -th output, where the i -th input is zero (assuming there is no permutation). By using this definition, SNR will be a measure of “separation”, that is, a high SNR means that there is not a large leakage from the other sources to this output. However, it can be a filtered version of the actual source, and a post-processing can be done for recovering the effect of this source on each sensor (as noted in the remark 2, page 61).

 **Experiment 5.1.** In this experiment, we mix two uniform random sources with zeros means and unit variances. The mixing system is:

$$\mathbf{A}(z) = \begin{bmatrix} 1 + 0.2z^{-1} + 0.1z^{-2} & 0.5 + 0.3z^{-1} + 0.1z^{-2} \\ 0.5 + 0.3z^{-1} + 0.1z^{-2} & 1 + 0.2z^{-1} + 0.1z^{-2} \end{bmatrix} \quad (5.16)$$

For separating this mixture, we use second order filters ($p = 2$), and $M = 2p = 4$. The adaptation rate is $\mu = 0.3$. The number of observations is 500, and the Pham's estimator is used for estimating the SFD.

The experiment is repeated 100 times with different realizations of the sources. Figure 5.2 shows the averaged SNR's (taken over 100 repetition of the experiment) versus iterations.

The averaged calculated $\mathbf{B}(z)$ over this 100 experiments is (up to 2 decimal digits):

$$\mathbf{B}(z) = \begin{bmatrix} 1.31 + 0.04z^{-1} + 0.04z^{-2} & -0.66 - 0.28z^{-1} - 0.04z^{-2} \\ -0.66 - 0.30z^{-1} - 0.06z^{-2} & 1.31 + 0.10z^{-1} + 0.07z^{-2} \end{bmatrix} \quad (5.17)$$

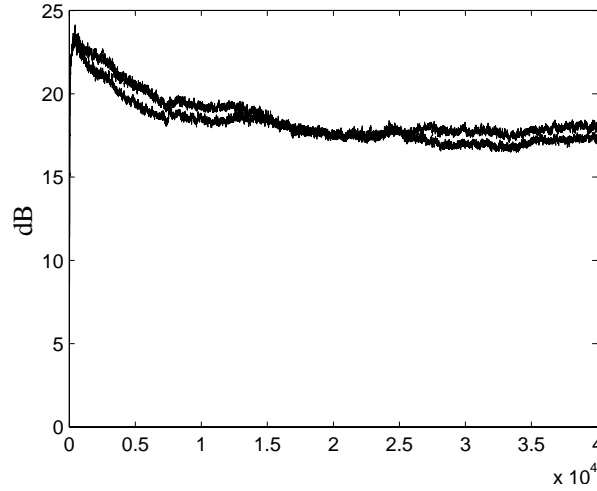



Figure 5.3: Averaged output SNR's versus iterations, with polynomial estimation of the SFD

which results in the overall mixing-separating system (up to two decimal digits):


$$\mathbf{B}(z)\mathbf{A}(z) =$$

$$\begin{bmatrix} 0.98 - 0.03z^{-1} + 0.02z^{-2} - 0.03z^{-3} + 0.00z^{-4} & 0.00 + 0.00z^{-1} + 0.01z^{-2} - 0.02z^{-3} + 0.00z^{-4} \\ -0.00 + 0.01z^{-1} + 0.00z^{-2} - 0.01z^{-3} + 0.00z^{-4} & 0.99 + 0.01z^{-1} + 0.03z^{-2} - 0.03z^{-3} + 0.00z^{-4} \end{bmatrix} \quad (5.18)$$

Run time: The time required for running 100 iterations of the experiment for each realization of the sources is about 4.94 seconds.

 **Experiment 5.2.** The above experiment is repeated with the polynomial estimator of SFD. Figure 5.3 shows the resulting SNR's (averaged on 50 experiments). It can be seen that the polynomial estimation of SFD leads to worse performance than the Pham's estimator. We have also tried the kernel estimation of SFD (not presented here). This estimator leads to very bad results in convolutional mixtures (although it gives better results than Pham's estimator in PNL mixtures). The interpretation of this result remains as an open question.

Run time: The time required for running 100 iterations of the experiment for each realization of the sources is about 1.37 seconds. However, the price of this speed-up compared with the previous one is the less separation quality.

 **Experiment 5.3.** To see the effect of M , we repeat the experiment 5.1 with $M = 2$. All the other parameters are the same as the experiment 5.1. Figure 5.4 shows the averaged output SNR's taken over 100 runs of the experiment.

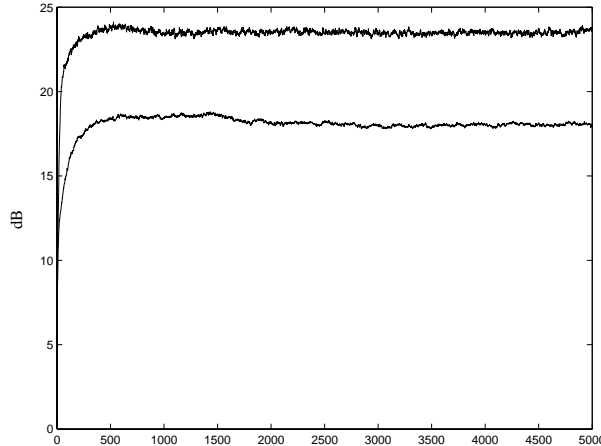


Figure 5.4: Averaged output SNR's versus iterations for $M = 2$, which is smaller than what is required.

We have repeated the experiment for $M = 6$, too. The obtained result is similar to the one shown in the Fig. 5.2, and we do not plot it again.

5.4 Projection approach

In this section, the projection algorithm for separating convolutive mixtures will be developed (refer to Section 4.8.3 for the idea of the projection approach). For using this approach, we must do, at each iteration, the following main tasks: (a) Updating the outputs by $\mathbf{y}^{(m)} \leftarrow \mathbf{y}^{(m)} - \mu \boldsymbol{\beta}_{\mathbf{y}^{(m)}}(\mathbf{y}^{(m)})$, (b) Finding the matrix $\mathbf{B}(z)$ which minimizes the error $E \left\{ \|\mathbf{y}(n) - [\mathbf{B}(z)] \mathbf{x}(n)\|^2 \right\}$, and (c) Replacing $\mathbf{y}(n)$ by $[\mathbf{B}(z)] \mathbf{x}(n)$.

5.4.1 Calculating the optimal mapping

Here, our objective is to find the filter $\mathbf{B}(z)$ which minimizes $E \left\{ \|\mathbf{y}(n) - [\mathbf{B}(z)] \mathbf{x}(n)\|^2 \right\}$. For simplicity we use the notation \mathbf{x}_i for $\mathbf{x}(n-i)$, and \mathbf{y} for $\mathbf{y}(n)$. Hence, we are looking for the matrices $\mathbf{B}_0, \mathbf{B}_1, \dots, \mathbf{B}_p$ which minimize the empirical expectation:

$$\mathcal{C} = \hat{E} \left\{ \left\| \mathbf{y} - \sum_{i=0}^p \mathbf{B}_i \mathbf{x}_i \right\|^2 \right\} \quad (5.19)$$

Note that in practice, we have a limited number of samples. Suppose that we know the values of $\mathbf{x}(n)$, for $n = 0, \dots, T-1$. Then, in computing the values of $\mathbf{y}(n)$ for $0 \leq n \leq p-1$, we need the values of $\mathbf{x}(n)$ for $n < 0$, and as usual we assume that they are zero. On the other hand, for estimating the expectation operation, we use a temporal averaging. However, since

we cannot trust on the first p values of $\mathbf{y}(n)$, we exclude them from the averaging. In other words, in the following equations, \hat{E} means:

$$\hat{E} \equiv \frac{1}{T-p} \sum_{n=p}^{T-1} \quad (5.20)$$

Now, we rewrite the error term (5.19) as:

$$\begin{aligned} \mathcal{C} &= \hat{E} \left\{ \left(\mathbf{y} - \sum_{i=0}^p \mathbf{B}_i \mathbf{x}_i \right)^T \left(\mathbf{y} - \sum_{j=0}^p \mathbf{B}_j \mathbf{x}_j \right) \right\} \\ &= \hat{E} \left\{ \left(\mathbf{y}^T - \sum_{i=0}^p \mathbf{x}_i^T \mathbf{B}_i^T \right) \left(\mathbf{y} - \sum_{j=0}^p \mathbf{B}_j \mathbf{x}_j \right) \right\} \\ &= \hat{E} \{ \mathbf{y}^T \mathbf{y} \} - 2 \sum_{j=0}^p \hat{E} \{ \mathbf{y}^T \mathbf{B}_j \mathbf{x}_j \} + \sum_{i=0}^p \sum_{j=0}^p \hat{E} \{ \mathbf{x}_i^T \mathbf{B}_i^T \mathbf{B}_j \mathbf{x}_j \} \\ &= \hat{E} \{ \mathbf{y}^T \mathbf{y} \} - 2 \sum_{j=0}^p \hat{E} \{ \mathbf{y}^T \mathbf{B}_j \mathbf{x}_j \} + \sum_{i=0}^p \hat{E} \{ \mathbf{x}_i^T \mathbf{B}_i^T \mathbf{B}_i \mathbf{x}_i \} + \sum_{i=0}^p \sum_{\substack{j=0 \\ j \neq i}}^p \hat{E} \{ \mathbf{x}_i^T \mathbf{B}_i^T \mathbf{B}_j \mathbf{x}_j \} \end{aligned}$$

By using Lemmas E.3 and E.4 of Appendix E, we will have:

$$\begin{aligned} \frac{\partial \mathcal{C}}{\partial \mathbf{B}_k} &= -2 \hat{E} \{ \mathbf{y} \mathbf{x}_k^T \} + 2 \mathbf{B}_k \hat{E} \{ \mathbf{x}_k \mathbf{x}_k^T \} + \sum_{\substack{j=0 \\ j \neq k}}^p \frac{\partial}{\partial \mathbf{B}_k} \hat{E} \{ \mathbf{x}_k^T \mathbf{B}_k^T \mathbf{B}_j \mathbf{x}_j \} + \sum_{\substack{i=0 \\ i \neq k}}^p \frac{\partial}{\partial \mathbf{B}_k} \hat{E} \{ \mathbf{x}_i^T \mathbf{B}_i^T \mathbf{B}_k \mathbf{x}_k \} \\ &= -2 \hat{E} \{ \mathbf{y} \mathbf{x}_k^T \} + 2 \mathbf{B}_k \hat{E} \{ \mathbf{x}_k \mathbf{x}_k^T \} + 2 \sum_{\substack{j=0 \\ j \neq k}}^p \frac{\partial}{\partial \mathbf{B}_k} \hat{E} \{ \mathbf{x}_k^T \mathbf{B}_k^T \mathbf{B}_j \mathbf{x}_j \} \\ &= -2 \hat{E} \{ \mathbf{y} \mathbf{x}_k^T \} + 2 \mathbf{B}_k \hat{E} \{ \mathbf{x}_k \mathbf{x}_k^T \} + 2 \sum_{\substack{j=0 \\ j \neq k}}^p \mathbf{B}_j \hat{E} \{ \mathbf{x}_j \mathbf{x}_k^T \} \\ &= -2 \hat{E} \{ \mathbf{y} \mathbf{x}_k^T \} + 2 \sum_{j=0}^p \mathbf{B}_j \hat{E} \{ \mathbf{x}_j \mathbf{x}_k^T \} \end{aligned}$$

Finally, we let $\partial \mathcal{C} / \partial \mathbf{B}_k = \mathbf{0}$ for $k = 0, \dots, p$, and from there we obtain the following equations for determining the matrices \mathbf{B}_k 's:

$$\sum_{j=0}^p \mathbf{B}_j \hat{E} \{ \mathbf{x}_j \mathbf{x}_k^T \} = \hat{E} \{ \mathbf{y} \mathbf{x}_k^T \} \quad , \quad k = 1, \dots, p \quad (5.21)$$

For writing the above equations in a more compact form, we first define the following $N \times N$ matrices (note that \hat{E} denotes the empirical expectation as defined in (5.20)):

$$\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(j, k) \triangleq \hat{E} \{ \mathbf{x}(n-j) \mathbf{x}^T(n-k) \} \quad (5.22)$$

$$\hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}}(j, k) \triangleq \hat{E} \{ \mathbf{y}(n-j) \mathbf{x}^T(n-k) \} \quad (5.23)$$

By these definitions, we can write (5.21) as:

$$\sum_{j=0}^p \mathbf{B}_j \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(j, k) = \hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}}(0, k) \quad , \quad k = 1, \dots, p \quad (5.24)$$

or:

$$\begin{aligned} [\mathbf{B}_0 \quad \mathbf{B}_1 \quad \dots \quad \mathbf{B}_p] & \begin{bmatrix} \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(0, 0) & \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(0, 1) & \dots & \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(0, p) \\ \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(1, 0) & \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(1, 1) & \dots & \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(1, p) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(p, 0) & \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(p, 1) & \dots & \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(p, p) \end{bmatrix} \\ & = \begin{bmatrix} \hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}}(0, 0) & \hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}}(0, 1) & \dots & \hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}}(0, p) \end{bmatrix} \end{aligned} \quad (5.25)$$

We define now the large matrices:

$$\mathbf{B} \triangleq [\mathbf{B}_0 \quad \mathbf{B}_1 \quad \dots \quad \mathbf{B}_p] \quad (5.26)$$

$$\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}} \triangleq \begin{bmatrix} \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(0, 0) & \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(0, 1) & \dots & \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(0, p) \\ \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(1, 0) & \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(1, 1) & \dots & \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(1, p) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(p, 0) & \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(p, 1) & \dots & \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}(p, p) \end{bmatrix} \quad (5.27)$$


$$\hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}} \triangleq \begin{bmatrix} \hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}}(0, 0) & \hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}}(0, 1) & \dots & \hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}}(0, p) \end{bmatrix} \quad (5.28)$$

The dimensions of \mathbf{B} , $\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}$ and $\hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}}$ are $N \times (p+1)N$, $(p+1)N \times (p+1)N$ and $N \times (p+1)N$, respectively.

Consequently, (5.25) will be written as $\mathbf{B}\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}} = \hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}}$, and hence:

$$\boxed{\mathbf{B} = \hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}} \hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}^{-1}} \quad (5.29)$$

which determines the \mathbf{B}_k 's.

 **Note:** If we assume that the signals are stationary, then $\hat{\mathbf{R}}_{\mathbf{x}\mathbf{x}}$ and $\hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}}$, as defined in equations (5.22) and (5.23), will be the autocorrelation and cross correlation matrices $\mathbf{R}_{\mathbf{x}\mathbf{x}}(k-j)$ and $\mathbf{R}_{\mathbf{y}\mathbf{x}}(k-j)$, where:

$$\mathbf{R}_{\mathbf{x}\mathbf{x}}(k) \triangleq E \{ \mathbf{x}(n) \mathbf{x}^T(n-k) \} \quad (5.30)$$

$$\mathbf{R}_{\mathbf{y}\mathbf{x}}(k) \triangleq E \{ \mathbf{y}(n) \mathbf{x}^T(n-k) \} \quad (5.31)$$


And the equation system (5.25) will be written as:

$$\begin{aligned}
 [\mathbf{B}_0 \quad \mathbf{B}_1 \quad \cdots \quad \mathbf{B}_p] \begin{bmatrix} \mathbf{R}_{xx}(0) & \mathbf{R}_{xx}(1) & \cdots & \mathbf{R}_{xx}(p) \\ \mathbf{R}_{xx}(-1) & \mathbf{R}_{xx}(0) & \cdots & \mathbf{R}_{xx}(p-1) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_{xx}(-p) & \mathbf{R}_{xx}(-p+1) & \cdots & \mathbf{R}_{xx}(0) \end{bmatrix} \\
 = \begin{bmatrix} \mathbf{R}_{yx}(0) & \mathbf{R}_{yx}(1) & \cdots & \mathbf{R}_{yx}(p) \end{bmatrix}
 \end{aligned} \tag{5.32}$$

or by taking the transpose of both sides:

$$\begin{bmatrix} \mathbf{R}_{xx}(0) & \mathbf{R}_{xx}(1) & \cdots & \mathbf{R}_{xx}(p) \\ \mathbf{R}_{xx}(-1) & \mathbf{R}_{xx}(0) & \cdots & \mathbf{R}_{xx}(p-1) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_{xx}(-p) & \mathbf{R}_{xx}(-p+1) & \cdots & \mathbf{R}_{xx}(0) \end{bmatrix} \begin{bmatrix} \mathbf{B}_0^T \\ \mathbf{B}_1^T \\ \vdots \\ \mathbf{B}_p^T \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{yx}(0) \\ \mathbf{R}_{yx}(-1) \\ \vdots \\ \mathbf{R}_{yx}(-p) \end{bmatrix} \tag{5.33}$$

which is very similar to Yule-Walker equations in Auto-Regressive (AR) modeling of data. However, here the dimension is higher, and each component of the above equation stands for a matrix, not a scalar. This similarity comes from the similarity of (5.9) to AR data modeling. Due to the similarity of equation (5.25) to the ‘‘covariance method’’ in AR data modeling, we call it the ‘‘covariance method’’. In the same manner, we call the method of equation (5.32), the ‘‘autocorrelation method’’. However, we prefer to use the covariance method, because there is no approximation in it. For example, if we use the autocorrelation method, we have implicitly assumed that $\hat{\mathbf{R}}_{xx}(0,0) = \hat{\mathbf{R}}_{xx}(1,1) = \cdots = \hat{\mathbf{R}}_{xx}(p,p) = \mathbf{R}_{xx}(0)$.

 **Experiment 5.4.** To see the effect of the approximations used in the autocorrelation method for estimating \mathbf{B}_k 's, we use two random signals $x_1(n)$ and $x_2(n)$, both with uniform distribution on $(-1, 1)$. Then, we mix them by the mixing filter:

$$\mathbf{A}(z) = \begin{bmatrix} 1 + z^{-1} + 0.7z^{-2} & 0.5 + 0.3z^{-1} + 0.2z^{-2} \\ 0.5 + 0.3z^{-1} + 0.9z^{-2} & 1 + 0.8z^{-1} + 0.8z^{-2} \end{bmatrix}$$

Now, by knowing $\mathbf{x}(n)$ and $\mathbf{y}(n) \triangleq [\mathbf{A}(z)]\mathbf{x}(n)$, we would like to estimate a second order filter $\mathbf{B}(z)$ which minimizes (5.9). Clearly, the true answer is $\mathbf{B}(z) = \mathbf{A}(z)$. If the ij 'th component of $\mathbf{B}(z)$ is denoted by $B_{ij}(z) = \sum_k b_{ij}^{(k)} z^{-k}$, then we define the estimation error by $\mathcal{E}^2 \triangleq \sum_{i,j,k} \left(b_{ij}^{(k)} - a_{ij}^{(k)} \right)^2$.

Now, if we use 1000 sample points, the covariance method results in $\mathcal{E} = 8.35 \times 10^{-16}$ but the autocorrelation method results in $\mathcal{E} = 0.0053$ (for 12 parameters). In fact, there is no approximation in the covariance method, and hence, even this small error must be considered as a result of the limitations of representing floating point numbers in digital computers. Of course, this smaller error has been obtained at the expense of an increased computational volume.

- Initialization: $\mathbf{y}(n) = \mathbf{x}(n)$.
- Loop:
 1. Choose a random m from the set $\{-M, \dots, +M\}$.
 2. Estimate $\beta_{\mathbf{y}^{(m)}}$, the SFD of $(y_1(n), y_2(n-m))^T$.
 3. Update the outputs by:


$$\mathbf{y}^{(m)} \leftarrow \mathbf{y}^{(m)} - \mu \beta_{\mathbf{y}^{(m)}}(\mathbf{y}^{(m)})$$
 4. Remove the DC of each output, and normalize its energy.
 5. Compute the matrices \mathbf{B}_k , $k = 0, \dots, p$, from (5.25).
 6. Let $\mathbf{y}(n) = [\mathbf{B}(z)] \mathbf{x}(n)$.
- Repeat until convergence.

Figure 5.5: Projection algorithm for separating convolutive mixtures.

5.4.2 Separating algorithm

From the previous section, the final separating algorithm for convolutive mixtures will be obtained as shown in Fig. 5.5.

5.4.3 Experimental results

 **Experiment 5.5.** The experiment 5.1 is repeated for the projection approach. That is, we mix two uniform random sources with zeros means and unit variances. The mixing system is:

$$\mathbf{A}(z) = \begin{bmatrix} 1 + 0.2z^{-1} + 0.1z^{-2} & 0.5 + 0.3z^{-1} + 0.1z^{-2} \\ 0.5 + 0.3z^{-1} + 0.1z^{-2} & 1 + 0.2z^{-1} + 0.1z^{-2} \end{bmatrix} \quad (5.34)$$

For separating this mixture, we use second order filters ($p = 2$), and $M = 2p = 4$. The adaptation rate is $\mu = 0.1$. The number of observations is 500, and the Pham's estimator is used for estimating the SFD.

The averaged SNR's (taken over 100 runs of the algorithm), can be seen in the Fig. 5.6.

A comparison between the figures 5.6 and 5.2 shows that the quality of the projection approach is a slightly less than the gradient approach.

Run time: The time required for running 100 iterations of the experiment for each realization of the sources is about 5.13 seconds.

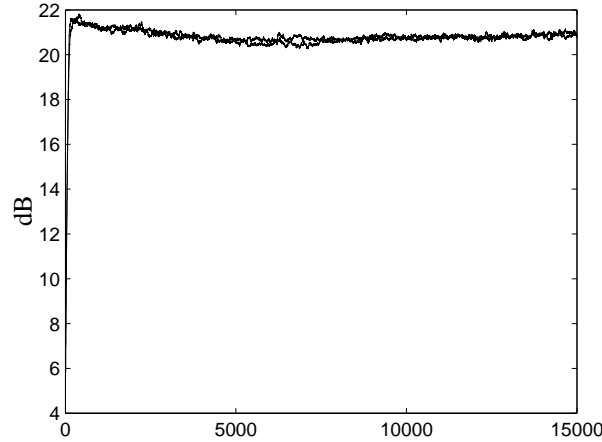


Figure 5.6: The output SNR's versus iterations by using the projection algorithm (Experiment 5.5).

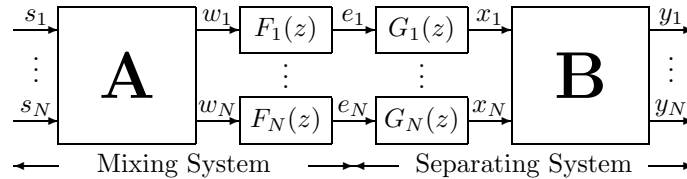


Figure 5.7: The mixing-separating system for Post-Convolutional mixtures.

5.5 A special case: Post-Convolutional mixtures

In this section, we present briefly a special kind of convolutional mixtures, called Post-Convolutional mixtures. We have presented the results of this section in [10] and [9].

5.5.1 Preliminary issues

In Post-Convolutional mixtures, it is assumed that the mixture is itself instantaneous, but the sensors are not ideal and introduce some frequency distortion (which can be due to the amplifiers associated to the sensors). For separating the mixture, we first compensate for the frequency distortion of the sensors, and then, separate the resulting instantaneous mixture. Figure 5.7 shows the whole mixing-separating system.

Let $H_i(z) \triangleq G_i(z)F_i(z)$. Then, for this special kind of convolutional mixtures, we can state the following theorem:

Theorem 5.1 *Let the sources s_1 and s_2 be iid with at most one Gaussian source, and the*

mixing matrix:

$$\mathbf{A} = \begin{bmatrix} 1 & a_1 \\ a_2 & 1 \end{bmatrix} \quad (5.35)$$

be a nonsingular matrix, and $a_1 a_2 \neq -1$. Moreover, suppose that there is an integer m such that $h_1(m)h_2(m) \neq 0$. If $y_1(n)$ and $y_2(n)$ are independent for all n , then $y_1(n)$ and $y_2(n-m)$ will be independent for all n and all m .

This theorem states that for iid sources, under some mild assumptions about the mixing system, random variable (instantaneous) independence is sufficient for source separation. Consequently, the outputs can be treated as random variables, and $I(\mathbf{y})$ can be used as a separation criterion. Moreover, although we proved the theorem only for iid sources, our experiments show that the separation algorithm works for non-iid sources, too.

The proof of the theorem has been left to the Appendix B. It is also interesting to note that the following corollary comes directly from the proof of the theorem:

Corollary 1 *Under the assumptions of theorem 5.1, and if the matrix \mathbf{A} satisfies the condition: $a_1 \neq 0$ or $a_2 \neq 0$, then the independence of $y_1(n)$ and $y_2(n)$ implies that $H_1(z) = cH_2(z)$, where c is an arbitrary constant.*

This corollary shows that if the sources are “really mixed” in the instantaneous stage, then the filtering indeterminacy is less than the general convolutive case: the unknown filters are the same (up to a scaling factor) on all the outputs.

