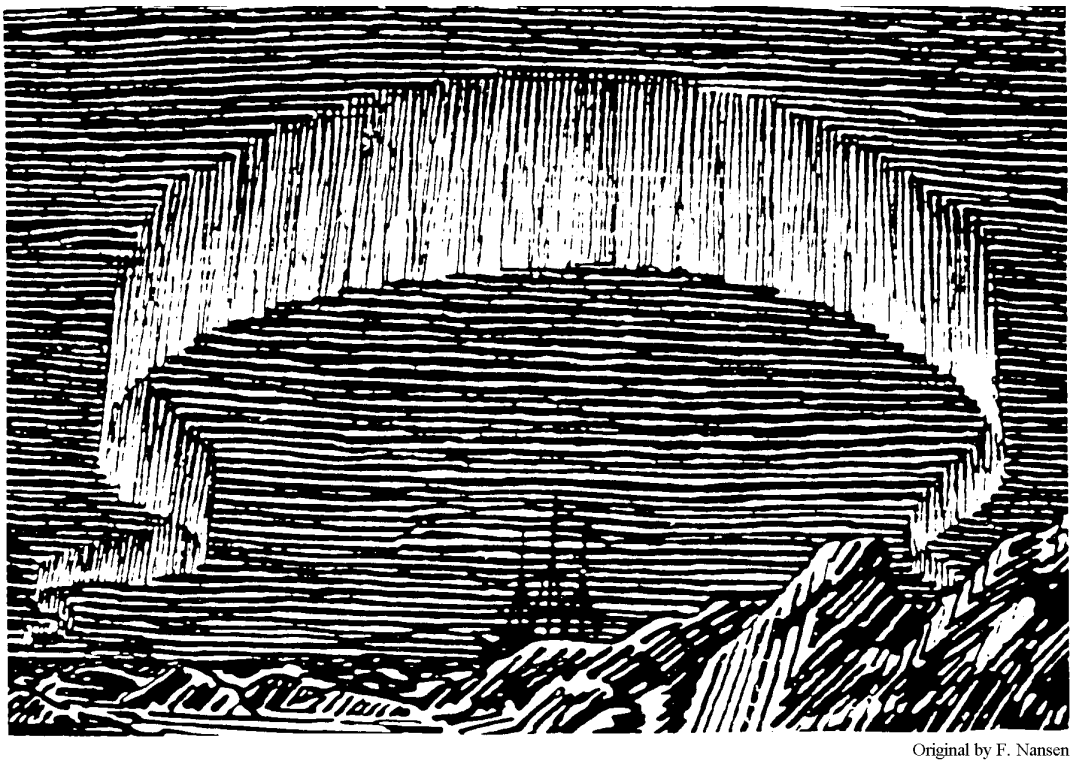
CAND. SCIENT. THESIS IN PHYSICS

A Comparative Study of Algorithms for Blind Source Separation in the Instantaneous Linear Mixture Model

Kevin Thon

June 2007



Original by F. Nansen

FACULTY OF SCIENCE
Department of Physics and Technology
University of Tromsø

Abstract

This thesis discusses some of the many techniques for performing blind source separation. Its focus is on the theoretical concepts that allow for the problem to be solved. It starts with presenting the EM algorithm, which is the method underpinning many of the algorithms that are presented later in the thesis. Some of the established methods are presented, and we proceed to devolop source separation algorithms based upon modelling the sources as scale mixtures of Gaussians. Such models are particularly well suited at modelling the super-Gaussian probability densities that characterise many real world signals, speech being perhaps the most commom.

When evaluating the performance of the algorithms in this