It also must be noted that this result is stated and proved only for two sources and two sensors. Unfortunately, its extension to more general case does not seem to be immediate.

5.5.2 Estimating equations

Here, we use the gradient approach for separating post-convolutive mixtures. However, since we had worked on these mixtures before finding the differential of the mutual information (Theorem 4.1), we don’t use it in this section. Hence, we will see how the SFD will appear in the final equations without using this theorem.

For developing estimating equations, we assume that:

1. The compensating filters can be FIR causal filters of order p .
2. The filters $F_i(z)$ are causal.
3. $f_i[0] \neq 0 \quad \forall i$.

With these assumptions, the compensating filters can be chosen of the form:

$$G_i(z) = 1 + \sum_{j=1}^p g_{ij} z^{-j} \quad (5.36)$$

From causality of the filters, and from $f_i[0] \neq 0$ and $g_i[0] \neq 0$, we can deduce $h_i[0] \neq 0$, and hence the theorem 5.1 can be applied.

Instantaneous stage

For the instantaneous part, all the calculations for obtaining (4.8) can be repeated, which results in the same equation:

$$\boxed{\frac{\partial I(\mathbf{y})}{\partial \mathbf{B}} = E \{ \boldsymbol{\psi}_{\mathbf{y}}(\mathbf{y}) \mathbf{x}^T \} - \mathbf{B}^{-T}} \quad (5.37)$$

where $\boldsymbol{\psi}_{\mathbf{y}}(\mathbf{y}) = (\psi_1(y_1), \dots, \psi_N(y_N))^T$.

Filtering stage

From $H(\mathbf{y}) = H(\mathbf{x}) + \ln |\det B|$, we have:

$$\begin{aligned} \frac{\partial H(\mathbf{y})}{\partial g_{ij}} &= \frac{\partial H(\mathbf{x})}{\partial g_{ij}} = -E \left\{ \frac{\partial}{\partial g_{ij}} \ln p_{\mathbf{x}}(\mathbf{x}) \right\} \\ &= -E \left\{ \frac{\partial \ln p_{\mathbf{x}}(\mathbf{x})}{\partial x_i} \cdot \frac{\partial x_i}{\partial g_{ij}} \right\} \end{aligned} \quad (5.38)$$

Thus:

$$\frac{\partial H(\mathbf{y}(n))}{\partial g_{ij}} = E \{ \varphi_{\mathbf{x},i}(\mathbf{x}(n)) e_i(n-j) \} \quad (5.39)$$

where $\varphi_{\mathbf{x},i}(\mathbf{x})$ is the i -th component of the JSF of \mathbf{x} . Now, for calculating the derivative of $\sum H(y_k)$ with respect to g_{ij} we write:

$$\begin{aligned} \frac{\partial H(y_k)}{\partial g_{ij}} &= -E \left\{ \frac{\partial}{\partial g_{ij}} \ln p_{y_k}(y_k) \right\} \\ &= E \left\{ \psi_k(y_k) \cdot \frac{\partial y_k}{\partial g_{ij}} \right\} \\ &= E \left\{ \psi_k(y_k) \frac{\partial}{\partial g_{ij}} \left(\sum_{l=1}^N b_{kl} x_l \right) \right\} \\ &= E \left\{ b_{ki} \psi_k(y_k) \frac{\partial x_i}{\partial g_{ij}} \right\} \end{aligned} \quad (5.40)$$

Thus:

$$\frac{\partial}{\partial g_{ij}} \sum_{k=1}^N H(y_k) = E \left\{ \left(\sum_{k=1}^N b_{ki} \psi_k(y_k(n)) \right) e_i(n-j) \right\} \quad (5.41)$$

By defining $N \times p$ matrices, $\mathbf{G} \triangleq [g_{ij}]$, and:

$$\mathbf{E}(n) \triangleq [\mathbf{e}(n-1), \mathbf{e}(n-2), \dots, \mathbf{e}(n-p)] \quad (5.42)$$

and the vector:

$$\boldsymbol{\alpha}(n) = \mathbf{B}^T \boldsymbol{\psi}_{\mathbf{y}}(\mathbf{y}(n)) - \boldsymbol{\varphi}_{\mathbf{x}}(\mathbf{x}(n)) \quad (5.43)$$

and applying (5.39) and (5.41), we will have:

$$\boxed{\frac{\partial I(\mathbf{y})}{\partial \mathbf{G}} = E \{ \text{diag}(\boldsymbol{\alpha}(n)) \mathbf{E}(n) \}} \quad (5.44)$$

It is interesting to note that from the property 4.5, we have $\boldsymbol{\varphi}_{\mathbf{x}}(\mathbf{x}) = \mathbf{B}^T \boldsymbol{\varphi}_{\mathbf{y}}(\mathbf{y})$, and thus:

$$\boldsymbol{\alpha}(n) = \mathbf{B}^T \boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y}(n)) \quad (5.45)$$

where $\boldsymbol{\beta}_{\mathbf{y}}$ is the SFD of \mathbf{y} . This quantity appears also in the estimation equations of PNL mixtures (see equation (6.33)). A similar quantity appears in separating CPNL mixtures, too (equation (7.14)).

5.5.3 Separating algorithm

We apply the steepest descent algorithm for estimating the parameters of the separating system.

In the instantaneous stage, we use the equivariant algorithm:

$$\mathbf{B} \leftarrow (\mathbf{I} + \mu_1 \mathbf{D}) \mathbf{B} \quad (5.46)$$

where:


$$d_{ij} = \begin{cases} 1 - y_i^2 & \text{if } i = j \\ \hat{E} \{ \psi_i(y_i) y_j \} & \text{if } i \neq j \end{cases} \quad (5.47)$$

Note that instead of zero, we have used $1 - y_i^2$ in the diagonal entries of \mathbf{D} . As it has been proposed in [96], this forces the algorithm to produce unit variance outputs and overcomes the scale indeterminacy.

For the filtering part, from (5.44), we use the algorithm:

$$\mathbf{G} \leftarrow \mathbf{G} - \mu_2 \hat{E} \{ \text{diag}(\boldsymbol{\alpha}(n)) \mathbf{E}(n) \} \quad (5.48)$$

5.5.4 Experiments

 **Experiment 5.6.** In the first experiment, we use two iid sources with uniform distribution on $(-0.5, 0.5)$. The mixing matrix is:

$$A = \begin{bmatrix} 1 & 0.5 \\ 0.3 & 1 \end{bmatrix}$$

and the sensor filters are first order low-pass filters:

$$F_1(z) = \frac{1}{1 - 0.6z^{-1}} \quad (5.49)$$


$$F_2(z) = \frac{1}{1 - 0.8z^{-1}} \quad (5.50)$$

For compensating the frequency response of the sensors, we have used the first order FIR filters:

$$F_i(z) = 1 + g_i z^{-1} \quad (5.51)$$


The parameters of the separation algorithm are: a block of observation with 100 samples, $\mu_1 = 0.1$ and $\mu_2 = 0.05$. The experiment has been repeated 50 times. The averaged values for g_1 and g_2 were -0.5954 and -0.7942 , with variances 0.0023 and 0.0008 , respectively.

Note that the optimal values for g_1 and g_2 are -0.6 and -0.8 , respectively. Consequently, it can be seen that the algorithm has been successfully found the values of g_1 and g_2 near these optimal values.

 **Experiment 5.7.** We repeat the previous experiment, but with the ill-conditioned mixing matrix:

$$A = \begin{bmatrix} 1 & 5 \\ 1 & 4 \end{bmatrix}$$

In this case, the values $\mu_1 = 0.1$ and $\mu_2 = 0.0005$ are used. Since the mixture is hard in this case, we are obliged to choose a very small μ_2 , otherwise the algorithm does not converge. The resulting estimated values for g_1 and g_2 are, in average, 0.6004 and 0.8001 , with the variances 7.1×10^{-5} and 3.7×10^{-5} , respectively. It can be seen the estimation error in estimating g_i 's is smaller in this case than the previous experiment. This is due to the fact that the separating matrix \mathbf{B} is ill-conditioned and hence is more sensitive to estimation errors in the compensating filters.

 **Experiment 5.8.** Theorem 5.1 is proved only for iid sources. It is interesting to test the robustness of the algorithm against this hypothesis. Hence, in the third experiment, we mix two non-iid sources: a sinusoidal and a triangular signal. We use 100 samples, and the mixing matrix:

$$A = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

and step sizes are $\mu_1 = 0.1$ and $\mu_2 = 0.01$. Figure 5.8 shows the output SNR's versus iteration. The results clearly show the ability of the algorithm to separate non-iid sources as well as iid sources.

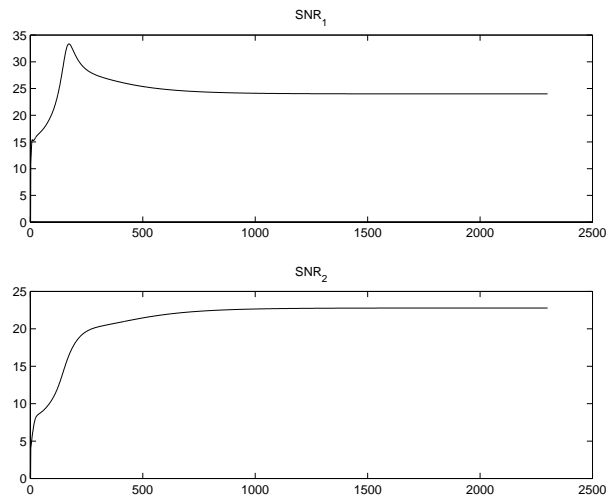


Figure 5.8: Separation of post-convolutive mixture of two non-iid sources (experiment 5.8): Output SNR's (in dB) versus iterations.

5.6 Conclusion

In this chapter, we showed how the mutual information can be used as a criterion for separating convolutive mixtures. As we knew, instantaneous independence is not sufficient for separating general convolutive mixtures, and thus the separation criterion contains the mutual information of the shifted versions of the outputs. Then, we used the gradient and projection approaches for separating convolutive mixtures. These approaches are based on minimization of the mutual information of the outputs. Consequently, they can be combined with the separation techniques used for separating PNL mixtures, to obtain a separation algorithm for CPNL mixtures. However, this combination needs some modifications in the previously known algorithms of PNL mixtures, which will be the subject of the next chapter.

In this chapter, we have also presented a special kind of convolutive mixtures called Post-convolutive mixtures. We showed that, if we use a separating structure adapted to the mixing model, we can apply a much simpler separation criterion than for the general convolutive mixtures. This property comes from the fact that, for these mixing-separating systems, the instantaneous independence of the outputs is sufficient for separating the sources.

Chapter 6

Post Non-Linear Mixtures

6.1 Introduction

In this chapter, the instantaneous Post Non-Linear (PNL) mixtures will be considered. In PNL mixtures, as it can be seen in Fig. 6.1, the mixing system is composed of a linear mixing part, followed by component-wise invertible nonlinearities. This model corresponds to the case where the mixture is itself linear but the sensors introduce the nonlinear effects (*e.g.* saturation of the amplifiers).

For separating these mixtures, we use the structure shown in Fig. 6.1, that is, we first compensate for the sensor nonlinearities (by using g_i 's), and then separate the resulting linear mixture. As we have seen in chapter 3 with this mixing-separating system, under some mild conditions, the output independence guaranties the separation of the sources. The interesting point is that the sources are separated only up to a scale and a permutation indeterminacy: even the nonlinear indeterminacy disappears (*i.e.* the sources will be obtained without any nonlinear distortion). This occurs under the condition that the sources are “really” mixed after the linear part, that is, there is at least two non-zero elements in each row or each column of the matrix \mathbf{A} .

In this chapter, except for the section 6.2, the independence criterion is the mutual

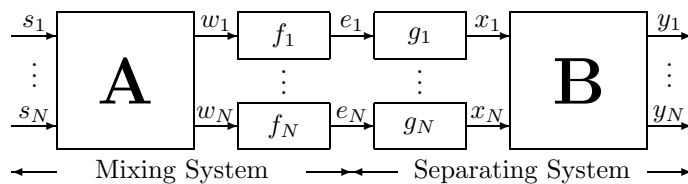


Figure 6.1: The mixing-separating system for PNL mixtures.

information of the outputs, and we will consider its minimization through gradient and projection approaches.

The mutual information minimization has been already used for separating PNL mixtures [95, 2]. However, our approach for calculating the gradients is different. In fact, the previous methods rely highly on the multiplicative relation between the output's PDF and the observation's PDF (see Section 4.4), and consequently they cannot be extended to the convolutive case. However, our approach is based on using SFD as the 'gradient' of the mutual information (Theorem 4.1), which can be easily extended to the convolutive case, too.

In this chapter, we consider 3 different methods for separating PNL mixtures. Two of them are, in fact, based on the gradient and projection approaches for minimizing the outputs' mutual information. The third one is completely different: it is a geometrical method. We first present the geometric method, because of its simplicity, and also because of the fact that it shows that the sensor nonlinearities can be compensated before separating the sources.

6.2 Geometric approach for separating PNL mixtures

Here, we consider the geometric method for separating PNL mixtures of two sources. We first state the basics of the method, then we present the algorithm, and finally we conclude with experimental results.

One limitation of this method is that it is applicable mainly for bounded sources which permit a good estimation of the borders of the scatter plot of the data (*e.g.* uniform distributions). Moreover, the method is mainly for the case of two sources and two sensors, and its generalization to higher dimensions seems tricky.

We have published this method in [12].

6.2.1 Preliminary issues

As we saw in chapter 2, Puntonet *et. al.* developed a geometrical approach for separating linear mixtures of bounded sources. The basic idea is that the scatter plot of two such independent sources forms a rectangular region in the (s_1, s_2) plane, which is transformed to a parallelogram by the linear transformation of the mixing system. The slopes of this parallelogram can determine the coefficients of the mixing matrix.

We can use a similar idea for separating PNL mixtures. Recall the separability proof of PNL mixtures which has been presented in chapter 3. Theorem 3.1 claims that the only component-wise transformation which transforms a parallelogram to another parallelogram,

is a linear transformation (provided that the borders of the parallelogram are not parallel to the coordinate axes).

Having in mind this uniqueness, and noting that, after achieving output independence, the scatter plot in the (x_1, x_2) plane must be a parallelogram, we conclude that if we find a mapping which transforms the scatter plot of observations to a parallelogram, then the sensor nonlinearities are compensated.

On the other hand, note that the Theorem 3.1 is stated only for the borders of the parallelogram. However, if we assume that the functions $h_i = g_i \circ f_i$ are monotone, the borders of a transformed region will be the images of the borders of that region (refer to the remark after the Theorem 3.1). Therefore, we can separate PNL mixtures in three steps: (i) estimating the borders of the joint plot, (ii) computing the nonlinear mappings g_1 and g_2 , which transform these curves to a parallelogram, and (iii) separating the sources from the resulting linear mixture.

Before continuing with the details of the method, we define some notations. Consider the vertical line $x_1 = c$, with values c for which two intersections occur with the borders of the joint plot. Let us denote $x_2 = \mathcal{L}_x(x_1)$ the set of intersection points with smaller coordinates x_2 (we call it the ‘lower’ boundary) and $x_2 = \mathcal{U}_x(x_1)$ the set of intersection points with larger coordinates x_2 (we call it the ‘upper’ boundary), as shown in Fig. 6.2. The corresponding curves in the (e_1, e_2) plane are denoted by $e_2 = \mathcal{L}_e(e_1)$ and $e_2 = \mathcal{U}_e(e_1)$. The upper and lower parts of the best parallelogram which fits on \mathcal{U}_x and \mathcal{L}_x are denoted by l_u and l_l , respectively (see Fig. 6.2). The corresponding curves in the (w_1, w_2) plane are denoted by $w_2 = \mathcal{L}_w(w_1)$ and $w_2 = \mathcal{U}_w(w_1)$.

Moreover, for the sake of simplicity, here we assume that the mixing matrix is in the form:

$$A = \begin{bmatrix} 1 & a \\ b & 1 \end{bmatrix} \quad (6.1)$$

where a and b are both positive. This restriction insures that the functions \mathcal{U}_e , \mathcal{L}_e , \mathcal{U}_x and \mathcal{L}_x are all invertible (see Fig. 6.3-left). Without this restriction, too, the method can be developed, but instead of working with upper and lower curves, we must consider each border piece (4 pieces), which makes the equations too complicated for stating the main points (Fig. 6.3-right).

6.2.2 Estimating the borders

First of all, we have to estimate the borders of the scatter plot of observations, that is, the curves \mathcal{L}_e and \mathcal{U}_e . For this purpose, we first divide e_1 in K regular intervals. Let $e'_1(k)$ be the mid-point of the k -th interval. Then we denote $e'_{2,L}(k)$ the minimum value of e_2 among

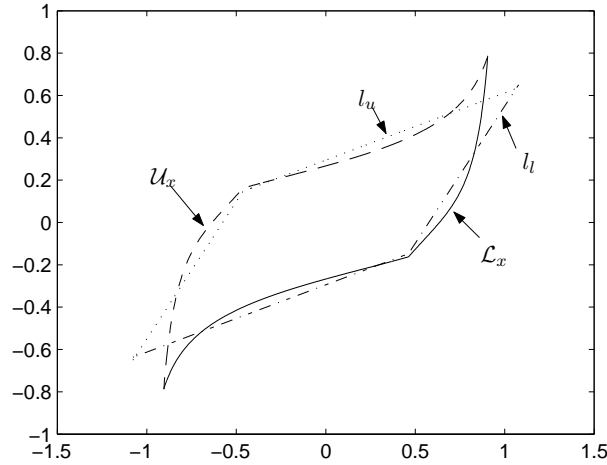


Figure 6.2: Lower and Upper curves in (x_1, x_2) plane, and the best parallelogram which fits on them. \mathcal{L}_x : Solid curve, \mathcal{U}_x : Dashed curve, l_l : Dash-dotted curve, l_u : Dotted curve.

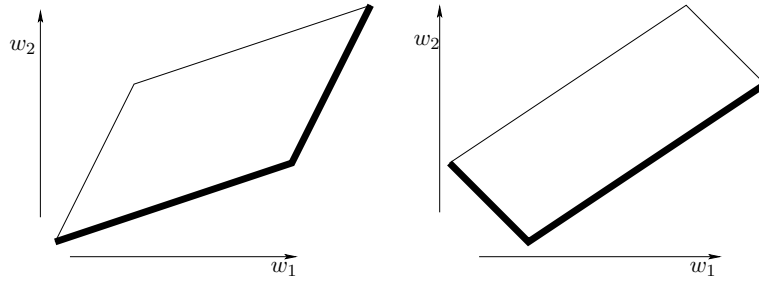


Figure 6.3: Lower and Upper curves in the (w_1, w_2) plane. Left) a and b (see equation (6.1)) are both positive and hence \mathcal{L}_w (the bold line) is invertible. Right) a is positive, b is negative. Hence \mathcal{L}_w is not invertible.

all the points which are in the corresponding strip of e_1 , and $e'_{2,U}(k)$ their maximum. Then, we estimate \mathcal{L}_e and \mathcal{U}_e as the smoothing splines fitted on the points $(e'_1(k), e'_{2,L}(k))$ and $(e'_1(k), e'_{2,U}(k))$, respectively. This choice results in error smoothing and hence improves the robustness against the errors due to the lack of data. However, the smoothing parameter of the splines (as defined in Appendix A) must be chosen close to 1, to allow modeling fast changes in the borders.

6.2.3 Compensating for the nonlinearities

In this section, we develop an iterative method for determining the nonlinear mappings which transform the boundary curves to a parallelogram. The main idea is to minimize the difference between the transformed boundary curves and the best parallelogram which fits

on them. Consequently, our objective is to find the nonlinear functions g_1 and g_2 which minimize a criterion of the form:

$$E \left\{ (\mathcal{L}_x(x_1) - l_l(x_1))^2 \right\} + E \left\{ (\mathcal{U}_x(x_1) - l_u(x_1))^2 \right\} \quad (6.2)$$

For minimizing this criterion, we use an iterative algorithm on the functions g_1 and g_2 . Hence, we need to calculate its ‘gradient’ with respect to the functions g_1 and g_2 (which are themselves some functions of x_1). However, with each modification in g_1 or g_2 , the functions l_l and l_u will be changed, which makes calculating these gradients too difficult. Consequently, we use the following approach: instead of using the ‘gradients’, we use the ‘partial gradients’, that is, in calculating the gradient of the criterion with respect to g_1 and g_2 we assume that l_l and l_u remain constant. Then, after each modification of these functions, we calculate the new functions l_l and l_u to their best values. The main loop for calculating g_1 and g_2 is then:

- Modify g_1 and g_2 by the algorithms $g_1 \leftarrow g_1 - \mu_1 \mathcal{G}_1$ and $g_2 \leftarrow g_2 - \mu_1 \mathcal{G}_2$. Here, \mathcal{G}_1 and \mathcal{G}_2 are the ‘partial gradients’ of the criterion with respect to g_1 and g_2 , that is, assuming l_l and l_u are fixed.
- Calculate l_l and l_u by finding the best parallelogram which fits on the curves \mathcal{L}_x and \mathcal{U}_x .



Note: One may wonder about the mathematical justification of this algorithm: Is this mathematically correct to assume l_l and l_u are constant while calculating the gradients? Here, with an example for functions of scalar values (not functions of functions, as in our problem), we justify this approach.

Suppose we would like to minimize a 2-variate function $f(x, y)$, and suppose that finding its minimum is not possible (or difficult) to do explicitly. Suppose also that the function is in such a form that for each fixed x , the minimum of $f(x, y)$ can be explicitly found with respect to y , and let:

$$g(x) = \underset{y}{\operatorname{argmin}} f(x, y)$$

As an example, we can consider $f(x, y) = (xe^x - y)^2$. Finding its minimum explicitly is not easy, but for each x , the function is a parabola with respect to y , and its minimum can be found explicitly.

Now, one way to find the minimum of such a function, is minimizing $f(x, g(x))$ with an iterative manner, which is equivalent to the algorithm:

$$\begin{cases} x_{n+1} = x_n - \mu \left. \frac{d}{dx} f(x, g(x)) \right|_{x=x_n} \\ y_{n+1} = g(x_{n+1}) \end{cases} \quad (6.3)$$

However, calculating $\frac{d}{dx} f(x, g(x))$ requires calculating $g'(x)$ which can be difficult (in our problem, it is difficult to calculate the gradient of l_l with respect to g_2).

One method for avoiding the computation of $g'(x)$, is to use an iteration on both variables x and y , which results in the algorithm:

$$\begin{cases} x_{n+1} = x_n - \mu_1 \frac{\partial f}{\partial x}(x_n, y_n) \\ y_{n+1} = y_n - \mu_2 \frac{\partial f}{\partial y}(x_n, y_n) \end{cases} \quad (6.4)$$

The problem with this method is that we have not used our ability to determine the exact minimum with respect to y , which results in moving on a two dimensional surface and hence higher eventuality of divergence or highly smaller rate of convergence. But, we can think about another method: replacing the iteration on y with its exact minimum, that is, the algorithm:

$$\begin{cases} x_{n+1} = x_n - \mu \frac{\partial f}{\partial x}(x_n, y_n) \\ y_{n+1} = g(x_{n+1}) \end{cases} \quad (6.5)$$

Intuitively, it seems to be a correct modification in the algorithm (6.4). However, the nice thing is that mathematically, the algorithm (6.5) is equivalent to (6.3). In fact, we have:

$$\left. \frac{d}{dx} f(x, g(x)) \right|_{x=x_n} = \frac{\partial f}{\partial x}(x_n, y_n) + \frac{\partial f}{\partial y}(x_n, y_n) g'(x_n)$$

However, as y_n is the minimizer of $f(x_n, y)$, we conclude immediately that $\frac{\partial f}{\partial y}(x_n, y_n) = 0$ and hence:

$$\left. \frac{d}{dx} f(x, g(x)) \right|_{x=x_n} = \frac{\partial f}{\partial x}(x_n, y_n)$$

Consequently, we must solve two problems: (i) Calculating the functions \mathcal{G}_1 and \mathcal{G}_2 , and (ii) Fitting a parallelogram to the curves \mathcal{U}_x and \mathcal{L}_x .

Calculating \mathcal{G}_1 and \mathcal{G}_2

For determining \mathcal{G}_2 we use the cost function (see Fig. 6.4-Right):

$$\mathcal{E}_2 = E \left\{ [\mathcal{L}_x(x_1) - l_l(x_1)]^2 \right\} + E \left\{ [\mathcal{U}_x(x_1) - l_u(x_1)]^2 \right\} \quad (6.6)$$

This criterion is in fact:

$$\begin{aligned} \mathcal{E}_2 &= E \left\{ [g_2(e_2) - l_l(g_1(e_1))]^2 \right\}_{e_2=\mathcal{L}_e(e_1)} \\ &\quad + E \left\{ [g_2(e_2) - l_u(g_1(e_1))]^2 \right\}_{e_2=\mathcal{U}_e(e_1)} \\ &= E \left\{ [g_2(e_2) - l_l(g_1(\mathcal{L}_e^{-1}(e_2)))]^2 \right\} \\ &\quad + E \left\{ [g_2(e_2) - l_u(g_1(\mathcal{U}_e^{-1}(e_2)))]^2 \right\} \end{aligned} \quad (6.7)$$

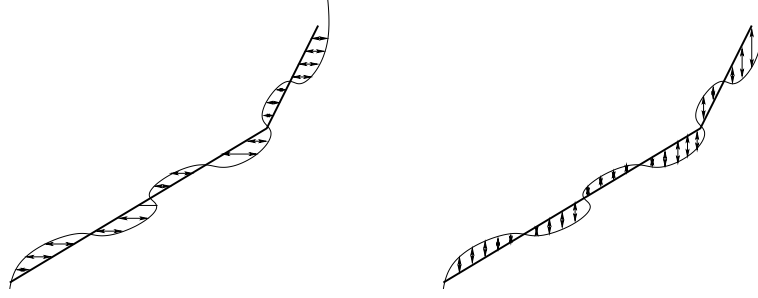


Figure 6.4: \mathcal{E}_1 and \mathcal{E}_2 error terms in estimating \mathcal{G}_1 and \mathcal{G}_2 . The bold line is l_l , and the curve is \mathcal{L}_x . Left) \mathcal{E}_1 is the horizontal error, and Right) \mathcal{E}_2 is the vertical error.

Note that, in the first equality of the above equation the expectations are calculated on the curves $e_2 = \mathcal{L}_e(e_1)$ and $e_2 = \mathcal{U}_e(e_1)$, while, in the second equality, they are taken over the whole range of e_2 , since the constraints are moved into the brackets.

Now, let $\hat{g}_2 = g_2 + \epsilon_2$, where ϵ_2 is a ‘small’ function. Direct calculations show that, up to the first order terms, we have (since l_l and l_u are assumed to be fixed):

$$\begin{aligned} \frac{\hat{\mathcal{E}}_2 - \mathcal{E}_2}{2} &= E \left\{ \left[g_2(e_2) - l_l \left(g_1 \left(\mathcal{L}_e^{-1}(e_2) \right) \right) \right] \epsilon_2(e_2) \right\} \\ &\quad + E \left\{ \left[g_2(e_2) - l_u \left(g_1 \left(\mathcal{U}_e^{-1}(e_2) \right) \right) \right] \epsilon_2(e_2) \right\} \\ &= \int_{-\infty}^{+\infty} \mathcal{G}_2(e_2) \epsilon_2(e_2) p_{e_2}(e_2) de_2 \end{aligned} \quad (6.8)$$

where $p_{e_2}(e_2)$ is the PDF of e_2 and:

$$\boxed{\mathcal{G}_2(e_2) \triangleq 2g_2(e_2) - l_l \left(g_1 \left(\mathcal{L}_e^{-1}(e_2) \right) \right) - l_u \left(g_1 \left(\mathcal{U}_e^{-1}(e_2) \right) \right)} \quad (6.9)$$

The above equation shows that the ‘gradient’ of \mathcal{E}_2 with respect to the function g_2 via the weighting function $p_{e_2}(e_2)$ can be defined as the function \mathcal{G}_2 . In other words, the schematic algorithm $g_2 \leftarrow g_2 - \mu \mathcal{G}_2$, where μ is a small positive constant, insures decreasing of \mathcal{E}_2 (provided that all the other parameters remain unchanged).

Calculating the gradient of \mathcal{E}_2 with respect to g_1 is difficult¹. Consequently, for g_1 we use the cost function (see Fig. 6.4-Left):

$$\mathcal{E}_1 = E \left\{ \left[\mathcal{L}_x^{-1}(x_2) - l_l^{-1}(x_2) \right]^2 \right\} + E \left\{ \left[\mathcal{U}_x^{-1}(x_2) - l_u^{-1}(x_2) \right]^2 \right\} \quad (6.10)$$

Similarly to what has been done for \mathcal{G}_2 , one finds that the algorithm $g_1 \leftarrow g_1 - \mu \mathcal{G}_1$ insures a reduction in \mathcal{E}_1 , where:

$$\boxed{\mathcal{G}_1(e_1) \triangleq 2g_1(e_1) - l_l^{-1} \left(g_2 \left(\mathcal{L}_e(e_1) \right) \right) - l_u^{-1} \left(g_2 \left(\mathcal{U}_e(e_1) \right) \right)} \quad (6.11)$$

¹Strictly speaking, such a gradient does not exist. This is because the functions l_l and l_u are not differentiable in their break-points.

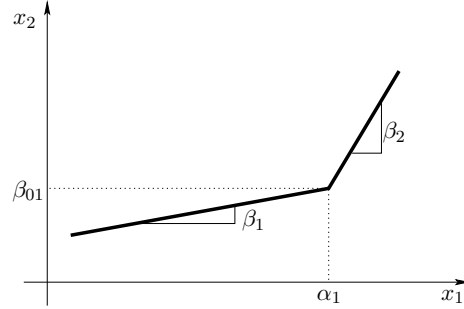


Figure 6.5: The diagram of the curve $l_l(x_1) = \beta_{01} + \beta_1(x_1 - \alpha_1)_- + \beta_2(x_1 - \alpha_1)_+$. (α_1, β_{01}) is the intersection point, β_1 is the slope of the left piece, and β_2 is the slope of the right piece.

Note that, although the criteria (6.6) and (6.10) are different, their joint minimization solves our problem. In fact, \mathcal{E}_2 is a measure of vertical error between the borders and the parallelogram, and \mathcal{E}_1 is a measure of horizontal error (as shown in Fig. 6.4).

Fitting the parallelogram

In this subsection, we deal with the problem of fitting a parallelogram to the curves \mathcal{L}_x and \mathcal{U}_x . As in the previous section, let l_l and l_u denote the lower and upper borders of the desired parallelogram. Hence, we use the following parameterization:

$$\begin{aligned} l_l(x_1) &= \beta_{01} + \beta_1(x_1 - \alpha_1)_- + \beta_2(x_1 - \alpha_1)_+ \\ l_u(x_1) &= \beta_{02} + \beta_2(x_1 - \alpha_2)_- + \beta_1(x_1 - \alpha_2)_+ \end{aligned} \quad (6.12)$$

where $(u)_- \triangleq \min(0, u)$ and $(u)_+ \triangleq \max(0, u)$. Figure 6.5 shows $l_l(x_1)$. Note that the choice of the β_1 and β_2 in the definition of the second curve (l_u), forces its segments to be parallel to the segments of l_l , and hence the two curves generate a parallelogram together.

Suppose that the curves \mathcal{L}_x and \mathcal{U}_x are known via M sample points $(x_1(k), x_{21}(k))$ and $(x_1(k), x_{22}(k))$, $k = 1, \dots, M$, where $x_{21}(k) = \mathcal{L}_x(x_1(k))$ and $x_{22}(k) = \mathcal{U}_x(x_1(k))$. We are looking for the constants $\alpha_1, \alpha_2, \beta_{01}, \beta_{02}, \beta_1$ and β_2 which minimize $\mathcal{C} = \mathcal{C}_l + \mathcal{C}_u$, where:

$$\begin{aligned} \mathcal{C}_l &= \sum_k [x_{21}(k) - l_l(x_1(k))]^2 \\ \mathcal{C}_u &= \sum_k [x_{22}(k) - l_u(x_1(k))]^2 \end{aligned} \quad (6.13)$$

Assume we know the change-points α_1 and α_2 . Then we must find $\boldsymbol{\beta} = (\beta_{01}, \beta_1, \beta_2, \beta_{02})^T$ which minimizes:

$$\mathcal{C} = \|\mathbf{x}_{21} - \mathbf{L}_1\boldsymbol{\beta}\|^2 + \|\mathbf{x}_{22} - \mathbf{L}_2\boldsymbol{\beta}\|^2 \quad (6.14)$$

where $\mathbf{x}_{21} \triangleq (x_{21}(1), \dots, x_{21}(M))^T$, $\mathbf{x}_{22} \triangleq (x_{22}(1), \dots, x_{22}(M))^T$ and:

$$\mathbf{L}_1 \triangleq \begin{bmatrix} 1 & (x_1(1) - \alpha_1)_- & (x_1(1) - \alpha_1)_+ & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & (x_1(M) - \alpha_1)_- & (x_1(M) - \alpha_1)_+ & 0 \end{bmatrix} \quad (6.15)$$

$$\mathbf{L}_2 \triangleq \begin{bmatrix} 0 & (x_1(1) - \alpha_2)_+ & (x_1(1) - \alpha_2)_- & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & (x_1(M) - \alpha_2)_+ & (x_1(M) - \alpha_2)_- & 1 \end{bmatrix} \quad (6.16)$$

Writing \mathcal{C} as:

$$\mathcal{C} = (\mathbf{L}_1 \boldsymbol{\beta} - \mathbf{x}_{21})^T (\mathbf{L}_1 \boldsymbol{\beta} - \mathbf{x}_{21}) + (\mathbf{L}_2 \boldsymbol{\beta} - \mathbf{x}_{22})^T (\mathbf{L}_2 \boldsymbol{\beta} - \mathbf{x}_{22}) \quad (6.17)$$

and using the well known equations:

$$\frac{\partial}{\partial \mathbf{x}} (\mathbf{a}^T \mathbf{x}) = \mathbf{a} \quad (6.18)$$

$$\frac{\partial}{\partial \mathbf{x}} (\mathbf{x}^T \mathbf{A}^T \mathbf{A} \mathbf{x}) = 2\mathbf{A}^T \mathbf{A} \mathbf{x} \quad (6.19)$$

we have:

$$\frac{\partial \mathcal{C}}{\partial \boldsymbol{\beta}} = 2\mathbf{L}_1^T \mathbf{L}_1 \boldsymbol{\beta} - 2\mathbf{L}_1^T \mathbf{x}_{21} + 2\mathbf{L}_2^T \mathbf{L}_2 \boldsymbol{\beta} - 2\mathbf{L}_2^T \mathbf{x}_{22} \quad (6.20)$$

Now, by letting $\partial \mathcal{C} / \partial \boldsymbol{\beta} = 0$, we obtain:

$$\boxed{\boldsymbol{\beta}_{\text{opt}} = (\mathbf{L}_1^T \mathbf{L}_1 + \mathbf{L}_2^T \mathbf{L}_2)^{-1} (\mathbf{L}_1^T \mathbf{x}_{21} + \mathbf{L}_2^T \mathbf{x}_{22})} \quad (6.21)$$

Determining the change-points α_1 and α_2 for minimizing \mathcal{C} is difficult. Instead of this, we compute the values of α_1 and α_2 which minimize \mathcal{C}_l and \mathcal{C}_u , respectively. This is done by using the Hudson's algorithm [48, 89] for fitting a segmented line to a set of data points. Although these values do not necessarily minimize \mathcal{C} , intuitively it seems a reasonable approximation, which is confirmed by experiments.

6.2.4 Separating the linear mixture

After compensating for the nonlinearities, we must separate the resulting linear mixture. Since at this point, we have already calculated the best parallelogram fitted on data, we can use the geometric linear source separation approach for separating it. The idea of geometric source separation, tells us that the separating matrix is:

$$\hat{\mathbf{B}} = \begin{bmatrix} 1 & 1/\beta_2 \\ \beta_1 & 1 \end{bmatrix}^{-1} \quad (6.22)$$


6.2.5 The algorithm

As a final note, we must limit the degree of freedom in determining the functions g_i , otherwise it can generate too varying functions, which are not invertible. In other words, we require that the functions g_1 and g_2 be somewhat smooth: if for two different data points (e_1, e_2) and (e'_1, e'_2) , e_1 and e_2 are close to each other, then the corresponding x_1 and x'_1 must be close, too. Therefore, at each iteration, we apply a smoothing procedure on these functions by using the smoothing splines. That is, at each iteration, g_i will be replaced by the smoothing spline which fits on the (e_i, x_i) data points.

On the other hand, in PNL mixtures, we have a DC and scale indeterminacy on x_i 's. To overcome these indeterminacies, at each iteration, we remove the DC of x_i 's and normalize their energies.

Figure 6.6 gives the final separating algorithm.

6.2.6 Experimental results

 **Experiment 6.1.** Here, we present the separation result for PNL mixtures of a sine and a triangle waveforms. The sources are mixed by:

$$\mathbf{A} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \quad (6.23)$$

and the sensor nonlinearities are:

$$\begin{aligned} f_1(x) &= \tanh(x) + 0.1x \\ f_2(x) &= \tanh(2x) + 0.1x \end{aligned} \quad (6.24)$$

The main parameters are: the sample size is 3,000, $M = 500$ (M is the number of points used for approximating the boundary curves), $\mu = 0.05$ and $\lambda = 0.99999$ (λ is the smoothing parameter of all the smoothing splines, used for estimating \mathcal{L}_e and \mathcal{U}_e , and for smoothing the g_i 's). Fifty intervals are used in the range of variations of e_1 for estimating \mathcal{L}_e and \mathcal{U}_e ($K = 50$).

Figure 6.7 shows the output Signal to Noise Ratios (SNR's) in dB, defined by (assuming there is no permutation):

$$\text{SNR}_i = 10 \log_{10} \frac{s_i^2}{(y_i - s_i)^2} \quad (6.25)$$

Figure 6.8 shows the scatter plot of the observations and the estimated \mathcal{L}_e and \mathcal{U}_e curves. Figure 6.9 shows the resulting estimated parallelogram in the (x_1, x_2) plane. The fluctuations in the estimated borders comes from estimation errors due to the sparse number of points close to the borders in the joint plot in the (e_1, e_2) plane. However, the experimental results show that,

- Initialization:
 1. $\mathbf{B}=\mathbf{I}$.
 2. $g_i(e_i) = e_i, i = 1, 2$.
 3. Estimate the functions \mathcal{U}_e and \mathcal{L}_e and compute them at M points $e'_{2,L}(k) = \mathcal{L}_e(e'_1(k)), e'_{2,U}(k) = \mathcal{U}_e(e'_1(k)), k = 1, \dots, M$.
- Loop:
 1. Find the best parallelogram which fits on $(g_1(e'_1(k)), g_2(e'_{2,L}(k)))$ and $(g_1(e'_1(k)), g_2(e'_{2,U}(k)))$.
 2. For $i = 1, 2$, modify the function g_i by $g_i \leftarrow g_i - \mu \mathcal{G}_i$ (See (6.9) and (6.11)).
 3. For each g_i , do a smoothing procedure (using smoothing splines), remove its DC and normalize its energy on the range of variations of its argument.
 4. Estimate \mathbf{B} by (6.22).
 5. Compute the outputs by $\mathbf{y} = \mathbf{B}\mathbf{g}(\mathbf{e})$.
- Repeat until convergence

Figure 6.6: Geometric algorithm for separating PNL mixtures.

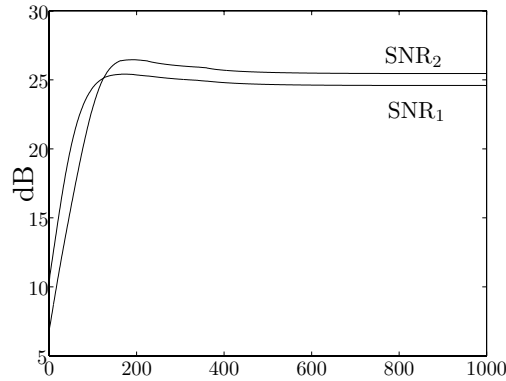


Figure 6.7: The result of geometrical approach for separating PNL mixtures: output SNR's versus iterations.

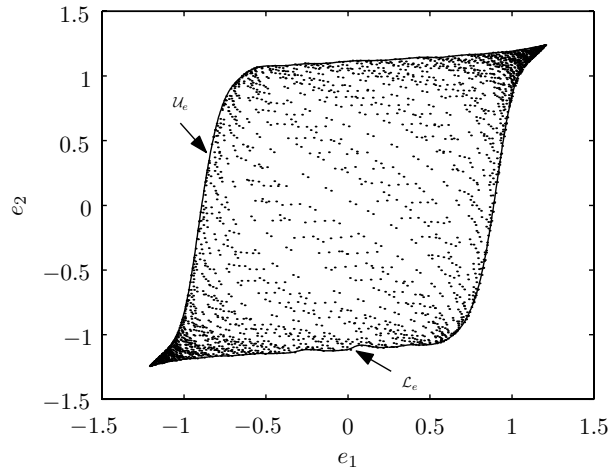


Figure 6.8: Scatter plot of observations and estimated curves \mathcal{L}_e and \mathcal{U}_e .

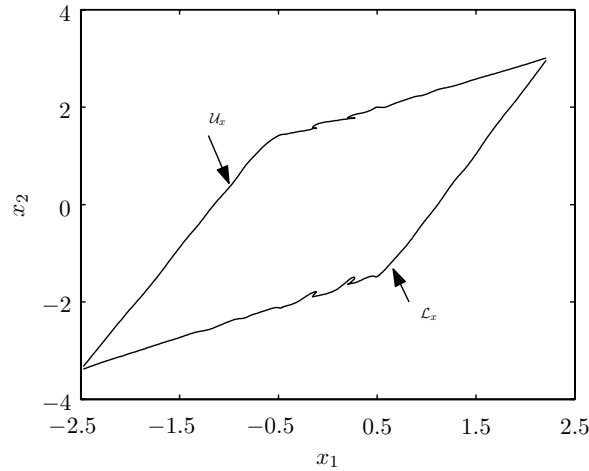
although these borders have been roughly estimated, the separating parameters are very close to the optimal ones, due to spline smoothing estimation of g_i 's.

Run time: The time required for running 100 iterations of the experiment for each realization of the sources is approximately 77 seconds.

6.3 Gradient approach

In this section, we consider the gradient approach (see Section 4.8.2) for separating PNL mixtures. The results of this section have been presented in [13].

In the gradient approach, we must first estimate the 'gradients' of $I(\mathbf{y})$ with respect to the parameters of the separating system (the matrix \mathbf{B} and the functions g_i), then use a steepest descent algorithm on these parameters for minimizing $I(\mathbf{y})$.

Figure 6.9: The curves \mathcal{L}_x and \mathcal{U}_x .

For calculating the ‘gradients’ of $I(\mathbf{y})$ with respect to the functions g_i , two different approaches can be used. One is the ‘parametric’ approach, that is, a parametric model for these functions is assumed, and then the gradients of $I(\mathbf{y})$ with respect to these parameters are calculated. In this approach, the final steepest descent gradient algorithm is applied on each parameter of the model. For example, neural networks and polynomial models have already been used for modeling g_i ’s [98, 93].

Another approach is the ‘non-parametric’ approach. In this approach, no parametric model is assumed for g_i ’s, and the ‘gradients’ of the criterion with respect to these functions, which itself is a function, is directly calculated. To clarify the idea, suppose we would like to minimize a cost function \mathcal{C} which depends on a function $g(x)$. Also, suppose that for any ‘small’ function $\epsilon(x)$ we have:

$$\mathcal{C}(g + \epsilon) - \mathcal{C}(g) = \int_{-\infty}^{+\infty} \epsilon(x) f(x) p(x) dx \quad (6.26)$$

where $p(x)$ is a non-negative function. Then, we say that the gradient of \mathcal{C} with respect to the function $g(x)$ is the function $f(x)p(x)$. Equivalently, we say that the gradient of \mathcal{C} with respect to g via the weighting function $p(x)$ is $f(x)$. The main point here is that a little movement in the opposite direction of these gradients reduces the cost function. In other words, the two algorithms $g \leftarrow g - \mu f p$ or $g \leftarrow g - \mu f$, where μ stands for a small positive constant, insure a reduction in the cost function.

6.3.1 Estimating equations

The first step for using the gradient approach, is to calculate the gradients of $I(\mathbf{y})$ with respect to the parameters of the separating system, which are \mathbf{B} and g_i ’s for a PNL separating

system.

Calculating the gradient of $I(\mathbf{y})$ with respect to the matrix \mathbf{B} leads to calculations already done in Section 4.8.2. Consequently, the desired gradient is:

$$\frac{\partial}{\partial \mathbf{B}} I(\mathbf{y}) = E \{ \boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y}) \mathbf{x}^T \} \quad (6.27)$$

and the natural gradient of $I(\mathbf{y})$ with respect to \mathbf{B} is (see (4.76)):

$$\boxed{\nabla_{\mathbf{B}} I(\mathbf{y}) = E \{ \boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y}) \mathbf{y}^T \}} \quad (6.28)$$

For calculating the gradients of $I(\mathbf{y})$ with respect to the functions g_i , suppose there is a small perturbation in these functions of the form:

$$\hat{g}_i = g_i + \epsilon_i \circ g_i \quad (6.29)$$

where ϵ_i denotes a ‘small’ function. This is equivalent to:

$$\hat{x}_i = x_i + \epsilon_i(x_i) = x_i + \delta_i \quad (6.30)$$

where $\delta_i \triangleq \epsilon_i(x_i)$. Consequently:

$$\hat{\mathbf{y}} \triangleq \mathbf{B}\hat{\mathbf{x}} = \mathbf{y} + \mathbf{B}\boldsymbol{\delta} \quad (6.31)$$

where $\boldsymbol{\delta} \triangleq (\delta_1, \dots, \delta_N)^T$. Therefore, from the theorem 4.1 we have:

$$I(\hat{\mathbf{y}}) - I(\mathbf{y}) = E \{ \boldsymbol{\delta}^T \mathbf{B}^T \boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y}) \} \quad (6.32)$$

Now we define²:

$$\boldsymbol{\alpha}(\mathbf{y}) \triangleq \mathbf{B}^T \boldsymbol{\beta}_{\mathbf{y}}(\mathbf{y}) \quad (6.33)$$

Hence:

$$\begin{aligned} I(\hat{\mathbf{y}}) - I(\mathbf{y}) &= E \{ \boldsymbol{\delta}^T \boldsymbol{\alpha}(\mathbf{y}) \} \\ &= \sum_i E \{ \epsilon_i(x_i) \alpha_i(\mathbf{y}) \} \\ &= \sum_i E \{ \epsilon_i(x_i) E \{ \alpha_i(\mathbf{y}) \mid x_i \} \} \\ &= \sum_i \int_{-\infty}^{\infty} \epsilon_i(x) E \{ \alpha_i(\mathbf{y}) \mid x_i = x \} p_{x_i}(x) dx \end{aligned} \quad (6.34)$$

Finally, from the above equation, we conclude that the (natural) gradient of $I(\mathbf{y})$ with respect to g_i and via the weighting function p_{x_i} is:

$$\boxed{(\nabla_{g_i} I)(x) = E \{ \alpha_i(\mathbf{y}) \mid x_i = x \}} \quad (6.35)$$

It must be noted that $\nabla_{g_i} I$ is itself a function.

²Recall that the same quantity appeared in (6.33) in separating post-convolutive mixtures.

6.3.2 Separating algorithm

From the gradients calculated in the previous section, we can apply the following algorithm for separating the sources:

$$\begin{cases} \mathbf{B} \leftarrow (\mathbf{I} - \mu_1 \nabla_{\mathbf{B}} I) \mathbf{B} \\ x_i \leftarrow x_i - \mu_2 (\nabla_{g_i} I)(x_i) \end{cases} \quad (6.36)$$

where the matrix $\nabla_{\mathbf{B}} I$ and the functions $(\nabla_{g_i} I)(x)$ are calculated from (6.28) and (6.35), respectively. For estimating the expectation operation in (6.28), the empirical average is used, and for estimating the conditional expectation in (6.35), we have used smoothing splines. In other words, (6.35) proposes that $(\nabla_{g_i} I)(x)$ is a regression from x_i to α_i , and for a non-parametric estimation of this regression curve, we used smoothing splines³. However, usually the points α_i with respect to x_i are too varying (their scatter plot fills a region like two “almost” independent random variables), and hence the smoothing parameter λ (as defined in Appendix A, equation (A.7)) must be small.

However, the simple algorithm (6.36) cannot separate the sources, and some other precautions are needed. First, the effects of the indeterminacies must be removed. There is scale and DC indeterminacies on both y_i 's and x_i 's. To overcome these indeterminacies, we remove their DC's and normalize their energies, at each iteration.

One other thing which needs special attention is the smoothness of the functions g_i . Since g_i must be the approximate inverse of f_i , and f_i is a relatively smooth function (it comes from the physical phenomena), g_i has to possess some degree of smoothness, too. Here, since the value of g_i at each data point $e_i(t)$ is estimated independently for different t 's, we have no constraint on the smoothness of these functions. This can result in a fluctuating function (which is non-invertible, and hence does not satisfy the separability property). Moreover, if the values $g_i(e_i(t))$ for different t 's are estimated totally independently, each value can be treated as a ‘parameter’ and hence for the two-dimensional case we must estimate $2T$ parameters from only T observations (T is the signal length), which is not mathematically well-defined. However, because of the smoothness of f_i 's and hence g_i 's, these values are not completely independent, that is, if $e_i(t_1)$ and $e_i(t_2)$ are close to each other, the values of $g_i(e_i(t_1))$ and $g_i(e_i(t_2))$ must be close, too. This problem does not exist in parametric model for estimating g_i 's, because the limited number of parameters produces a degree of smoothness in these functions.

The above paragraph proposes that a smoothness must be forced in estimating g_i 's. Our approach for creating this smoothness is using smoothing splines. That is, at each iteration,

³Another approach which could be used, was fitting a parametric curve (e.g. a polynomial, or a 10 pieces cubic spline) on the (x_i, α_i) data points.

- Initialization:
 1. $\mathbf{B}=\mathbf{I}$
 2. $\mathbf{x}=\mathbf{e}$
- Loop:
 1. Compute outputs by $\mathbf{y} = \mathbf{B}\mathbf{x}$.
 2. Estimate $\beta_{\mathbf{y}}(\mathbf{y})$ (SFD of \mathbf{y}).
 3. Let $\alpha_{\mathbf{y}}(\mathbf{y}) \triangleq \mathbf{B}^T \beta_{\mathbf{y}}(\mathbf{y})$.
 4. For $i = 1, \dots, N$, estimate $(\nabla_{g_i} I)(x_i)$, by fitting a smoothing spline on (x_i, α_i) .
 5. For $i = 1, \dots, N$, 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 $i = 1, \dots, N$, 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 (6.28).
 10. Replace the main diagonal of \mathbf{D} by $\text{diag}(\sigma_{y_1}^2 - 1, \dots, \sigma_{y_N}^2 - 1)$.
 11. Modify \mathbf{B} by $\mathbf{B} \leftarrow (\mathbf{I} - \mu_1 \mathbf{D})\mathbf{B}$.
- Repeat until convergence

Figure 6.10: The gradient approach for separating PNL mixtures.

after modifying x_i 's, we compute the smoothing spline which fits on the (e_i, x_i) points, and then replace x_i 's by the values of this smoothing spline at e_i 's. However, since this smoothing process is done at each iteration, the value of smoothing parameter λ (as defined in (A.7)) must be very close to 1. Consequently, in the convergence, there is a kind of equilibrium between the new adjustment in x_i (by the gradient algorithm), and the smoothing process which follows it. Normally, we have used values such as 0.9999 or 0.99999 for the smoothing parameter.


Figure 6.10 shows the final separating algorithm.

A note on choosing the parameters of the algorithm: The algorithm of Fig. 6.10 contains several parameters. If we use kernel estimators for estimating SFD, we have 3 smoothing parameters for the three smoothing splines in steps 2, 4 and 6; the kernel band-

width (h), and the step sizes μ_1 and μ_2 . In choosing these parameters, these guidelines may be helpful:

- h , the bandwidth of the kernel: a value about the optimum bandwidth which is given by the equation (D.6) of the Appendix D.
- λ_1 , the smoothing parameter for the spline in step 2: This spline is for estimating MSF from JSF. The value must be close to 1 to let estimating rapid changes of MSF, but not too close to 1 to do some smoothing. A value near 0.99 seems to be suitable.
- λ_2 , the smoothing parameter for the spline in step 4: This value must be rather small, because the points (x_i, α_i) normally fill a region, and we would like to calculate a conditional expectation $E\{\alpha_i | x_i\}$. This means that we need a lot of smoothing and we must think about the values such as 0.1.
- λ_3 , the smoothing parameter for the spline in step 6: This spline is for preventing the estimated g_i 's to become too varying. As this smoothing is done at each iteration, the value of the smoothing parameter must be very very close to 1; usually the values about 0.9999 or 0.99999.

6.3.3 Experimental results

 **Experiment 6.2.** In this experiment, we use two uniform random sources with zero means and unit variances. The mixing system composed of:

$$\mathbf{A} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \quad (6.37)$$

$$f_1(x) = f_2(x) = 0.1x + \tanh(2x) \quad (6.38)$$

Then we have used the separation algorithm of Fig. 6.10 with the parameters: 1000 points data block, kernel estimation of SFD with Gaussian kernels and $\lambda_1 = 0.99$ and $h = h_{\text{opt}} = 0.3047$, $\lambda_2 = 0.1$, $\lambda_3 = 0.9999$ and $\mu_1 = \mu_2 = 0.2$. This experiment has been done 100 times. For evaluating the separation quality, output SNR's has been measured which are defined by:

$$\text{SNR}_i = 10 \log_{10} \frac{s_i^2}{(y_i - s_i)^2} \quad (6.39)$$

Figure 6.11 shows the averaged output SNR's versus iteration, taken over these 100 experiments.

Run time: The time required for running 100 iterations of the experiment for each realization of the sources is approximately 40 seconds.

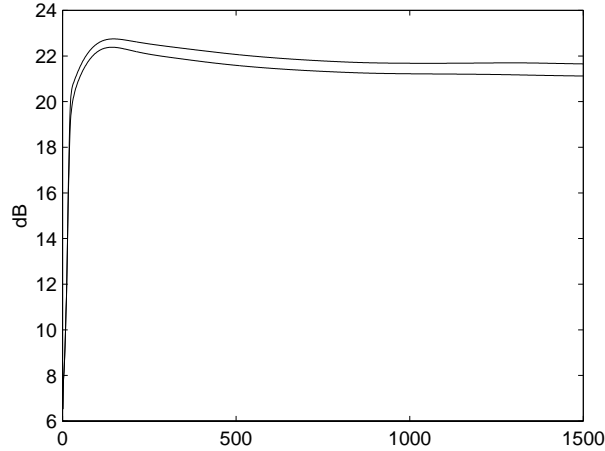


Figure 6.11: Averaged output SNR's versus iterations in separating PNL mixtures of two random sources by the gradient approach.

6.4 Projection approach

In this section, we deal with the projection approach for separating PNL mixtures. As it has been stated in section 4.8.3, this method consists in modify \mathbf{y} by $\mathbf{y} \leftarrow \mathbf{y} - \mu\beta_{\mathbf{y}}(\mathbf{y})$ at each iteration (which insures a reduction in $I(\mathbf{y})$), and then finding the system which optimally maps the observations to this newly computed outputs, and finally replacing the outputs by the actual outputs of the computed system.

6.4.1 Finding the optimal system

For using the gradient approach in a PNL separating system, the first problem which must be solved is computing the PNL separating system which optimally maps the observations to a set of *known* outputs. In other words, knowing the observations \mathbf{e} and the outputs \mathbf{y} , we are searching the functions g_i and the matrix \mathbf{B} , which minimize $E \left\{ \|\mathbf{y} - \mathbf{B}\mathbf{g}(\mathbf{e})\|^2 \right\}$. Here, we present an iterative algorithm for solving this problem.

Suppose that we know \mathbf{x} . Then, the \mathbf{B} which optimally maps \mathbf{x} to \mathbf{y} is obtained from (4.81). Knowing \mathbf{B} , we can compute \mathbf{x} again by $\mathbf{x} = \mathbf{B}^{-1}\mathbf{y}$. Then, a function g_i which map e_i to x_i can be any function, provided that it is monotonous (here we assume that it is ascending). To be sure that a g_i is ascending, we change the order of the values $x_i(1), x_i(2), \dots, x_i(T)$ in such a way that for any $e_i(k_1) > e_i(k_2)$ we have $x_i(k_1) > x_i(k_2)$. This may be better explained by its MATLAB code:


```
[temp, index_i] = sort(e_i); % Outside the main loop.
x_i(index_i) = sort(x_i);    % Within the main loop, as the step 3.
```

- Initialization: $\mathbf{x} = \mathbf{e}$.
- Loop:
 1. Let $\mathbf{B} = E\{\mathbf{y}\mathbf{x}^T\} (E\{\mathbf{x}\mathbf{x}^T\})^{-1}$.
 2. Let $\mathbf{x} = \mathbf{B}^{-1}\mathbf{y}$.
 3. For $i = 1, \dots, N$, 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 6.12: Finding the optimal PNL mapping.

This time permutation insures that the function $g_i : e_i \mapsto x_i$ is ascending. It must be noted that we are using a steepest descent iterative algorithm and our initial value for g_i is ascending. Moreover, at each iteration, only a rather small modification is done in the values of $x_i(t)$. Consequently, the above time permutation does not result in a huge modification of the estimated g_i ; it must be seen as a manner for preventing the algorithm to produce a non-ascending g_i .

After this new estimation of \mathbf{x} , we compute a new \mathbf{B} and then the loop is repeated. This results in the iterative algorithm of Fig. 6.12 for calculating g_i 's and \mathbf{B} .

 **Experiment 6.3.** To verify the efficacy of this algorithm, we mix two zero mean and unit variance sources by the mixing system:

$$\mathbf{A} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \quad (6.40)$$

$$f_1(x) = f_2(x) = 0.1x + \tanh(2x) \quad (6.41)$$

and we let $\mathbf{y} = \mathbf{s}$. Then we apply the algorithm of Fig. 6.12 for obtaining g_i 's and \mathbf{B} . We also define the estimation error by $E\{\|\mathbf{y} - \mathbf{B}\mathbf{g}(\mathbf{e})\|^2\}$. Evidently, the optimal error is zero and is obtained for $g_i = f_i^{-1}$ and $\mathbf{B} = \mathbf{A}^{-1}$. Figure 6.13 shows the variations of this error term versus iterations.

This experiment shows that the algorithm of Fig. 6.12 converges very fast, essentially after 2 or 3 iterations.

6.4.2 Separating algorithm

Now, the final separation algorithm is evident: first, modifying \mathbf{y} by $\mathbf{y} \leftarrow \mathbf{y} - \mu\beta_{\mathbf{y}}(\mathbf{y})$, then finding g_i 's and \mathbf{B} by the algorithm of Fig. 6.12, and finally replacing \mathbf{y} by $\mathbf{B}\mathbf{g}(\mathbf{e})$; and

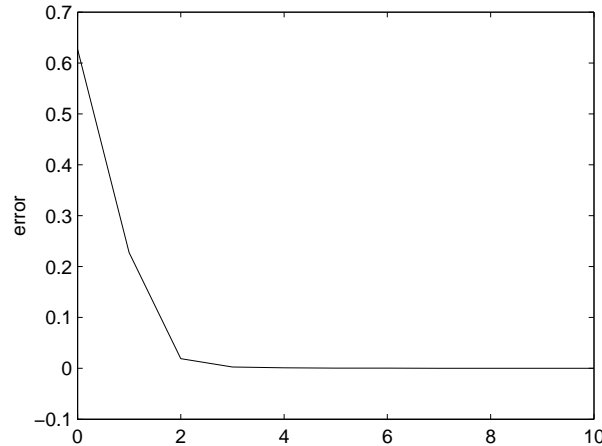



Figure 6.13: Estimation error in finding best PNL mapping which maps \mathbf{x} to \mathbf{s} .

repeating this procedure until convergence.

However, some precautions are required. As in section 6.3.2, we need to remove the DC and normalize the energy of the outputs and of the x_i 's at each iteration. We also must apply a smoothing process on the estimated g_i 's at each iteration. Moreover, two modifications must be done in applying the algorithm of Fig. 6.12. First, instead of initializing by $\mathbf{x} = \mathbf{e}$, we can use the value of \mathbf{x} obtained in the previous iteration which is a better initial estimation of \mathbf{x} . Secondly, we do not wait for the projection algorithm to converge (even, it is possible that it does not converge, when the outputs cannot be expressed as $\mathbf{B}\mathbf{g}(\mathbf{e})$). Instead, we simply repeat the loop for some fixed number of iterations, say 5 or 10 (in fact, even 1 iteration seems sufficient in many cases, because the whole algorithm is itself iterative, and we use the value of \mathbf{x} in the previous iteration for its initial value of this iteration).

The final separating algorithm is shown in Fig. 6.14. The value of K in this algorithm (number of repetitions of the internal loop), is a small number, *e.g.* 1 to 10.

6.4.3 Experimental results

 **Experiment 6.4.** We repeat the experiment 6.2 using the projection approach. In this experiment, we use two uniform random sources with zero means and unit variances. The mixing system composed of:

$$\mathbf{A} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \quad (6.42)$$

$$f_1(x) = f_2(x) = 0.1x + \tanh(2x) \quad (6.43)$$

The parameters of the separating system are: $\mu = 0.1$, 1000 samples data block, Pham's estima-

- Initialization: $\mathbf{y} = \mathbf{x} = \mathbf{e}$.
- Loop:
 1. Estimate $\beta_{\mathbf{y}}(\mathbf{y})$, the SFD of \mathbf{y} .
 2. Modify the outputs by $\mathbf{y} \leftarrow \mathbf{y} - \mu\beta_{\mathbf{y}}(\mathbf{y})$.
 3. For $k = 1, \dots, K$, do:
 - (a) Let $\mathbf{B} = E\{\mathbf{y}\mathbf{x}^T\} (E\{\mathbf{x}\mathbf{x}^T\})^{-1}$.
 - (b) Let $\mathbf{x} = \mathbf{B}^{-1}\mathbf{y}$.
 - (c) For $i = 1, \dots, N$, change the order of $x_i(k)$ such that the function $x_i = g_i(e_i)$ be ascending.
 4. For $i = 1, \dots, N$, remove the DC of x_i and normalize its energy.
 5. For $i = 1, \dots, N$, 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 $i = 1, \dots, N$, remove the DC of y_i and normalize its energy.
- Repeat until convergence

Figure 6.14: Projection algorithm for separating PNL mixtures.

tion method for estimating the SFD, $K = 5$, and the smoothing parameter used for smoothing g_i 's is 0.999. Figure 6.15 shows the averaged output SNR's (versus iteration) taken over 100 runs of the algorithm.

Run time: The time required for running 100 iterations of the experiment for each realization of the sources is approximately 12.5 seconds.

Comment: It can be seen in both figures 6.11 and 6.15, that the output SNR's reach to a maximum value and then fall a little. In my opinion, one of the reasons of this phenomena is due to the 'finite' data set. That is, after finding a solution, the algorithms continue to modify the nonlinearities to decrease the criterion which is only an estimation of the actual one (because of the limited number of samples). And finally, the algorithm converges because of the the smoothing process done at each iteration (it cannot produce too fluctuating nonlinearities). For example, Fig. 6.16 shows the distributions of \mathbf{w} and \mathbf{e} for one realization of the above experiment. Figure 6.17 represents the output SNR's for this realization after 50 and after 1200 iterations. Figures 6.18 and 6.19 show the distributions of \mathbf{x} and \mathbf{y} and the compensated functions $g_i \circ f_i$ after 50 and 1200 iterations. It can be seen, that the bad estimation of g_i 's comes from the lack of data (e_i) in the middle of the range (see Fig. 6.16-b).

Of course, one solution for this problem, is to use a larger data block. However, since our

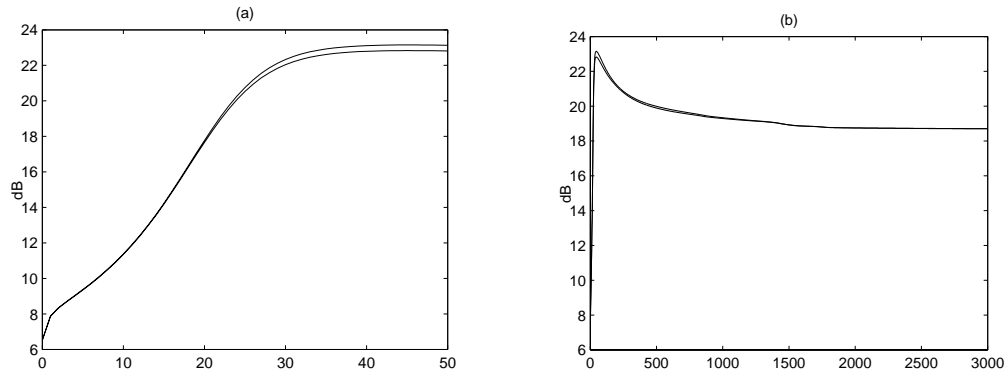


Figure 6.15: Averaged output SNR's versus iterations in separating PNL mixture by using the projection approach, a) After 50 iterations, b) After 2500 iterations.

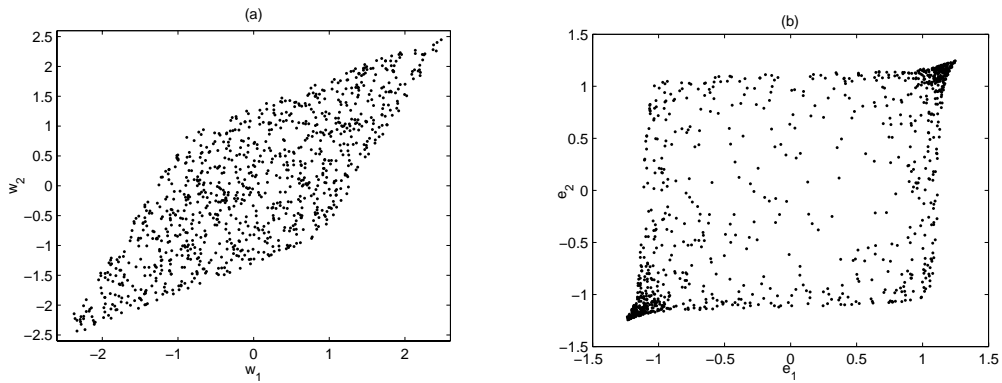



Figure 6.16: Distribution of two mixed random sources: a) Before nonlinear distortion, b) After nonlinear distortion.

estimation algorithms for estimating SFD are block oriented, this dramatically slows down the algorithm. Another better approach, is using a semi-adaptive approach. That is, after each iteration (or after a few iterations), we change the data block and work with another (the next) frame of data. For example, in the projection algorithm of Fig. 6.14, before doing the step 6, we take another block of observations. With this approach, the number of observations are virtually infinite, without slowing down the algorithm.

 **Experiment 6.5.** In this experiment, we have used two zero mean and unit variance random sources with uniform distributions and the same mixing system as that of experiment 6.2. A block of data with 1000 samples and the kernel estimation of SFD have been used, but before doing the step 6 of the algorithm of the Fig. 6.14, we change the data block. Figure 6.20 shows the output SNR's and compensated nonlinearities after 2500 iterations.

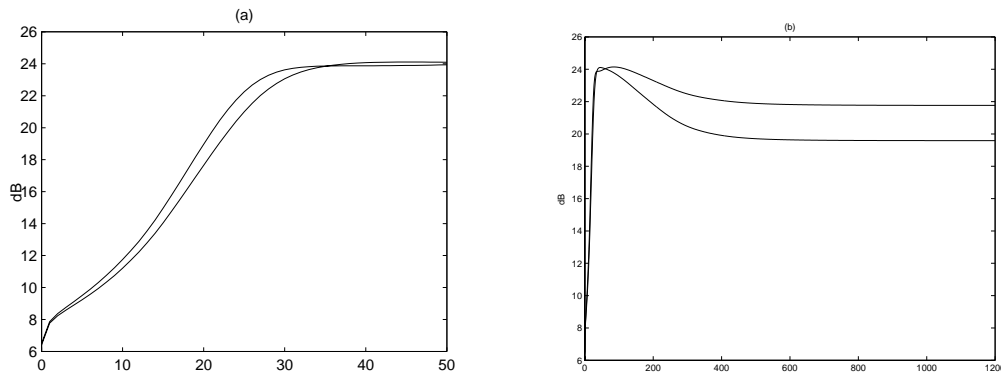


Figure 6.17: Output SNR's versus iterations, a) After 50 iterations, b) After 1200 iterations.

The above experiment confirms the opinion that one reason for the SNR decreasing after reaching a maximum is due to too small data blocks. However, if we repeat the above experiment with the Pham's SFD estimator (not presented here), the problem does not disappear. Consequently, the phenomena is also related to the quality of the estimation of the SFD.

6.5 Conclusion

In this chapter, we considered PNL mixtures, and presented three different approaches for their separation. The first approach was a geometric approach, which proves that compensating nonlinearities before separating the sources is possible. This approach is limited to the two dimensional case, and with the bounded sources that allows a good estimation of the borders of their joint scatter plot. The two other approaches was based on SFD, the gradient of the mutual information. We have used SFD of the outputs for designing gradient based algorithm for minimizing their mutual information.

As for the linear mixtures, using SFD for minimizing the mutual information of the outputs, does not rely on the multiplicative relation between \mathbf{x} and \mathbf{y} . Consequently, it is possible to extend the results to convolutive mixtures, which is the subject of the next chapter.

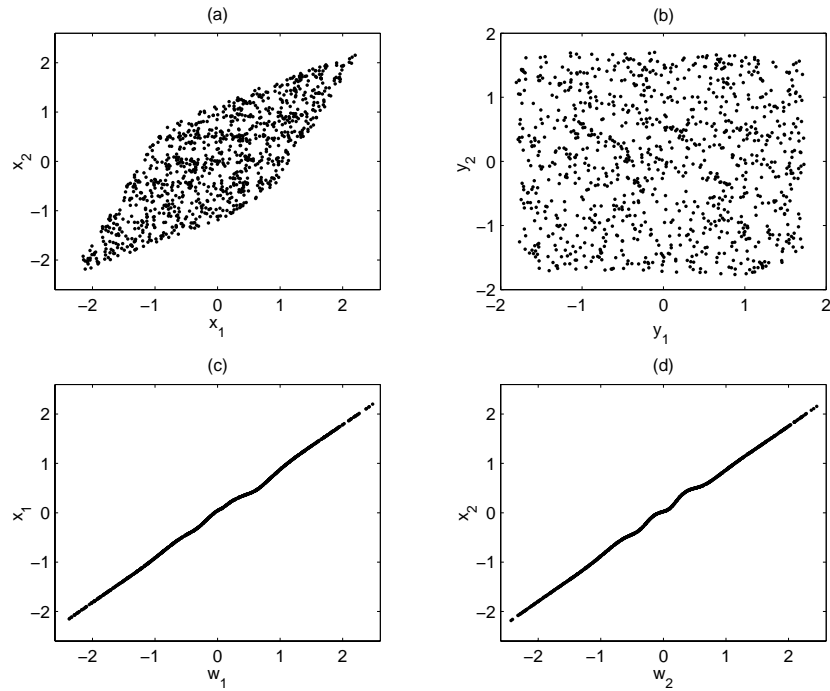


Figure 6.18: Separation result after 50 iterations: a) Distribution of \mathbf{x} , b) Output distribution, c) $g_1 \circ f_1$, and d) $g_2 \circ f_2$.

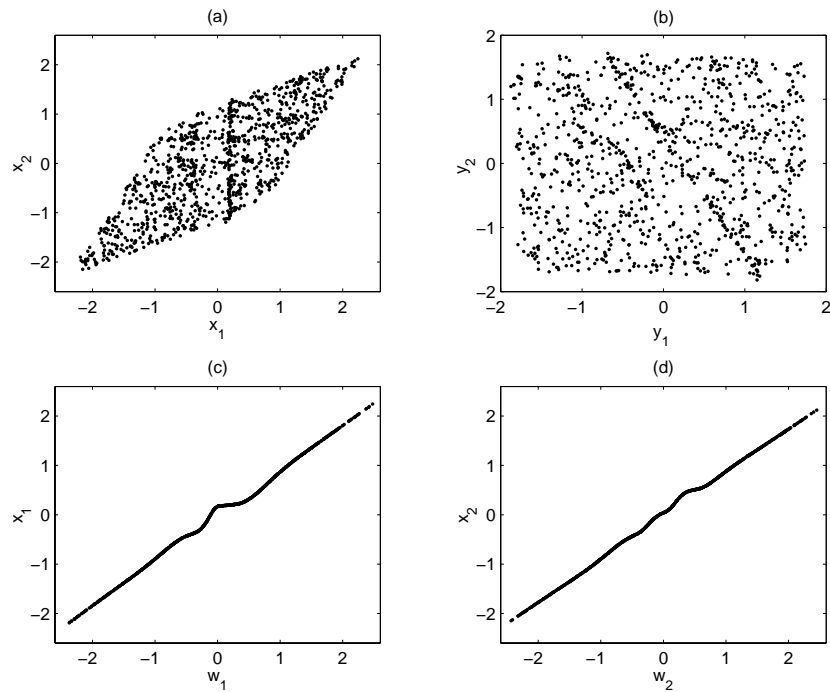


Figure 6.19: Separation result after 1200 iterations: a) Distribution of \mathbf{x} , b) Output distribution, c) $g_1 \circ f_1$, and d) $g_2 \circ f_2$.

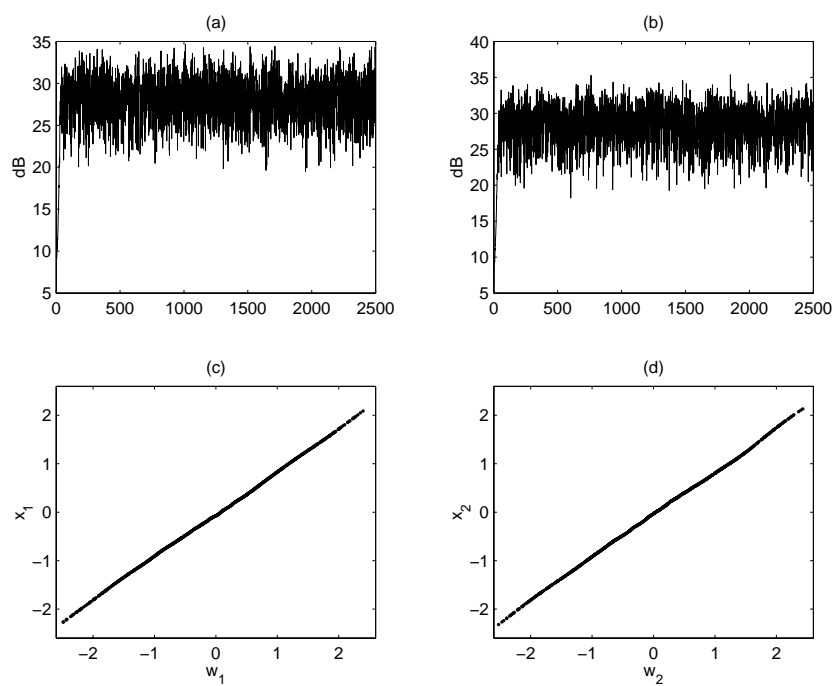


Figure 6.20: Separation result after 2500 iterations using framing: a) SNR_1 , b) SNR_1 , c) $g_1 \circ f_1$, and d) $g_2 \circ f_2$.

Chapter 7

CPNL Mixtures

7.1 Introduction

In this chapter, the Convolutional Post Non-Linear (CPNL) mixtures are considered. In fact, having prepared the basics of the method (criterion, estimation equations, ...) in the previous chapters, we have simply to combine the results of the chapters 5 and 6 for separating these mixtures.

In CPNL mixtures, the sources are mixed by a linear convolutional system followed by componentwise nonlinearities, as shown in the Fig. 7.1. This mixing model corresponds to the case where the mixing system is itself convolutional-linear, but the sensors have nonlinear effects (such as saturation). Taking into account the mixing structure, we construct the separating structure as its mirror structure. Hence, we first compensate for the nonlinearities (using the functions g_i) and then separate the resulting linear (but convolutional) mixture by the FIR separating filter:

$$\mathbf{B}(z) = \mathbf{B}_0 + \mathbf{B}_1 z^{-1} + \dots + \mathbf{B}_p z^{-p} \quad (7.1)$$

In Section 3.5, we justified the separability of CPNL mixtures (for iid sources) by extending the separability result of the instantaneous PNL mixtures. In this chapter, we use

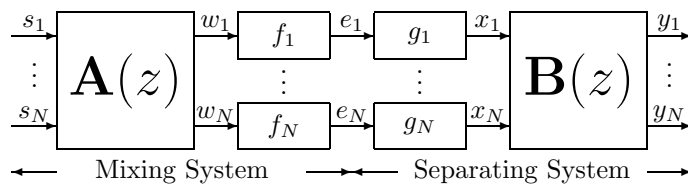


Figure 7.1: The mixing-separating system for CPNL mixtures.

the gradient and projection approaches (chapter 4) for separating them.

As for the convolutive mixtures, we consider only the case of two sources and two sensors, and we use the following independence criterion:

$$J = \sum_{m=-M}^M I(y_1(n), y_2(n-m)) \quad (7.2)$$

where I denotes the mutual information. For minimizing this criterion, we choose randomly a different m at each iteration, and then apply a gradient or projection approach for minimizing $I(y_1(n), y_2(n-m))$.

Principally, it seems that the results of this chapter can be extended to more than two sources, by using a criterion in the form $\sum_{m_1, \dots, m_{N-1}} I(y_1(n), y_2(n-m_1), \dots, y_N(n-m_{N-1}))$. However, such a generalization results in a huge computational volume and requires a lot of data. Hence, we had not any experience with higher than two sources, and in this chapter we remain in the case of two sources and two sensors.

7.2 Gradient approach

In this approach, we must first calculate the gradients of $I(y_1(n), y_2(n-m))$ with respect to the parameters of the separating system, that is, with respect to the functions g_i and to the matrices \mathbf{B}_i , and then apply a steepest descent algorithm on each parameter. Here, we first calculate these gradients, and after developing the separating algorithm we present some experimental results.

The results of this section have been presented in [8].

7.2.1 Estimating equations

Using the calculations of the section 5.3.1, we have the gradients of $I(y_1(n), y_2(n-m))$ with respect to the matrices \mathbf{B}_k :

$$\boxed{\frac{\partial}{\partial \mathbf{B}_k} I(y_1(n), y_2(n-m)) = E \{ \boldsymbol{\beta}_m(n) \mathbf{x}(n-k)^T \}, \quad k = 0, \dots, p} \quad (7.3)$$

where:

$$\boldsymbol{\beta}_m(n) \triangleq \boldsymbol{\beta}_{\mathbf{y}^{(m)}}^{(-m)}(n) = \begin{bmatrix} \boldsymbol{\beta}_{\mathbf{y}^{(m)},1}(n) \\ \boldsymbol{\beta}_{\mathbf{y}^{(m)},2}(n+m) \end{bmatrix} \quad (7.4)$$

and $\boldsymbol{\beta}_{\mathbf{y}^{(m)}}(\cdot)$ denotes the SFD of $\mathbf{y}^{(m)}$ (see the notation introduced in equation (5.10), page 62). In other words:

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

Now, we must calculate the gradients of $I(y_1(n), y_2(n-m))$ with respect to the functions g_1 and g_2 . As in section 6.3.1, we assume that each g_i is modified by the composition with a ‘small’ function ϵ_i :

$$\hat{g}_i = g_i + \epsilon_i \circ g_i \quad (7.5)$$

This is equivalent to:

$$\hat{x}_i = x_i + \epsilon_i(x_i) = x_i + \delta_i \quad (7.6)$$

where $\delta_i \triangleq \epsilon_i(x_i)$. Consequently:

$$\hat{\mathbf{y}}(n) \triangleq [\mathbf{B}(z)] \hat{\mathbf{x}}(n) = \mathbf{y}(n) + [\mathbf{B}(z)] \boldsymbol{\delta}(n) \quad (7.7)$$

where $\boldsymbol{\delta}(n) \triangleq (\delta_1(n), \dots, \delta_N(n))^T$. Defining $\boldsymbol{\Delta}(n) \triangleq [\mathbf{B}(z)] \boldsymbol{\delta}(n)$ we have:

$$\hat{\mathbf{y}}(n) = \mathbf{y}(n) + \boldsymbol{\Delta}(n) \quad (7.8)$$

and hence:

$$\hat{\mathbf{y}}^{(m)}(n) = \mathbf{y}^{(m)}(n) + \boldsymbol{\Delta}^{(m)}(n) \quad (7.9)$$

Now, from Theorem 4.1 we can write:

$$I(\hat{\mathbf{y}}^{(m)}(n)) - I(\mathbf{y}^{(m)}(n)) = E \left\{ \boldsymbol{\Delta}^{(m)}(n)^T \boldsymbol{\beta}_{\mathbf{y}^{(m)}}(n) \right\} \quad (7.10)$$

where $\boldsymbol{\beta}_{\mathbf{y}^{(m)}}(n)$ stands for $\boldsymbol{\beta}_{\mathbf{y}^{(m)}}(\mathbf{y}^{(m)}(n))$. Assuming that all the sources are stationary, the above relation can be simplified as follows:

$$\begin{aligned} I(\hat{\mathbf{y}}^{(m)}(n)) - I(\mathbf{y}^{(m)}(n)) &= E \left\{ \boldsymbol{\Delta}^{(m)}(n)^T \boldsymbol{\beta}_{\mathbf{y}^{(m)}}(n) \right\} \\ &= E \left\{ \Delta_1(n) \boldsymbol{\beta}_{\mathbf{y}^{(m)},1}(n) \right\} + E \left\{ \Delta_2(n-m) \boldsymbol{\beta}_{\mathbf{y}^{(m)},1}(n) \right\} \\ &= E \left\{ \Delta_1(n) \boldsymbol{\beta}_{\mathbf{y}^{(m)},1}(n) \right\} + E \left\{ \Delta_2(n) \boldsymbol{\beta}_{\mathbf{y}^{(m)},1}(n+m) \right\} \\ &= E \left\{ \boldsymbol{\Delta}^T(n) \boldsymbol{\beta}_m(n) \right\} \end{aligned} \quad (7.11)$$

where $\boldsymbol{\beta}_m(n)$ is defined in (7.4). Now, from:

$$\boldsymbol{\Delta}(n) \triangleq [\mathbf{B}(z)] \boldsymbol{\delta}(n) = \sum_{k=0}^p \mathbf{B}_k \boldsymbol{\delta}(n-k) \quad (7.12)$$

we have:

$$\begin{aligned} I(\hat{\mathbf{y}}^{(m)}(n)) - I(\mathbf{y}^{(m)}(n)) &= \sum_{k=0}^p E \left\{ \boldsymbol{\delta}^T(n-k) \mathbf{B}_k^T \boldsymbol{\beta}_m(n) \right\} \\ &= \sum_{k=0}^p E \left\{ \boldsymbol{\delta}^T(n) \mathbf{B}_k^T \boldsymbol{\beta}_m(n+k) \right\} \\ &= E \left\{ \boldsymbol{\delta}^T(n) \boldsymbol{\alpha}(n) \right\} \end{aligned} \quad (7.13)$$

where (compare this definition with (6.33)):

$$\boldsymbol{\alpha}(n) \triangleq \sum_{k=0}^p \mathbf{B}_k^T \boldsymbol{\beta}_m(n+k) = \left[\mathbf{B}^T \begin{pmatrix} 1 \\ z \end{pmatrix} \right] \boldsymbol{\beta}_m(n) \quad (7.14)$$

Now, the simplification (6.34) can be repeated here, from where we obtain:

$$\boxed{(\nabla_{g_i} I)(x) = E \{ \alpha_i \mid x_i = x \} \quad i = 1, 2} \quad (7.15)$$

This relation, proposes that $\nabla_{g_i} I$ can be obtained by a regression from x_i to α_i .



Note: Since the nonlinearities are instantaneous, they cannot result in time delay of the outputs. Hence, we can use the results of section 6.3.1, that is, equations (6.33) and (6.35) for the gradient with respect to g_i 's. With this method, we would use two different independence criterion, $\sum_m I(y_1(n), y_2(n-m))$ for the convolutive part, and $I(\mathbf{y}(n))$ for the nonlinear part.


7.2.2 Separating algorithm

Having the gradients (7.3) and (7.15), we can apply a steepest descent gradient algorithm on each parameter for separating the sources. However, the indeterminacies require some attention.

Since there are a DC and scale indeterminacies on each x_i and each y_i , we have to remove the DC and normalize the energies at each iteration. Moreover, to prevent the algorithm from generating too varying functions, we apply a smoothing process on them at each iteration (using smoothing splines). We also change the data set at each iteration. Finally, as for the convolutive mixtures, at each iteration we use a randomly chosen m between $-M$ and M . We set $M = 2p$, where p denotes the maximum degree of the separating filters.

Figure 7.2 shows the final separating algorithm.

7.2.3 Experimental results

 **Experiment 7.1.** In this experiment, we use zero-mean unit-variance random sources with uniform distributions. The sources are mixed by the convolutive mixing matrix:

$$\mathbf{A}(z) = \begin{bmatrix} 1 + 0.2z^{-1} + 0.1z^{-2} & 0.5 + 0.3z^{-1} + 0.1z^{-2} \\ 0.5 + 0.3z^{-1} + 0.1z^{-2} & 1 + 0.2z^{-1} + 0.1z^{-2} \end{bmatrix} \quad (7.16)$$

followed by nonlinearities:

$$f_1(x) = f_2(x) = \tanh(2x) + 0.1x \quad (7.17)$$

- 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 $\boldsymbol{\beta}^*(n)$, the SFD of $(y_1(n), y_2(n-m))^T$.
 3. Let $\boldsymbol{\beta}_m(n) = (\beta_1^*(n), \beta_2^*(n+m))^T$.
 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 $k = 0 \dots p$:

$$\mathbf{B}_k \leftarrow \mathbf{B}_k - \mu_2 \hat{E} \{ \boldsymbol{\beta}_m(n) \mathbf{x}(n-k)^T \}$$
 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 row of $\mathbf{B}(z)$ by σ_i .
- Repeat until convergence.

Figure 7.2: Gradient algorithm for separating CPNL mixtures.

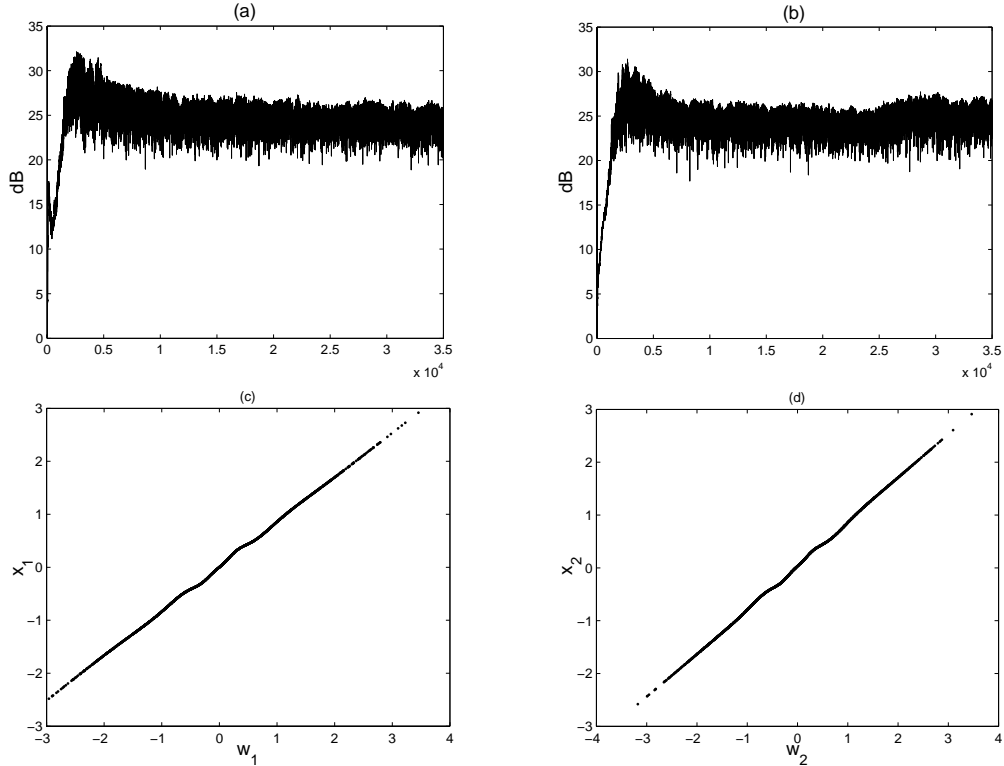


Figure 7.3: Result of CPNL mixture separation by gradient approach (experiment 7.1): a) SNR_1 versus iterations, b) SNR_2 versus iterations, c) $g_1 \circ f_1$, and d) $g_2 \circ f_2$.


For separating this mixture, we use the algorithm of Fig. 7.2, with the parameters: $\mu_1 = \mu_2 = 0.2$, $p = 2$ (second order separating filters), $M = 4$, Pham's estimator for estimating SFD, $\lambda = 0.1$ for the smoothing splines of step 4 of the algorithm, $\lambda = 0.99999$ for the smoothing splines applied for estimating g_i 's (step 6 of the algorithm). At each iteration a different data block of length 2000 is used. In the program, the data block is changed before the step 10 of the Fig. 7.2.

The quality of separation is estimated by the output Signal to Noise Ratios (SNR's), defined by (assuming there is no permutation):

$$\text{SNR}_i = 10 \log_{10} \frac{E \{y_i^2\}}{E \left\{ (y_i|_{s_i=0})^2 \right\}} \quad (7.18)$$

where $y_i|_{s_i=0}$ stands for the signal on the i -th output, when the i -th input is zero (resulting from the leakage of the other sources).

Run time: The time required for running 100 iterations of the experiment for each realization of the sources is approximately 36 seconds.

 **Experiment 7.2.** The experiment 7.2 is repeated, but with blocks of length 1000

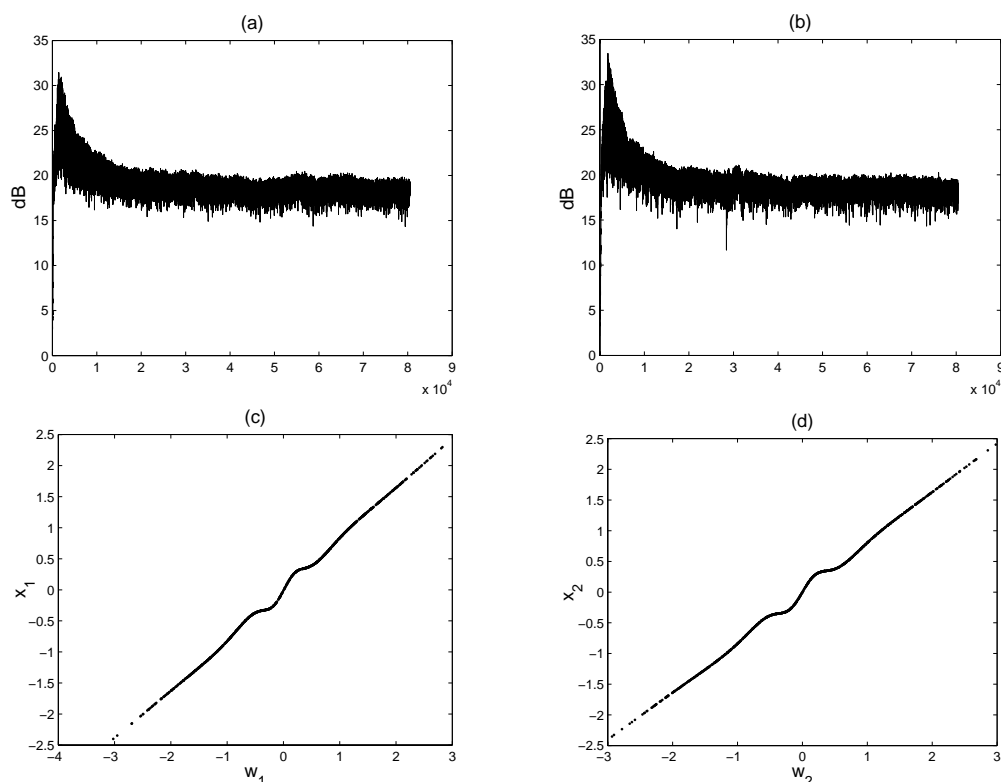



Figure 7.4: Separation result of separating a CPNL mixture by gradient approach (experiment 7.2): a) SNR_1 versus iterations, b) SNR_2 versus iterations, c) $g_1 \circ f_1$, and d) $g_2 \circ f_2$.

samples, instead of 2000 samples. The separation result can be seen in Fig. 7.4.

This experiment shows that although we change the data block at each iteration, to prevent the phenomena of falling SNR's after reaching a maximum (which is discussed in the page 99), but still the length of the data block is an important factor. It can be seen that the phenomena is repeated here. Another reason is using the Pham's SFD estimator for the nonlinear part (which has the same effect in PNL mixtures, too).

Run time: The time required for running 100 iterations of the experiment for each realization of the sources is approximately 17 seconds.

 **Experiment 7.3.** In previous chapters, we saw that Pham's estimator acts better for convolutive mixture while kernel estimator works better for PNL mixtures. In this experiment, we repeat the previous experiment, but with two different SFD estimation: kernel estimation for the nonlinear part, and Pham's method for the convolutive part. Moreover, as the nonlinear part cannot result in a time delay in signals, we used $m = 0$ for the nonlinear part. All the other parameters are left unchanged. Figure 7.5 shows the final separation result.

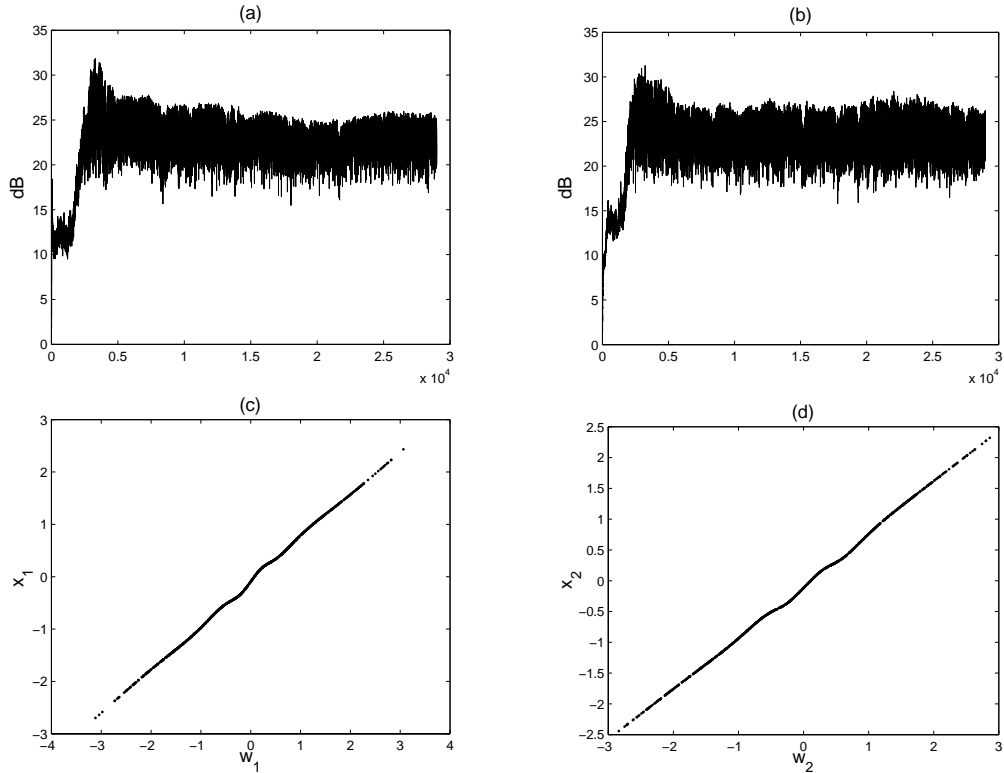


Figure 7.5: Separation result of separating a CPNL mixture by gradient approach (experiment 7.3): a) SNR_1 versus iterations, b) SNR_2 versus iterations, c) $g_1 \circ f_1$, and d) $g_2 \circ f_2$.

Comparison of the results of this experiment with the previous one (Fig. 7.5) shows the efficiency of using two different SFD estimation for convolutive and nonlinear parts.

Run time: The time required for running 100 iterations of the experiment for each realization of the sources is approximately 54 seconds.

7.3 Projection approach

In this section, we use the projection approach for separating CPNL mixtures. As it has been explained in the section 4.8.3, the basic steps of this approach are: i) modifying the outputs by $\mathbf{y} \leftarrow \mathbf{y} - \mu\beta_{\mathbf{y}}(\mathbf{y})$ which decreases their mutual information, ii) finding the best system which maps the observations onto these new outputs, iii) replacing the outputs by the actual outputs of this computed system, and repeating this procedure.

- Initialization: $\mathbf{x}(n) = \mathbf{e}(n)$.
- Loop:
 1. Using (5.29), find the best $\mathbf{B}(z)$ which maps $\mathbf{x}(n)$ to $\mathbf{y}(n)$.
 2. Compute $\mathbf{x}(n) = [\text{inv}(\mathbf{B}(z))] \mathbf{y}(n)$ using the recursive equation (7.20).
 3. For $i = 1, \dots, N$, change the order of $x_i(k)$ such that the function $x_i = g_i(e_i)$ become ascending.
- Repeat until convergence

Figure 7.6: Finding the optimal CPNL mapping.

7.3.1 Finding the optimal system

The first problem which must be solved for using the projection approach, is finding the best system which maps the known observations to the known outputs. In other words, knowing $\mathbf{e}(n)$ and $\mathbf{y}(n)$, we are looking for the nonlinearities g_i 's and the filter $\mathbf{B}(z)$ which minimize $E \left\{ \|\mathbf{y}(n) - [\mathbf{B}(z)] \mathbf{g}(\mathbf{e}(n))\|^2 \right\}$.

For finding this optimal system, we use an iterative approach similar to the one presented in the previous chapter (see Fig. 6.12). However, as the linear part is convolutive, this approach needs two modifications.

A modification is done in the step 1 of the algorithm of Fig. 6.12. In this step, the best instantaneous linear system which maps \mathbf{x} to \mathbf{y} was estimated by the simple relation $\mathbf{B} = E \{ \mathbf{y} \mathbf{x}^T \} (E \{ \mathbf{x} \mathbf{x}^T \})^{-1}$. Now, the system to be estimated is a convolutive system $\mathbf{B}(z)$, and one has to use the method presented in the section 5.4.1 for convolutive mixtures and mainly the equation (5.29).

The second modification occurs in the second step of Fig. 6.12. In this step, knowing the linear system and its output, the input of the system must be estimated. Previously, the linear system being instantaneous, this was simply achieved by $\mathbf{x} = \mathbf{B}^{-1} \mathbf{y}$. Now, the system is a filter, and:

$$\mathbf{y}(n) = \mathbf{B}_0 \mathbf{x}(n) + \mathbf{B}_1 \mathbf{x}(n-1) + \dots + \mathbf{B}_p \mathbf{x}(n-p) \quad (7.19)$$

Then, we compute $\mathbf{x}(n)$ from the recursive relation:

$$\mathbf{x}(n) = \mathbf{B}_0^{-1} [\mathbf{y}(n) - \mathbf{B}_1 \mathbf{x}(n-1) - \dots - \mathbf{B}_p \mathbf{x}(n-p)] \quad (7.20)$$

The final algorithm for computing the optimal mapping is sketched in Figure 7.6.

- Initialization: $\mathbf{y} = \mathbf{x} = \mathbf{e}$.
- Loop:
 1. Choose a random m from the set $\{-M, \dots, M\}$.
 2. Estimate $\beta_{\mathbf{y}^{(m)}}(\mathbf{y})$, the SFD of $\mathbf{y}^{(m)}$.
 3. Modify the outputs by $\mathbf{y}^{(m)} \leftarrow \mathbf{y}^{(m)} - \mu \beta_{\mathbf{y}^{(m)}}(\mathbf{y}^{(m)})$.
 4. For $k = 1, \dots, K$, do:
 - (a) Find the best $\mathbf{B}(z)$ which maps $\mathbf{x}(n)$ to $\mathbf{y}(n)$, using (5.29).
 - (b) Compute $\mathbf{x}(n) = [\text{inv}(\mathbf{B}(z))] \mathbf{y}(n)$ using the recursive equation (7.20).
 - (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 $\mathbf{y}(n) = [\mathbf{B}(z)] \mathbf{g}(\mathbf{e}(n))$.
 8. For $i = 1, 2$, remove the DC of y_i and normalize its energy.
- Repeat until convergence

Figure 7.7: Projection algorithm for separating CPNL mixtures.

7.3.2 Separation algorithm

The separation algorithm is then evident: first, modify $\mathbf{y}^{(m)}$ by $\mathbf{y}^{(m)} \leftarrow \mathbf{y}^{(m)} - \mu \beta_{\mathbf{y}^{(m)}}(\mathbf{y}^{(m)})$, then find g_i 's and $\mathbf{B}(z)$ by the algorithm of Fig. 7.6, and finally replace $\mathbf{y}(n)$ by $[\mathbf{B}(z)] \mathbf{g}(\mathbf{e}(n))$; and repeat this procedure until convergence.

As in the previous chapters, the indeterminacies require a special attention. First, we remove the DC's and normalize the energies of both \mathbf{x} and \mathbf{y} at each iteration. Secondly, at each iteration, we replace g_i 's by the smoothing spline which maps e_i to x_i . Finally, like with PNL mixtures, we repeat the algorithm loop of the Fig. 7.6 only a few times (5 to 10 times) and for its initialization we use the previous value of \mathbf{x} (that is, the value of \mathbf{x} which has been obtained in the previous iteration of the global algorithm).

The complete algorithm is shown in Fig. 7.7.

7.3.3 Experimental results



Experiment 7.4. This is the repetition of the experiment 7.2 with the projection approach.

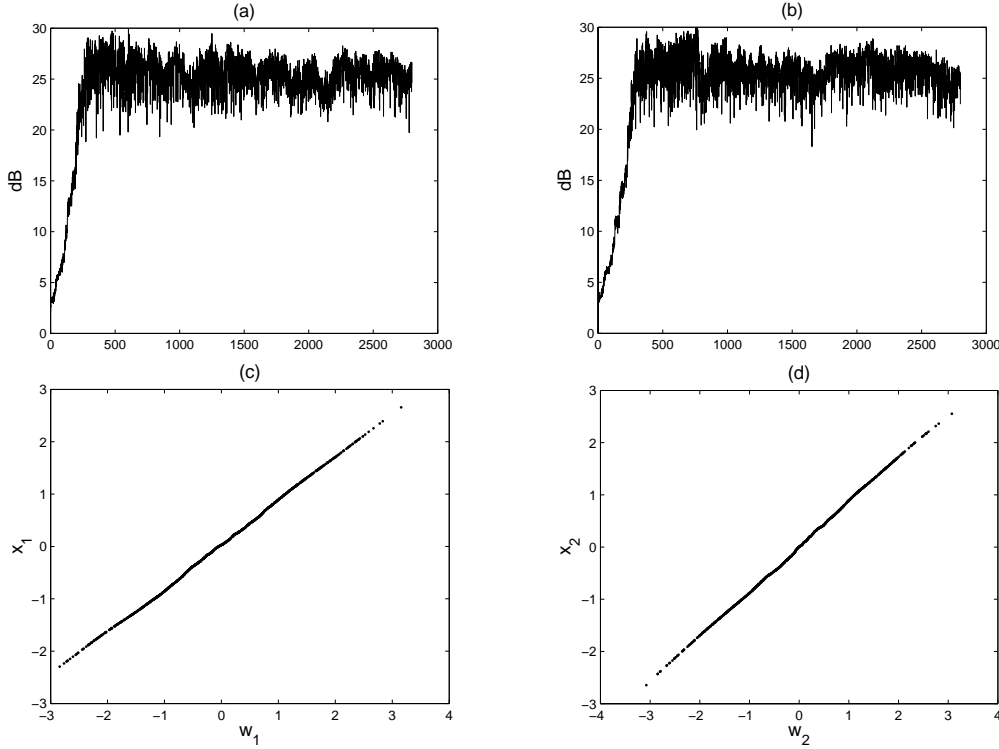


Figure 7.8: Separation result of separating a CPNL mixture by the projection approach (experiment 7.3): a) SNR_1 versus iterations, b) SNR_2 versus iterations, c) $g_1 \circ f_1$, and d) $g_2 \circ f_2$.

In this experiment, we use zero-mean unit-variance random sources with uniform distributions. The sources are mixed by the convolutive mixing matrix:

$$\mathbf{A}(z) = \begin{bmatrix} 1 + 0.2z^{-1} + 0.1z^{-2} & 0.5 + 0.3z^{-1} + 0.1z^{-2} \\ 0.5 + 0.3z^{-1} + 0.1z^{-2} & 1 + 0.2z^{-1} + 0.1z^{-2} \end{bmatrix} \quad (7.21)$$

followed by nonlinearities:

$$f_1(x) = f_2(x) = \tanh(2x) + 0.1x \quad (7.22)$$

The separation algorithm of Fig. 7.7 has been used with the following parameters: Pham's estimation method for estimating SFD, $\mu = 0.1$, $p = 2$, $M = 4$, and $\lambda = 0.99999$ for the smoothing spline used for estimating (and smoothing) g_i 's. The length of data block is 1000 (as in the experiment 7.2), but a new data block is taken at each iteration (in programming, this is done just before the step 7 of the algorithm of Fig. 7.7).

Signal to Noise Ratio (SNR), as defined in (7.18), is used for measuring the separation performance. The final separation result can be seen in Fig. 7.8.

Compare the results of this experiment with 7.2 (the figures 7.4 and 7.8). In both experiments

the same data block length and the same SFD estimator has been used. It can be seen that the results of the projection approach is better.

Run time: The time required for running 100 iterations of the experiment for each realization of the sources is approximately 40 seconds. Here, most of the time is consumed in computing the inputs of $\mathbf{B}(z)$ from its outputs (the recursive equation (7.20)).

7.4 Conclusion

In this chapter, we combined the methods of the chapters 5 and 6 for separating CPNL mixtures. The criterion was based on the mutual information of the outputs, and for its minimization, we have used SFD (as the gradient of the mutual information) through the gradient and projection approaches presented in the chapter 4.

Some experimental results have been presented. These experiments show that the projection approach results in better separation in CPNL mixtures (when the Pham's estimation of SFD is used).

Chapter 8

Conclusion and Perspectives

In this thesis, we have presented source separation algorithms for separating linear instantaneous, convolutive, PNL and CPNL mixtures. The principle of all the algorithms is to minimize the mutual information of the outputs, by using an iterative manner based on the steepest descent gradient algorithm. In fact, knowing the differential of the mutual information (Theorem 4.1), we have proposed two general approach for minimizing the mutual information of the outputs of a separating system: gradient and projection approaches. Then, we used these approaches in separating linear instantaneous, convolutive, PNL and CPNL mixtures, for them, a separability result is available.

Moreover, by mentioning an example (page 24), we show that even smooth mappings may mix the sources and preserve their independence. This fact shows that one cannot rely on the smoothness of the nonlinear separating system for separating the sources. In other words, in nonlinear source separation, it is essential¹ to have a structural constraint along with a separability result for that structure.

For PNL mixtures, we have presented a geometrical algorithm (for the case of two sources and two sensors). This algorithm has two interesting points. Firstly, it provides a proof for the separability of PNL mixtures where the sources are assumed to be bounded. Secondly, it shows that the nonlinearity in PNL mixtures can be compensated, before and independently from separating the sources.

The most interesting results of the thesis are the followings:

¹In this example (as well as the example of page 3), the outputs are only *instantaneously* independent. Recently, S. Hosseini and C. Jutten [47] have proposed that the independence in the sense of stochastic processes (*i.e.* the independence of $y_1(n)$ and $y_2(n-m)$ for all n and all m) is much more stronger than the instantaneous independence. Consequently, one may still hope to solve a nonlinear source separation problem by taking into account the time correlation of the signals and the independence of their shifted versions. The possibility of such an approach is still an open question.

- We derived a general expression for the differential of the mutual information (Theorem 4.1). This result can be used for its minimization. Two general approaches, gradient and projection approaches, have been proposed for its minimization. Moreover, we proved that the mutual information has no “local minimum” (Theorem 4.2).
- We introduced the concept of SFD, as the stochastic gradient of the mutual information. Using this concept, even with its simple estimations (*e.g.* histogram estimation) we can obtain good separation results.
- We provided another proof for separability of PNL mixtures of bounded sources (section 3.4).
- We designed algorithms for separating CPNL mixtures and new methods for separating linear instantaneous, convolutive and PNL mixtures.

As main perspectives of this work, we can give the following list:

- Considering more realistic mixtures. In the thesis, we have always used very simple convolutive mixtures, and simple signals. However, the case of real signals, and real filters (which may have large lengths) needs more investigation. By now, one great difficulty is the computational load of the algorithms.
- The CPNL mixing model corresponds to instantaneous nonlinear sensors after a linear convolutive mixture. The case where the sensors have nonlinear effects which varies in different frequencies needs more consideration. If one can model such a sensor by a linear filter followed by a nonlinear memoryless system, then the CPNL mixture is still a good model for the mixing system. Otherwise, some different models may need to be introduced.
- The performance of the algorithms, and a theoretical convergence analysis may be considered.
- The separation of the cascade of the PNL blocks can be considered. Our first experiments (with the gradient approach) have been somewhat successful in separating the sources. However, a separability result for such a system is not yet available. Consequently, even after using gradient or projection approaches for generating independent outputs, one cannot be sure that the sources have been separated.
- Searching for other SFD (or conditional score function) estimation methods, to achieve better separation quality. Especially, finding an optimal estimation (in MMSE sense) needs some investigation.

- As we saw, in separating PNL and CPNL mixtures, the output SNR's reaches a maximum, and then falls and converges to a smaller value. Although our experiments show that two reasons of this phenomena are the signal lengths and the SFD estimation method, the reason is not yet completely clear, and requires more investigation.
- The generalization of our method for separating convolutive and CPNL mixtures for the case of more than two sources, may be considered.

Appendix A

Spline Functions

The goal of this appendix is a brief review on the major aspects of the spline theory which have been used in this thesis.

A.1 Definitions

Definition A.1 A polynomial of “order” n is a function of the form:

$$p(x) = a_1 + a_2x + \cdots + a_nx^{n-1} \quad (\text{A.1})$$

Note that the degree of a polynomial of order n is not necessarily $n - 1$ (because it is possible that $a_n = 0$). However, it can be said that the degree of a polynomial of order n is strictly smaller than n .

Definition A.2 Let $\xi = \{\xi_i\}_1^{l+1}$ be a strictly increasing sequence, and let k be a positive integer. A “piecewise polynomial” (or *pp*) function of order k with “breakpoint sequence” ξ is a function which is a k -th order polynomial between each two successive breakpoints.

In other words, if $p_i(x)$ ($i = 1, \dots, l$) stand for a k -th order polynomials, then a *pp* function with the breakpoints $\{\xi_i, i = 1, \dots, l + 1\}$ is:

$$f(x) \triangleq p_i(x), \quad \xi_i < x < \xi_{i+1} \quad (\text{A.2})$$

Usually the function is extended beyond the first and last breakpoints, by the continuation of the first and last polynomial pieces.

A piecewise polynomial may be discontinuous at its breakpoints. If we add some smoothness constraints at the breakpoints of a *pp*-function, we obtain a *spline*:

Definition A.3 A “spline” of order k is a k -th order *pp*-function with smoothness constraints at its breakpoints.

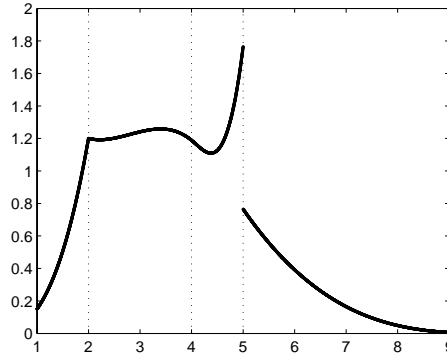


Figure A.1: An example of a cubic spline with different smoothness constraints at the knots 2, 4 and 5.

By a smoothness constraint, we mean a constraint on the continuity of the function itself, or on its derivatives.

Example. Figure A.1 shows an example of a 4-th order (also called cubic) spline. In other words, each piece of the function is 4-th order polynomial (*i.e.* a polynomial with degree 3 or less). The breakpoints of this spline are 2, 4 and 5. At 4, the spline, its derivative and its second derivative are all continuous. At 2, the spline is continuous, but its derivatives are not. At 5, even the spline itself is not continuous.

In the spline terminology, one often encounters the term “*knot sequence*”. The knot sequence is an increasing sequence (not necessarily strictly increasing) which contains the information about both of the breakpoints and the smoothness constraints at each breakpoint (provided that the order of spline is known). The convention is:

$$\boxed{\text{Number of smoothness conditions at a knot} + \text{multiplicity of the knot} = \text{order}}$$

In other words, for a knot ξ_i with multiplicity 1 (*i.e.* a breakpoint which is occurred only 1 time in the knot sequence) we have $k - 1$ continuity constraints, that is, the spline itself, and all of its derivatives until the $(k - 2)$ -th derivative must be continuous at this knot. Such a knot is also called a *simple* knot. In the same way, a k -th order spline may be discontinuous at a knot with multiplicity k .

Example. The knot sequence of the fourth order (cubic) spline of the previous example is $\{2, 2, 2, 4, 5, 5, 5, 5\}$.

A.2 Some interesting properties of splines

The splines have some properties which make them an interesting tool in many domains, including approximation theory, computer graphics, engineering and approximation theory. For example, the font shapes of the letters you are reading have been designed using the splines. Here we list some of these interesting points:

Property 1. For modeling or interpolating a function, we can use polynomial models. However, if the order of the polynomial is too small, it will not result in a good approximation for the rapid varying parts of the function. Conversely, if the order is chosen too large, the estimated function may be too varying in the other points. In other words, as shown in [33], with a polynomial approximation, “*if the function to be approximated is badly behaved anywhere in the interval of approximation, then the approximation is poor everywhere*”. However, by using low order splines (2nd degree or 3rd degree pp functions) we can well approximate rapid varying parts of a function (provided that there is enough knot points in that region), without affecting the other parts of the function.

Property 2. Another nice property of the splines, is the *minimum curvature* property of the cubic splines [4, 46]:

Theorem A.1 (Holladay) *Let the mesh $a = x_0 < x_1 < \dots < x_N = b$ and a set of real numbers $\{y_i\}_{i=0}^N$ are given. Among all the functions $f(x)$ with a continuous second order derivative on $[a, b]$ and $f(x_i) = y_i$ ($i = 0, \dots, N$), the cubic spline $\text{sp}(x)$ with the knot sequence $\{x_i\}$ and the end conditions $\text{sp}''(a) = \text{sp}''(b) = 0$ is the function which minimizes the integral:*

$$\int_a^b |f''(x)|^2 dx$$

Property 3. One of the properties of the splines, which leads to design fast algorithms for their computations, and hence a lot of applications in computer graphics, is the property that any k -th order spline with the knot sequence $\boldsymbol{\xi} = \{\xi_i\}_1^{l+1}$ can be represented as the sum:

$$\text{sp}(x) = a_1 B_1(x) + a_2 B_2(x) + \dots + a_N B_N(x) \quad (\text{A.3})$$

where $B_i(x)$ are some predetermined and fixed splines which are determined only by knowing the knot sequence. In other words, $\{B_i(x)\}$ are the basis functions of the linear space of the splines with a known knot sequence. Because of this property, the splines $B_i(x)$ are usually called B-splines.

There is a simple recursive algorithm for calculating the B-splines which correspond to a knot sequence $\{\xi_i\}$. Let us denote the k -th order B-splines of this knot sequence by $B_i^{(k)}(x)$.

Then we have:

$$B_i^{(1)}(x) = \begin{cases} 1 & ; \text{ if } \xi_i \leq x < \xi_{i+1} \\ 0 & ; \text{ Otherwise} \end{cases} \quad (\text{A.4})$$

In particular, $\xi_i = \xi_{i+1}$ implies that $B_i^{(1)}(x) \equiv 0$. Now, the k -th order B-spline can be obtained from the $(k-1)$ -th order B-spline by the following recursive formula:

$$B_i^{(k)}(x) = \omega_{ik}(x) B_i^{(k-1)}(x) + (1 - \omega_{i+1,k}(x)) B_{i+1}^{(k-1)}(x) \quad (\text{A.5})$$

where:

$$\omega_{ik}(x) \triangleq \begin{cases} \frac{x - \xi_i}{\xi_{i+k-1} - \xi_i} & ; \text{ if } \xi_i \neq \xi_{i+k-1} \\ 0 & ; \text{ Otherwise} \end{cases} \quad (\text{A.6})$$

From these relations, it can be seen that the B-spline $B_i^{(k)}(x)$ is zero outside the interval $[\xi_i, \xi_{i+k})$, and is completely specified by the knots ξ_i, \dots, ξ_{i+k} . The function `bspline` of MATLAB's spline toolbox, plots the B-spline $B_i^{(k)}(x)$ and its polynomial pieces, knowing the knots ξ_i, \dots, ξ_{i+k} .

The property 3 also shows that if we model a function on an interval (with some predetermined knot sequence), the model is a *linear* model with respect to the parameters of the model (*i.e.* the coefficients a_i).

On the other hand, this property provides two methods for representing a spline: pp-form and B-form. In the pp-form, the spline is represented by its *breakpoint sequence*, and the coefficients of each polynomial piece. In the B-form, the spline is represented by its *knot sequence* and the coefficients of the above sum (we emphasize on the difference between the breakpoint sequence and the knot sequence: the former is *strictly* increasing, but in the latter the multiplicity of elements contains some information about the smoothness at that knot). MATLAB's spline toolbox uses both of these representation and have some tools for converting one form to the other one.

Property 4 (Smoothing Splines). Another problem, which, at the first glance, may seem unrelated to the splines, is the problem of non-parametric regression. To state the problem more clearly, suppose that we have a set of N points (x_i, y_i) , and we are looking for a 'smooth' function which maps x_i 's to y_i 's, that is, a smooth non-parametric regression curve from x to y . In fact, we can find many functions f which give an exact mapping from x to y , that is, $y_i = f(x_i)$, for $i = 1, \dots, N$. Among them, the cubic spline is the smoothest. However, here we drop the requirement that $y_i = f(x_i)$ for all points and instead we are searching for a function f which minimizes the criterion:

$$\lambda \sum_i (y_i - f(x_i))^2 + (1 - \lambda) \int_{-\infty}^{\infty} (f''(x))^2 dx \quad (\text{A.7})$$

The parameter $0 \leq \lambda \leq 1$ determines the smoothness of the function: the smaller λ the smoother function, the greater λ the better fidelity to data.

Surprisingly, the answer of this problem, too, is a cubic spline with the knot sequence x_i . This property has been first presented by Reinsch [84, 85]. A simple proof using Dirac delta function can be found in [72]. A good reference on this subject is [39]. The extreme cases $\lambda = 0$ and $\lambda = 1$ corresponds to the linear regression function and the interpolating cubic spline, respectively [72].

In MATLAB spline toolbox, the smoothing spline can be computed by the “`csaps`” function.

A.3 End-conditions in approximation with cubic splines

A practically important subject in using the cubic splines, is their end-conditions. Suppose that we are looking for a cubic spline with the breakpoint sequence t_1, t_2, \dots, t_N and its known values at these points. The problem is that knowing the breakpoint sequence and the values of the function at these points *does not uniquely determine* a cubic spline which passes through these points. In fact, we have $N - 1$ polynomial piece, each with 4 coefficients, resulting in $4(N - 1)$ unknown coefficients. Moreover, for each of the interior points t_2, \dots, t_{N-1} , we have 4 equations (2 equations for the equality of the left and right polynomial pieces to $f(t_i)$ and 2 equations for the continuity of the first and second derivatives at these points), resulting to $4(N - 2) = 4N - 8$ equations for the interior points. However, for the end-points t_1 and t_N , we have only 2 equations. In summary, we must determine $4N - 4$ unknown coefficients from $4N - 6$ linear equations, and hence we need two other equations.

Depending on how we choose the two other necessary equations, we obtain different splines. Some of the most usual methods (also supported by the “`csape`” function of the MATLAB’s spline toolbox) are [33]:

- **Complete:** We use the known values for the slopes of the spline at the end points, that is, we use the equations $sp'(t_1) = s_1$ and $sp'(t_N) = s_N$.
- Using known values for the second derivatives of the spline at the end-points. A special case is taking $sp''(t_1) = sp''(t_N) = 0$ which is called **free-end** conditions, or **natural** spline interpolation. Generally, this method is not considered [33], unless there is good reasons for using it.
- **Periodic:** In this method, which is suited for periodic functions, it is assumed that $sp'(t_1) = sp'(t_N)$ and $sp''(t_1) = sp''(t_N)$.

- **not-a-knot end conditions:** This is the most frequently used method. In this method, it is assumed that the third derivatives, too, are continuous at the points t_2 and t_{N-1} . This assumption results in identical two first and two last polynomial pieces. In other words, this method is like that we have the knot sequence $\{t_1, t_3, \dots, t_{N-2}, t_N\}$ and we compute the cubic spline with two additional known values for $\text{sp}(t_2)$ and $\text{sp}(t_{N-1})$. In other words, in this method the knot points t_2 and t_{N-1} are not active, hence the name “not-a-knot” end conditions.

A.4 Multi-dimensional smoothing splines

The concept of the smoothing splines can be extended to multi-variate functions [39]. In the multi-variate non-parametric regression, we are searching for a smooth function $y = f(\mathbf{x})$, where $\mathbf{x} \triangleq (x_1, \dots, x_p)$, from data points $\{(\mathbf{x}_i, y_i), i = 1, \dots, N\}$. For taking into account the smoothness of the function, we minimize the criterion:

$$\frac{1}{N} \sum_i (y_i - f(\mathbf{x}_i))^2 + \lambda J_{m,p} \quad (\text{A.8})$$

where:

$$J_{m,p} \triangleq \sum_{\alpha_1 + \dots + \alpha_p = m} \left\{ \frac{m!}{\alpha_1! \dots \alpha_p!} \int_{-\infty}^{+\infty} \dots \int_{-\infty}^{+\infty} \left(\frac{\partial^m f}{\partial x_1^{\alpha_1} \dots \partial x_p^{\alpha_p}}(\mathbf{x}) \right)^2 d\mathbf{x} \right\} \quad (\text{A.9})$$

The answer of this problem is usually called a “Laplacian smoothing spline” [39]. For the case $m = p = 2$, (A.8) can be interpreted as a measure of the amount of energy required to bend a thin plate of infinite extent that is connected to the y_i by springs. In that case, λ represents a constant related to the flexibility of the plate [39]. Consequently, these splines are also called “thin-plate splines”.

A closed form expression for the Laplacian smoothing spline has been found by Wahaba and Wendelberger [39]. Because of the complexity of the solution, we do not repeat it here. But we cite the web page “<http://www.wisc.edu/mathsoft/msgdls.html>”, from it, a set of FORTRAN routines are available for computing the multi-dimensional smoothing splines. We have written the programs of the section 4.8.1 by using these routines.

Appendix B

Proofs

B.1 Theorem 4.1

To prove the theorem, we first have to prove two lemmas. The scalar versions of these lemmas have been already proposed [2].

Lemma B.1 *Let $\mathbf{x} = (x_1, \dots, x_N)^T$ be a bounded random vector and $\mathbf{\Delta} = (\Delta_1, \dots, \Delta_N)^T$ be a ‘small’ random vector, then*

$$p_{\mathbf{x}+\mathbf{\Delta}}(\mathbf{t}) - p_{\mathbf{x}}(\mathbf{t}) = - \sum_{i=1}^N \frac{\partial}{\partial t_i} \{E[\Delta_i | \mathbf{x} = \mathbf{t}] p_{\mathbf{x}}(\mathbf{t})\} + o(\mathbf{\Delta}) \quad (\text{B.1})$$

Proof. For any differentiable functions $h(\mathbf{t})$, we have:

$$h(\mathbf{x} + \mathbf{\Delta}) - h(\mathbf{x}) = \sum_i \Delta_i \frac{\partial h}{\partial t_i}(\mathbf{x}) + o(\mathbf{\Delta}) \quad (\text{B.2})$$

Thus:

$$E\{h(\mathbf{x} + \mathbf{\Delta}) - h(\mathbf{x})\} = \sum_i E\left\{\Delta_i \frac{\partial h}{\partial t_i}(\mathbf{x})\right\} + o(\mathbf{\Delta}) \quad (\text{B.3})$$

As a result:

$$\int_{\mathbf{t}} h(\mathbf{t}) \left(p_{\mathbf{x}+\mathbf{\Delta}}(\mathbf{t}) - p_{\mathbf{x}}(\mathbf{t})\right) d\mathbf{t} = \sum_i E\left\{\frac{\partial h}{\partial t_i}(\mathbf{x}) E[\Delta_i | \mathbf{x} = \mathbf{t}]\right\} + o(\mathbf{\Delta}) \quad (\text{B.4})$$

Using integration by parts, the i -th right term of (B.4) can be written:

$$\begin{aligned} E\left\{\frac{\partial h}{\partial t_i}(\mathbf{x}) E[\Delta_i | \mathbf{x} = \mathbf{t}]\right\} &= \int_{\mathbf{t}} \frac{\partial h}{\partial t_i}(\mathbf{t}) E[\Delta_i | \mathbf{x} = \mathbf{t}] p_{\mathbf{x}}(\mathbf{t}) d\mathbf{t} \\ &= - \int_{\mathbf{t}} h(\mathbf{t}) \frac{\partial}{\partial t_i} \{E[\Delta_i | \mathbf{x} = \mathbf{t}] p_{\mathbf{x}}(\mathbf{t})\} d\mathbf{t} \end{aligned} \quad (\text{B.5})$$

Combining (B.4) and (B.5), we obtain:

$$\int_{\mathbf{t}} h(\mathbf{t}) \left(p_{\mathbf{x}+\mathbf{\Delta}}(\mathbf{t}) - p_{\mathbf{x}}(\mathbf{t}) \right) d\mathbf{t} = - \int_{\mathbf{t}} h(\mathbf{t}) \sum_{i=1}^N \frac{\partial}{\partial t_i} \{ E [\Delta_i | \mathbf{x} = \mathbf{t}] p_{\mathbf{x}}(\mathbf{t}) \} d\mathbf{t} + o(\mathbf{\Delta}) \quad (\text{B.6})$$

Equation (B.1) can be obtained since the above equality holds for any functions h .

▲

Lemma B.2 *Let \mathbf{x} and $\mathbf{\Delta}$ be as defined in Lemma B.1, then:*

$$H(\mathbf{x} + \mathbf{\Delta}) - H(\mathbf{x}) = -E \{ \mathbf{\Delta}^T \boldsymbol{\varphi}_{\mathbf{x}}(\mathbf{x}) \} + o(\mathbf{\Delta}) \quad (\text{B.7})$$

where H denotes the Shannon's entropy, and $p_{\mathbf{x}}(\cdot)$ and $\boldsymbol{\varphi}_{\mathbf{x}}(\cdot)$ are the PDF and the JSF of \mathbf{x} , respectively.

Proof. We write:

$$\begin{aligned} H(\mathbf{x} + \mathbf{\Delta}) - H(\mathbf{x}) &= -E \left\{ \ln p_{\mathbf{x}+\mathbf{\Delta}}(\mathbf{x} + \mathbf{\Delta}) \right\} + E \{ \ln p_{\mathbf{x}}(\mathbf{x}) \} \\ &= E \left\{ \ln \frac{p_{\mathbf{x}}(\mathbf{x} + \mathbf{\Delta})}{p_{\mathbf{x}+\mathbf{\Delta}}(\mathbf{x} + \mathbf{\Delta})} \right\} - E \left\{ \ln \frac{p_{\mathbf{x}}(\mathbf{x} + \mathbf{\Delta})}{p_{\mathbf{x}}(\mathbf{x})} \right\} \end{aligned} \quad (\text{B.8})$$

In the neighborhood of 1, $\ln x = (x - 1) - \frac{1}{2}(x - 1)^2 + \dots$, thus by defining $\mathbf{T} = \mathbf{x} + \mathbf{\Delta}$, we can simplify the first term of (B.8) as:

$$\begin{aligned} E \left\{ \ln \frac{p_{\mathbf{x}}(\mathbf{T})}{p_{\mathbf{T}}(\mathbf{T})} \right\} &= E \left\{ \frac{p_{\mathbf{x}}(\mathbf{T})}{p_{\mathbf{T}}(\mathbf{T})} - 1 \right\} + o(\mathbf{\Delta}) \\ &= \int_{\mathbf{t}} \left(\frac{p_{\mathbf{x}}(\mathbf{t})}{p_{\mathbf{T}}(\mathbf{t})} - 1 \right) p_{\mathbf{T}}(\mathbf{t}) d\mathbf{t} + o(\mathbf{\Delta}) \\ &= o(\mathbf{\Delta}) \end{aligned} \quad (\text{B.9})$$

Now, the second right term of (B.8) becomes:

$$\begin{aligned} -E \left\{ \ln \frac{p_{\mathbf{x}}(\mathbf{x} + \mathbf{\Delta})}{p_{\mathbf{x}}(\mathbf{x})} \right\} &= E \{ \ln p_{\mathbf{x}}(\mathbf{x}) \} - E \{ \ln p_{\mathbf{x}}(\mathbf{x} + \mathbf{\Delta}) \} \\ &= \int_{\mathbf{t}} \ln p_{\mathbf{x}}(\mathbf{t}) p_{\mathbf{x}}(\mathbf{t}) d\mathbf{t} - \int_{\mathbf{t}} \ln p_{\mathbf{x}}(\mathbf{t}) p_{\mathbf{x}+\mathbf{\Delta}}(\mathbf{t}) d\mathbf{t} \\ &= \int_{\mathbf{t}} \ln p_{\mathbf{x}}(\mathbf{t}) \left(p_{\mathbf{x}}(\mathbf{t}) - p_{\mathbf{x}+\mathbf{\Delta}}(\mathbf{t}) \right) d\mathbf{t} \end{aligned} \quad (\text{B.10})$$

Using Lemma B.1, this term will be:

$$-E \left\{ \ln \frac{p_{\mathbf{x}}(\mathbf{x} + \mathbf{\Delta})}{p_{\mathbf{x}}(\mathbf{x})} \right\} = \sum_i \int_{\mathbf{t}} \ln p_{\mathbf{x}}(\mathbf{t}) \frac{\partial}{\partial t_i} \{ E [\Delta_i | \mathbf{x} = \mathbf{t}] p_{\mathbf{x}}(\mathbf{t}) \} d\mathbf{t} + o(\mathbf{\Delta}) \quad (\text{B.11})$$

And after integrating by parts, we will have:

$$\begin{aligned}
-E \left\{ \ln \frac{p_{\mathbf{x}}(\mathbf{x} + \Delta)}{p_{\mathbf{x}}(\mathbf{x})} \right\} &= - \sum_i \int_{\mathbf{t}} E [\Delta_i | \mathbf{x} = \mathbf{t}] \varphi_i(\mathbf{t}) p_{\mathbf{x}}(\mathbf{t}) d\mathbf{t} + o(\Delta) \\
&= - \sum_i E \{ E [\Delta_i | \mathbf{x}] \varphi_i(\mathbf{x}) \} + o(\Delta) \\
&= - \sum_i E \{ \Delta_i \varphi_i(\mathbf{x}) \} + o(\Delta) \\
&= - E \{ \Delta^T \varphi_{\mathbf{x}}(\mathbf{x}) \} + o(\Delta)
\end{aligned} \tag{B.12}$$

The proof of the lemma is achieved by combining the equations (B.8), (B.9) and (B.12). ▲

Corollary 1 For scalar random variables x_i and Δ_i , we have:

$$H(x_i + \Delta_i) - H(x_i) = -E \{ \Delta_i \cdot \psi_{x_i}(x_i) \} + o(\Delta_i) \tag{B.13}$$

Proof of Theorem 4.1. Combining the usual expression $I(\mathbf{x}) = \sum_i H(x_i) - H(\mathbf{x})$ with equations (B.7) and (B.13) proves the theorem. ▲

B.2 Theorem 5.1

Proof. By using the separating matrix $B = \begin{bmatrix} 1 & b_1 \\ b_2 & 1 \end{bmatrix}$, we will have:

$$\begin{aligned}
y_1(n) &= \sum_{k=-\infty}^{\infty} \{h_1(k) + a_2 b_1 h_2(k)\} s_1(n-k) \\
&+ \sum_{k=-\infty}^{\infty} \{a_1 h_1(k) + b_1 h_2(k)\} s_2(n-k)
\end{aligned} \tag{B.14}$$

$$\begin{aligned}
y_2(n) &= \sum_{k=-\infty}^{\infty} \{b_2 h_1(k) + a_2 h_2(k)\} s_1(n-k) \\
&+ \sum_{k=-\infty}^{\infty} \{a_1 b_2 h_1(k) + h_2(k)\} s_2(n-k)
\end{aligned} \tag{B.15}$$

Suppose that $y_1(n)$ and $y_2(n)$ are independent. Applying Darmois-Skitovich theorem page 2 for the random variables $s_1(n-k)$ and $s_2(n-k)$, we will have:

$$\begin{cases} b_2 h_1(k) + a_2 h_2(k) = 0 \\ a_1 h_1(k) + b_1 h_2(k) = 0 \end{cases} \tag{B.16}$$

or:

$$\begin{cases} h_1(k) + a_2 b_1 h_2(k) = 0 \\ a_1 b_2 h_1(k) + h_2(k) = 0 \end{cases} \quad (\text{B.17})$$

By the assumptions: $\exists m$ such that $h_1(m)h_2(m) \neq 0$. Now suppose that for $k = m$, equation (B.16) holds. Hence:

$$b_1 = -a_1 \frac{h_1(m)}{h_2(m)} \quad (\text{B.18})$$

$$b_2 = -a_2 \frac{h_2(m)}{h_1(m)} \quad (\text{B.19})$$

Let $c = \frac{h_1(m)}{h_2(m)}$ and suppose that at least one of the values of a_1 or a_2 is nonzero. Inserting the above values in (B.16), we have $h_1(k) = ch_2(k)$, and inserting them in (B.17) we obtain:

$$\begin{bmatrix} h_2(m) & -a_1 a_2 h_1(m) \\ -a_1 a_2 h_2(m) & h_1(m) \end{bmatrix} \begin{bmatrix} h_1(k) \\ h_2(k) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (\text{B.20})$$

The determinant of the coefficient matrix of this equation is $(1 - a_1^2 a_2^2) h_1(m) h_2(m)$ and cannot vanish by the assumptions. Hence, its unique solution is $h_1(k) = h_2(k) = 0$, which is the solution of (B.16), too. Therefore, (B.16) is satisfied for all k , and we have always $h_1(k) = ch_2(k)$. Also, we have:

$$y_1(n) = \sum_{k=-\infty}^{\infty} \{h_1(k) + a_2 b_1 h_2(k)\} s_1(n - k) \quad (\text{B.21})$$

$$y_2(n) = \sum_{k=-\infty}^{\infty} \{a_1 b_2 h_1(k) + h_2(k)\} s_2(n - k) \quad (\text{B.22})$$

Hence, it is clear that $y_1(n)$ and $y_2(n - m)$ are independent for all m . The same result can be obtained from the assumption that equation (B.17) holds for $k = m$. \blacktriangle

Appendix C

Scalar Score Functions

Definition C.1 Let x be a random variable with the probability density function (PDF) $p_x(x)$. The score function of x is:

$$\psi_x(x) \triangleq -\frac{d}{dx} \ln p_x(x) = -\frac{p'_x(x)}{p_x(x)} \quad (\text{C.1})$$

The following property has been used by Cox [28] for designing an estimator for score functions. It has been applied in source separation by Taleb and Jutten [96, 98].

Property C.1 Let x be a random variable with the PDF $p_x(x)$ and the score function $\psi_x(x)$. Moreover, let f be a continuously differentiable function and:

$$\lim_{x \rightarrow \pm\infty} f(x)p_x(x) = 0 \quad (\text{C.2})$$

Then:

$$E \{f(x)\psi_x(x)\} = E \{f'(x)\} \quad (\text{C.3})$$

Note that the condition (C.2) is not so restrictive, because it holds for most “usual” random variables which one encounters in the physical phenomena. For example, it is satisfied for all bounded random variables.

Proof. We have:

$$\begin{aligned} E \{f(x)\psi_x(x)\} &= \int_{-\infty}^{+\infty} f(x)\psi_x(x)p_x(x)dx \\ &= - \int_{-\infty}^{+\infty} f(x)\frac{p'_x(x)}{p_x(x)}p_x(x)dx \\ &= - \int_{-\infty}^{+\infty} f(x)p'_x(x)dx \\ &= \int_{-\infty}^{+\infty} f'(x)p_x(x)dx \quad (\text{Integration by parts and (C.2)}) \\ &= E \{f'(x)\} \end{aligned}$$

which proves the property. ▲

Corollary 1 *For a bounded random variable x , we have:*

$$E \{ \psi_x(x) x \} = 1 \quad (\text{C.4})$$

Corollary 2 *Let x be a bounded random variable. For the parametric function $f(\mathbf{w}, x)$, where $\mathbf{w} = (w_1, \dots, w_M)^T$ denotes the parameter vector, we have:*

$$\operatorname{argmin}_{\mathbf{w}} E \left\{ (\psi_x(x) - f(\mathbf{w}, x))^2 \right\} = \operatorname{argmin}_{\mathbf{w}} \left\{ E \{ f^2(\mathbf{w}, x) \} - 2E \{ f'(\mathbf{w}, x) \} \right\} \quad (\text{C.5})$$

This corollary shows that one can easily design a Minimum Mean Square Error (MMSE) estimator for the score function of a random variable.

For example, suppose that we would like to estimate the score function as a linear sum of the functions $k_1(x), k_2(x), \dots, k_M(x)$, that is:

$$\hat{\psi}_x(x) = w_1 k_1(x) + w_2 k_2(x) + \dots + w_M k_M(x) = \mathbf{k}(x)^T \mathbf{w} \quad (\text{C.6})$$

where $\mathbf{k}(x) \triangleq (k_1(x), \dots, k_M(x))^T$. The coefficient w_1, \dots, w_M must be determined such that the error term $E \left\{ (\psi_x(x) - \hat{\psi}_x(x))^2 \right\}$ is minimized.

From the orthogonality principle [77] we can write:

$$E \left\{ \mathbf{k}(x) (\psi_x(x) - \hat{\psi}_x(x)) \right\} = 0 \quad (\text{C.7})$$

which results in:

$$E \left\{ \mathbf{k}(x) \mathbf{k}(x)^T \right\} \mathbf{w} = E \left\{ \mathbf{k}(x) \psi_x(x) \right\} \quad (\text{C.8})$$

From the property C.1, we can rewrite this equation as:

$$E \left\{ \mathbf{k}(x) \mathbf{k}(x)^T \right\} \mathbf{w} = E \left\{ \mathbf{k}'(x) \right\} \quad (\text{C.9})$$

which determines \mathbf{w} .

Appendix D

Kernel Estimators

Kernel estimation is a widely used method for estimating the Probability Density Function (PDF) of a random variable, from its observed samples [66, 90]. Its original idea belongs to Rosenblat [88].

D.1 Scalar density kernel estimation

A “*kernel*” is a function that satisfies the conditions:

1. $\forall x, \quad k(x) \geq 0.$
2. $\int_{-\infty}^{+\infty} k(x) dx = 1.$
3. $\int_{-\infty}^{+\infty} x k(x) dx = 0.$

In other words, $k(x)$ is a kernel if it is the PDF of a zero mean random variable. The standard deviation of this random variable is called the “*bandwidth*” or the “*smoothing parameter*” of the kernel.

Definition D.1 *Let $k_h(x)$ be any kernel function with bandwidth h . Then the kernel estimation of the PDF of the random variable x from the observations $\{x_1, \dots, x_N\}$ is:*

$$\hat{p}_x(x) \triangleq \frac{1}{N} \sum_{i=1}^N k_h(x - x_i) \tag{D.1}$$

Figure D.1 illustrates the idea of the kernel estimation. With this method, a kernel is placed around each sample, and then they are summed up to form an estimation of the PDF of the random variable. The more samples occurred in a region, the higher PDF in that region. This is heuristically what one expects from the meaning of a PDF.

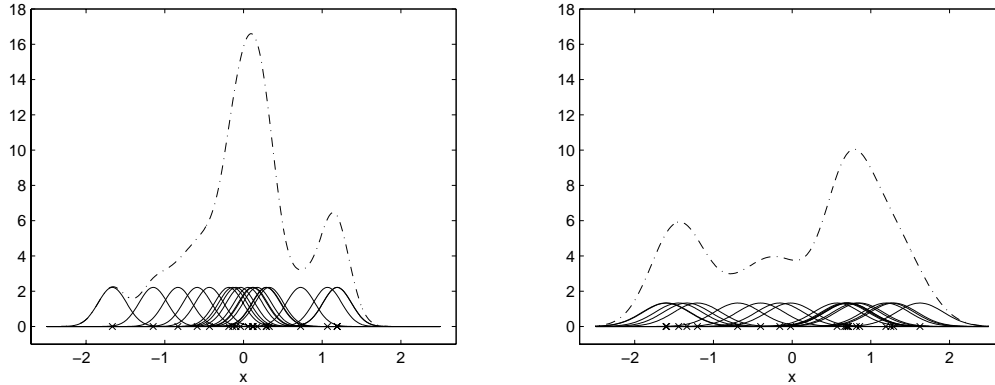


Figure D.1: The idea of the kernel estimators. Left) small bandwidth, Right) Large bandwidth

The bandwidth of the kernel (h) determines the smoothness of the estimator (see Fig. D.1). The larger h , the smoother estimation of the PDF.

Theoretically [66], the kernel estimator is a biased estimation of the PDF. The bias of the estimation tends to zero, when $h \rightarrow 0$, and the variance of the estimation tends to zero, when $Nh \rightarrow \infty$. Hence, in practice, when a limited number of observations is available, the choice of h is important. If h chosen too large, the estimated PDF will be the rough estimate of the shape of the kernel. On the contrary, if h is chosen too small, the estimated PDF will be too varying. A rule of thumb for choosing h , when Gaussian kernels are used, is [66]:

$$h = \hat{\sigma}_x \left(\frac{4}{3N} \right)^{\frac{1}{5}} \approx 1.06 \hat{\sigma}_x N^{-1/5} \quad (\text{D.2})$$

where $\hat{\sigma}_x$ denotes the standard deviation (this formula is optimal when the PDF's are Gaussian).

D.2 Multivariate density kernel estimation

The kernel estimation can also be used in estimating the PDF of random vectors. A d -variate kernel function $k(\mathbf{x}) = k(x_1, \dots, x_d)$, is the PDF of a d -dimensional zero mean random vector. Usually, the symmetrical kernels are used, for example the d -dimensional Gaussian kernel:

$$k(\mathbf{x}) = (2h\pi)^{-d/2} \exp\left(-\frac{1}{2h^2} \mathbf{x}^T \mathbf{x}\right) \quad (\text{D.3})$$

In multivariate kernels, the bandwidth in different directions (that is, $h_i \triangleq \int x_i^2 k(\mathbf{x}) d\mathbf{x}$) can be different. A kernel with equal bandwidth in all directions is called an isotropic kernel. If $k_h(\mathbf{x})$ denotes an isotropic kernel with bandwidth h , then the kernel estimation of the PDF

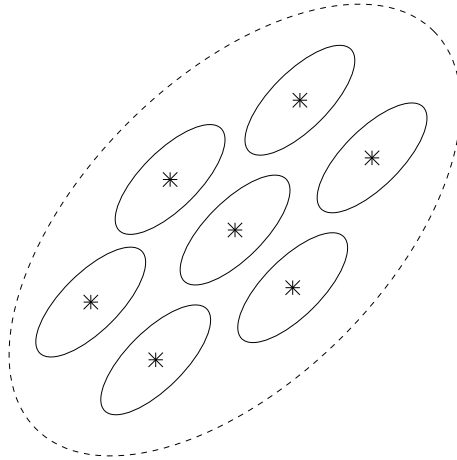


Figure D.2: Shape of kernel and shape of distribution, if Fukunaga method is used.

of the random vector \mathbf{x} , from the observations $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ is:

$$\hat{p}_{\mathbf{x}}(\mathbf{x}) \triangleq \frac{1}{N} \sum_{i=1}^N k_h(\mathbf{x} - \mathbf{x}_i) \quad (\text{D.4})$$

With isotropic kernels, the kernel placed on each data point is scaled equally in all directions. In certain circumstances, it may be more appropriate to use different smoothing parameters in different directions. This will be the case, for example, if the spread of the data points is very much greater in one of the coordinate directions than the others. For doing this, an attractive approach, suggested by Fukunaga [40, 90], is the following: (1) “prewhiten” the data, (2) PDF estimation by isotropic kernels, and (3) transform back again. This is equivalent to use non-isotropic kernels. Figure D.2 shows this idea graphically.

To state this method more precisely, let $\mathbf{R}_{\mathbf{x}}$ denote the covariance matrix of \mathbf{x} , and \mathbf{T} be its Cholesky decomposition, *i.e.* an upper triangular matrix which satisfies $\mathbf{R}_{\mathbf{x}} = \mathbf{T}\mathbf{T}^T$. Now, with the transformation $\mathbf{y} = \mathbf{T}^{-1}\mathbf{x}$, we have $\mathbf{R}_{\mathbf{y}} = \mathbf{I}$, that is, the variance of \mathbf{y} is the same in all directions, and the variables y_i are uncorrelated. Hence, it is natural to use the isotropic kernels for estimating the PDF of \mathbf{y} , and then we use:

$$\hat{p}_{\mathbf{x}}(\mathbf{x}) = \frac{\hat{p}_{\mathbf{y}}(\mathbf{y})}{|\det \mathbf{T}|} \quad (\text{D.5})$$

for estimating the PDF of \mathbf{x} .

After prewhitening the data, we must choose a value for the bandwidth h . In [90], the following rule of thumb has been proposed for choosing h , provided that the prewhitening has been done:

$$h_{opt} = cN^{-1/(d+4)} \quad (\text{D.6})$$

where d is the dimension of the random vector, and c is a constant which depends on the type of the kernel. For the d -dimensional Gaussian kernels:

$$c = \left(\frac{4}{2d+1} \right)^{\frac{1}{d+4}} \quad (\text{D.7})$$

For example, for scalar Gaussian kernels we have $c = 1.06$ (and (D.6) is converted to (D.2)), and for 2-dimensional Gaussian kernels $c = 0.96$.

Appendix E

Some simple lemmas

This appendix contains some simple mathematical definitions and lemmas which have been used in the thesis.

Definition E.1 Let $\mathbf{A} = [a_{ij}]$ and $\mathbf{B} = [b_{ij}]$ be two $M \times N$ matrices. Then, their scalar (Euclidean) product is:

$$\langle \mathbf{A}, \mathbf{B} \rangle \triangleq \sum_{i=1}^M \sum_{j=1}^N a_{ij} b_{ij} = \text{trace}(\mathbf{A}\mathbf{B}^T) \quad (\text{E.1})$$

where trace is the sum of the diagonal elements of a square matrix.

Lemma E.1 Suppose that the product $\mathbf{x}^T \mathbf{A} \mathbf{y}$ is defined, where \mathbf{A} is a matrix and \mathbf{x} and \mathbf{y} are column vectors. Then:

$$\mathbf{x}^T \mathbf{A} \mathbf{y} = \langle \mathbf{A}, \mathbf{y}\mathbf{x}^T \rangle \quad (\text{E.2})$$

Proof. Recall that if for the matrices \mathbf{M} and \mathbf{N} the products \mathbf{MN} and \mathbf{NM} are both defined, then $\text{trace}(\mathbf{MN}) = \text{trace}(\mathbf{NM})$. From this property we can write:

$$\begin{aligned} \mathbf{x}^T \mathbf{A} \mathbf{y} &= \text{trace}(\mathbf{x}^T \mathbf{A} \mathbf{y}) \\ &= \text{trace}(\mathbf{A} \mathbf{y} \mathbf{x}^T) \\ &= \text{trace}(\mathbf{A} (\mathbf{y} \mathbf{x}^T)^T) \\ &= \langle \mathbf{A}, \mathbf{y} \mathbf{x}^T \rangle \end{aligned} \quad (\text{E.3})$$

▲

Lemma E.2 Let f be a differentiable function with respect to the matrix \mathbf{A} . Then:

$$f(\mathbf{A} + \mathcal{E}) = f(\mathbf{A}) + \left\langle \mathcal{E}, \frac{\partial f}{\partial \mathbf{A}}(\mathbf{A}) \right\rangle + o(\mathcal{E}) \quad (\text{E.4})$$

where \mathcal{E} is a “small” matrix and $o(\mathcal{E})$ denotes higher order terms in \mathcal{E} .

Lemma E.3 Suppose that the product $\mathbf{x}^T \mathbf{A} \mathbf{y}$ is defined, where \mathbf{A} is a matrix and \mathbf{x} and \mathbf{y} are column vectors. Then:

$$\frac{\partial}{\partial \mathbf{A}} (\mathbf{x}^T \mathbf{A} \mathbf{y}) = \mathbf{xy}^T \quad (\text{E.5})$$

Proof. Is evident from the lemma E.1. ▲

Lemma E.4 Suppose for the matrix \mathbf{A} and the vector \mathbf{x} the product $\mathbf{A} \mathbf{x}$ is defined. Then:

$$\frac{\partial}{\partial \mathbf{A}} (\mathbf{x}^T \mathbf{A}^T \mathbf{A} \mathbf{x}) = \frac{\partial}{\partial \mathbf{A}} (\|\mathbf{A} \mathbf{x}\|^2) = 2 \mathbf{A} \mathbf{x} \mathbf{x}^T \quad (\text{E.6})$$

Proof. Defining $\mathbf{y} \triangleq \mathbf{A} \mathbf{x}$, we have:

$$\begin{aligned} \frac{\partial}{\partial a_{ij}} (\|\mathbf{A} \mathbf{x}\|^2) &= \frac{\partial}{\partial a_{ij}} \|\mathbf{y}\|^2 \\ &= \frac{\partial}{\partial a_{ij}} \sum_k y_k^2 \\ &= 2y_i x_j \end{aligned} \quad (\text{E.7})$$

where the last equation has been written from $y_k = \sum_j a_{kj} x_j$. From the above equation we have $\frac{\partial}{\partial \mathbf{A}} (\|\mathbf{A} \mathbf{x}\|^2) = 2 \mathbf{y} \mathbf{x}^T$, which proves the lemma. ▲

Bibliography

- [1] K. Abed-Meraim, Ph. Loubaton, and E. Moulines, “A subspace algorithm for certain blind identification problem,” *IEEE Transaction on Information Theory*, pp. 499–511, March 1997.
- [2] S. Achard, *Initiation a la Séparation aveugle de sources dans des mélanges post non linéaires*, DEA de l’INP de Grenoble, June 2000, (in French).
- [3] S. Achard and D.-T. Pham, “Blind source separation in post nonlinear mixtures,” in *ICA2001*, San Diego, California, December 2001, pp. 295–300.
- [4] J. H. Ahlberg, E. N. Nilson, and J. L. Walsh, *The theory of splines and their applications*, Academic Press, 1967.
- [5] L. B. Almeida, “ICA of linear and nonlinear mixtures based on mutual information,” in *International Joint Conference on Neural Networks*, Washington, DC, USA, July 2001.
- [6] S. I. Amari, “Natural gradient works efficiently in learning,” *Neural Computation*, vol. 10, pp. 251–276, 1998.
- [7] B. Ans, J. Héroult, and C. Jutten, “Adaptive neural architectures: Detection of primitives,” in *Proceedings of COGNITIVA’85*, Paris, France, 4-7 June 1985, pp. 593–597.
- [8] M. Babaie-Zadeh, C. Jutten, and K. Nayebi, “Blind separating Convolutive Post-Nonlinear mixtures,” in *ICA2001*, San Diego, California, December 2001, pp. 138–143.
- [9] 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.

-
- [10] 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).
- [11] 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.
- [12] 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.
- [13] M. Babaie-Zadeh, C. Jutten, and K. Nayebi, "Using joint score functions in separating post non-linear mixtures," *Scientia-Iranica*, 2002, accepted.
- [14] T. Bell and T. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural Comutation*, vol. 7, no. 6, pp. 1004–1034, 1995.
- [15] G. Burel, "Blind separation of sources: a nonlinear neural algorithm," *Neural Networks*, vol. 5, no. 6, pp. 937–947, 1992.
- [16] V. Capdevielle, Ch. Servière, and Lacoume J.-L., "Blind separation of wide-band sources in the frequency domain," in *ICASSP*, 1995, pp. 2080–2083.
- [17] V. Capdevielle, Ch. Servière, and Lacoume J.-L., "Separation of wide band sources," in *HOS95*, June 1995, pp. 66–70.
- [18] J.-F. Cardoso, "Blind signal separation: statistical principles," *Proceedings IEEE*, vol. 9, pp. 2009–2025, 1998.
- [19] J.-F. Cardoso, "High order contrasts for independent component analysis," *Neural Computation*, vol. 11, pp. 157–192, 1999.
- [20] J.-F. Cardoso, "The three easy routes to independent component analysis; contrasts and geometry," in *ICA2001*, San Diego, California, December 2001, pp. 1–6.
- [21] J.-F. Cardoso and B. Laheld, "Equivariant adaptive source separation," *IEEE Trans. on SP*, vol. 44, no. 12, pp. 3017–3030, December 1996.
- [22] J.-F. Cardoso and A. Souloumiac, "Blind beamforming for non Gaussian signals," *IEE Proceedings-F*, vol. 140, pp. 362–370, December 1993.

- [23] N. Charkani, *Séparation auto-adaptative de sources pour des mélanges convolutifs. Application à la téléphonie mains-libres dans les voitures*, Thèse de l'INP Grenoble, 1996, (in French).
- [24] N. Charkani and Y. Deville, "A convolutive source separation method with self-optimizing nonlinearities," in *ICASSP*, April 1999, pp. 2909–2912.
- [25] P. Comon, "Analyse en composantes indépendantes et identification aveugle," *Traitement du signal*, vol. 7, no. 5, pp. 435–450, 1990, (in French).
- [26] P. Comon, "Independent component analysis, a new concept?," *Signal Processing*, vol. 36, no. 3, pp. 287–314, 1994.
- [27] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley Series in Telecommunications, 1991.
- [28] D. D. Cox, "A penalty method for nonparametric estimation of the logarithmic derivative of a density function," *Ann. Instit. Statist. Math.*, vol. 37, pp. 271–288, 1985.
- [29] S. Dapena and C. Servière, "A simplified frequency domain approach for blind separation of convolutive mixtures," in *ICA2001*, San Diego, California, December 2001, pp. 295–300.
- [30] G. A. Darbellay and P. Tichavský, "Independent component analysis through direct estimation of the mutual information," in *ICA2000*, Helsinki, Finland, June 2000, pp. 69–74.
- [31] G. Darmois, "Analyse des liaisons de probabilité," in *Proceeding Intern. Stat. conference 1947*, Washington (D.C.), 1951, vol. III A, p. 231, (in French).
- [32] G. Darmois, "Analyse générale des liaisons stochastiques," *Rev. Inst. Intern. Stat.*, vol. 21, pp. 2–8, 1953, (in French).
- [33] Carl Deboor, *A practical guide to splines*, Springer-Verlag, 1978.
- [34] G. Deco and W. Brauer, "Nonlinear higher-order statistical decorrelation by volume-conserving architectures," *Neural Networks*, vol. 8, pp. 525–535, 1995.
- [35] N. Delfosse and Ph. Loubaton, "Adaptive blind separation of independent sources: A deflation approach," *Signal Processing*, vol. 45, pp. 59–83, 1995.
- [36] N. Delfosse and Ph. Loubaton, "Adaptive blind separation of independent sources: A second order stale algorithm for the general case," *IEEE transactions on Circuits and Systems*, vol. 47, no. 7, pp. 1056–1071, July 2000.

- [37] D. L. Donoho, "On minimum entropy deconvolution," in *Proc. 2nd Applied Time Series Symp.*, Tulsa, 1980, reprinted in *Applied Time Series Analysis II*, Academic Press, New York, 1981, pp. 565-609.
- [38] J. Eriksson and V. Koivunen, "Blind identifiability of a class of nonlinear instantaneous ICA models," in *EUSIPCO*, Toulouse (France), September 2002.
- [39] R. L. Eubank, *Spline smoothing and nonparametric regression*, Dekker, 1988.
- [40] K. Fukunaga, *Introduction to statistical pattern recognition*, New York: Academic Press, 1972.
- [41] S.V. Gerven and D.V. Compernelle, "Signal separation by symmetric adaptive decorrelation: stability, convergence and uniqueness," *IEEE transactions on Signal Processing*, vol. 43, no. 7, pp. 1602-1612, July 1995.
- [42] A. Gorokhov and Ph. Loubaton, "Second-order blind identification of convolutive mixtures with temporally correlated sources: a subspace based approach," in *Proc. European Signal Processing Conf. EUSIPCO 96*, Trieste, Italia, September 1996, pp. 2093-2096.
- [43] S. Haykin, *Neural Networks - A Comprehensive Foundation*, Prentice Hall, 1998, Second edition.
- [44] J. Héroult and C. Jutten, "Space or time adaptive signal processing by neural networks models," in *Intern. Conf. on Neural Networks for Computing*, Snowbird (Utah, USA), 1986, pp. 206-211.
- [45] J. Héroult, C. Jutten, and B. Ans, "Détection de grandeurs primitives dans un message composite par une architecture de calcul neuromimétique en apprentissage non supervisé," in *Actes du Xeme colloque GRETSI*, Nice, France, 20-24 Mai 1985, pp. 1017-1022, (in French).
- [46] J. C. Holladay, "Smoothest curve approximation," *Math. Tables Aids Computation*, vol. 11, pp. 233-243, 1957.
- [47] S. Hosseini and C. Jutten, "On the separability of nonlinear mixtures of temporally correlated sources," *IEEE signal processing letters*, 2002, accepted.
- [48] D. J. Hudson, "Fitting segmented curves whose joint points have to be estimated," *Journal of the American Statistical Association*, vol. 61, no. 316, pp. 1097-1129, December 1966.

- [49] C. Jutten, *Calcul neuromimétique et traitement du signal : analyse en composantes indépendantes*, Thèse d'état ès sciences physiques, UJF-INP Grenoble, 1987, (in French).
- [50] C. Jutten, L. Nguyen Thi, E. Dijkstra, E. Vittoz, and J. Caelen, "Blind separation of sources : an algorithm for separation of convolutive mixtures," in *International Signal Processing Workshop on Higher Order Statistics*, Chamrousse, France, July 1991, pp. 273–276.
- [51] C. Jutten and A. Taleb, "Source separation: From dusk till dawn," in *ICA2000*, Helsinki, Finland, June 2000, pp. 15–26.
- [52] A. M. Kagan, Y. V. Linnik, and C. R. Rao, *Characterization Problems in Mathematics Statistics*, John Wiley & Sons, 1973.
- [53] A. M. Kagan, Y. V. Linnik, and C. R. Rao, "Extension of darmois-skitovich theorem to functions of random variables satisfying an addition theorem," *communications in statistics*, vol. 1, no. 5, pp. 471–474, 1973.
- [54] J. Karhunen, "Neural approaches to independent component analysis and source separation," in *ESANN'96, European Symposium on Artificial Neural Networks*, Bruges, Belgium, April 1996, pp. 249–266.
- [55] M. Kendall and A. Stuart, *The Advanced Theory of Statistics, Distribution Theory*, vol. 1, Griffin, 1977.
- [56] J.-L. Lacoume and P. Ruiz, "Sources identification: a solution based on cumulants," in *IEEE ASSP Workshop*, Mineapolis, USA, August 1988.
- [57] T.W. Lee, M.S. Lewicki, M. Girolami, and T.J. Sejnowski, "Blind source separation of more sources than mixtures using overcomplete representations," *IEEE Signal Processing Letters*, vol. 4, no. 4, pp. 87–90, April 1999.
- [58] U.A. Lindgren and H. Broman, "Source separation using a criterion based on second-order statistics," *IEEE Trans. Signal Processing*, pp. 1837–1850, July 1998.
- [59] O. Macchi and O. Moreau, "Self-adaptive source separation, Part I: Convergence analysis of a direct linear network controed by Héroult-Jutten algorithm," *IEEE transaction on Signal Processing*, vol. 45, no. 4, April 1997.
- [60] Z. Malouche and O. Macchi, "Adaptive unsupervised extraction of one component of a linear mixtre with a single neuron," *IEEE Transactions on Neural Networks*, vol. 9, no. 1, pp. 123–138, 1998.

- [61] A. Mansour and C. Jutten, "A direct solution for blind separation of sources," *IEEE Trans. on Signal Processing*, vol. 44, pp. 746–748, 1996.
- [62] A. Mansour, C. Jutten, and Ph. Loubaton, "Subspace method for blind separation of sources in convolutive mixtures," in *Proc. European Signal Processing Conf. EUSIPCO 96*, Trieste, Italia, September 1996, pp. 2081–2084.
- [63] A. Mansour, C. Jutten, and Ph. Loubaton, "Adaptive subspace algorithm for blind separation of independent sources in convolutive mixture," *IEEE Trans. on Signal Processing*, vol. 48, pp. 583–586, 2000.
- [64] A. Mansour, C.G. Puntonet, and N. Ohnishi, "A simple ICA algorithm based on geometrical approach," in *International symposium on signal processing and its applications (ISSPA)*, Kuala Lumpur, Malaysia, August 2001.
- [65] K. Matsuoka, M. Ohya, and M. Kawamoto, "A neural net for blind separation of nonstationary signals," *Neural Networks*, vol. 8, no. 3, pp. 411–419, 1995.
- [66] W. Härdle, *Smoothing techniques with implementation in S*, Springer-Verlag, 1991.
- [67] Hyvärinen, "Fast and robust fixed-point algorithms for independent component analysis," *IEEE Transactions on Neural Networks*, vol. 10, no. 3, pp. 626–634, 1999.
- [68] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*, John Wiley & Sons, 2001.
- [69] A. Hyvärinen and E. Oja, "A fast fixed point algorithm for independent component analysis," *Neural computation*, vol. 9, pp. 1483–1492, 1997.
- [70] A. Hyvärinen and P. Pajunen, "Nonlinear independent component analysis: Existence and uniqueness results," *Neural Networks*, vol. 12, pp. 429–439, 1999.
- [71] E. Moreau, *Apprentissage et adaptativité, séparation auto-adaptative de sources indépendantes*, Ph.D. thesis, Université de Paris-Sud, 1995.
- [72] Pin T. NG, "Smoothing spline score estimation," *SIAM journal sci. comput.*, vol. 15, no. 5, pp. 1003–1025, September 1994.
- [73] C. L. Nikias and A. P. Petropulu, *Higher-Order Spectra Analysis: A Nonlinear Signal Processing Framework*, Prentice-Hall, 1993.
- [74] E. Oja, "The nonlinear PCA learning rule in independent component analysis," *Neurocomputing*, vol. 17, pp. 25–45, 1997.

-
- [75] P. Pajunen, A. Hyvärinen, and J. Karhunen, “Non linear source separation by self-organizing maps,” in *ICONIP 96*, Hong-Kong, September 1996.
- [76] P. Pajunen and J. Karhunen, “A maximum likelihood approach to nonlinear blind source separation,” in *ICANN97*, Lausanne (Switzerland), October 1997, pp. 541–546.
- [77] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, 1991.
- [78] D. T. Pham, “Mutual information approach to blind separation of stationary sources,” in *Proceedings of ICA ’99*, Aussois, France, January 1999, pp. 215–220.
- [79] D. T. Pham, “Blind separation of instantaneous mixture of sources based on order statistics,” *IEEE Trans. on SP*, vol. 48, pp. 363–375, 2000.
- [80] D. T. Pham, “Contrast functions for blind separation and deconvolution of the sources,” in *ICA2001*, San Diego, California, December 2001, pp. 37–42.
- [81] D. T. Pham, “Estimation de la fonction score conditionnelle et l’entropie conditionnelle,” Tech. Rep., 2002, (in French).
- [82] D. T. Pham and J.-F. Cardoso, “Blind separation of instantaneous mixtures of non stationary sources,” *IEEE Transaction on Signal Processing*, vol. 49, no. 9, pp. 1837–1848, 2001.
- [83] C. Puntonet, A. Mansour, and C. Jutten, “A geometrical algorithm for blind separation of sources,” in *Actes du XVème Colloque GRETSI 95*, Juan-Les-Pins, France, Septembre 1995, pp. 273–276.
- [84] C. H. Reinsch, “Smoothing by spline functions,” *Numer. Math.*, vol. 10, pp. 177–183, 1967.
- [85] C. H. Reinsch, “Smoothing by spline functions, ii,” *Numer. Math.*, vol. 16, pp. 451–454, 1971.
- [86] S. Rickard, R. Balan, and Rosca, “Real-time time-frequency based blind source separation,” in *ICA2001*, San Diego, USA, December 2001, pp. 651–657.
- [87] J. Rosca, N. Fan, and R. Balan, “Real-time audio source separation by delay and attenuation compensation in the time domain,” in *ICA2001*, San Diego, USA, December 2001, pp. 406–411.

- [88] M. Rosenblat, “Remarks on some non-parametric estimates of a density function,” *Annals of Mathematical Statistics*, vol. 27, pp. 642–669, 1956.
- [89] G. A. F. Seber and C. J. Wild, *Nonlinear regression*, John Wiley and Sons, 1989.
- [90] B. W. Silverman, *Density estimation for statistics and data analysis*, Chapman and Hamm, 1986.
- [91] C. Simon, *Séparation aveugle des sources en mélange convolutif*, Ph.D. thesis, l’université de Marne la Vallée, Novembre 1999, (in French).
- [92] V. P. Skitovich, “Linear forms of independent random variables and the normal distribution law,” *Izvestiya Akademii Nauk SSSR. Seriya Matematicheskaya*, vol. 18, pp. 185–200, 1954, (in Russian).
- [93] J. Solé, C. Jutten, and A. Taleb, “Parametric approach to blind deconvolution of nonlinear channels,” *Neuro Computing*, 2002, accepted.
- [94] E. Sorouchyari, “Blind separation of sources, Part III: Stability analysis,” *Signal Processing*, vol. 24, no. 1, pp. 21–29, 1991.
- [95] A. Taleb, *Séparation de sources dans des mélanges post non-linéaires*, Thèse de l’INP de Grenoble, 1999, (in French).
- [96] A. Taleb and C. Jutten, “Entropy optimization, application to blind source separation,” in *ICANN*, Lausanne, Switzerland, October 1997, pp. 529–534.
- [97] A. Taleb and C. Jutten, “Batch algorithm for source separation in postnonlinear mixtures,” in *ICA ’99*, Aussois, France, January 1999, pp. 155–160.
- [98] A. Taleb and C. Jutten, “Source separation in post nonlinear mixtures,” *IEEE Transactions on Signal Processing*, vol. 47, no. 10, pp. 2807–2820, 1999.
- [99] F. J. Theis and E. W. Lang, “Maximum entropy and minimal mutual information in a nonlinear model,” in *ICA2001*, San Diego, California, December 2001, pp. 669–674.
- [100] H.L. Nguyen Thi and C. Jutten, “Blind sources separation for convolutive mixtures,” *Signal Processing*, vol. 45, pp. 209–229, 1995.
- [101] L. Tong, V. Soon, R. Liu, and Y. Huang, “AMUSE: a new blind identification algorithm,” in *Proc. ISCAS*, New Orleans, USA, 1990.
- [102] H. Valpola, “Nonlinear independent component analysis using ensemble learning: Theory,” in *ICA2000*, Helsinki, Finland, 2000, pp. 251–256.

-
- [103] H. Valpola, X. Giannakopoulos, A. Honkela, and J. Karhunen, “Nonlinear independent component analysis using ensemble learning: Experiments and discussion,” in *ICA2000*, Helsinki, Finland, 2000, pp. 351–356.
- [104] E. Weinstein, M. Feder, and A.V. Oppenheim, “Multi-channel signal separation by decorrelation,” *IEEE Transaction on Speech and Audio Processing*, vol. 1, no. 4, pp. 405–413, October 1993.
- [105] B. Widrow, J. R. Glover, J. M. MacCool, J. Kaunitz, C. S. Williams, R. H. Hearn, J. R. Zeidler, E. Dong, and R. Goodlin, “Adaptive noise cancelation: principle and applications,” *IEEE Proceedings*, vol. 63, no. 12, pp. 1692–1716, Decembre 1975.
- [106] J.-C. Wu and J. C. Principe, “Simultaneous diagonalization in the frequency domain (SDIF) for source separation,” in *ICA99*, January 1999, pp. 245–250.
- [107] H.-H. Yang, S.I. Amari, and A. Cichocki, “Information-theoretic approach to blind separation of sources in non-linear mixtures,” *Signal Processing*, pp. 291–300, February 1998.
- [108] D. Yellin and E. Weinstein, “Criteria for multichannel signal separation,” *IEEE Trans. Signal Processing*, pp. 2158–2168, August 1994.
- [109] A. Ziehe, M. Kawanabe, S. Harmeling, and K.-R. Müller, “Separation of post-nonlinear mixtures using ACE and temporal decorrelation,” in *ICA2001*, San Diego, California, December 2001, pp. 433–438.

ABSTRACT

In this thesis, Blind Source Separation (BSS) of Convolutional Post Non-Linear (CPNL) mixtures is addressed. For separating these mixtures, we have first developed new methods for separating convolutional and Post Non-Linear (PNL) mixtures. These methods are all based on the minimization of the mutual information of the outputs. For minimizing the mutual information, we first compute its “differential”, that is, its variation with respect to a small variation in its argument. This differential is then used for designing gradient based approaches for minimizing the mutual information of the outputs. These approaches can be applied for blindly separating linear instantaneous, convolutional, PNL and CPNL mixtures.

Keywords

Blind Source Separation (BSS), Independent Component Analysis (ICA), Convolutional mixtures, Post Non-Linear (PNL) mixtures, Convolutional Post Non-Linear (CPNL) mixtures, Mutual Information.

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 des 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, convolutifs, PNL et CPNL.

MOTS-CLÉS

Séparation aveugle de sources, Analyse à Composantes Indépendantes (ACI), mélanges convolutifs, mélanges Post Non-Linéaires (PNL), mélanges Convolutifs Post Non-Linéaires (CPNL), information mutuelle.
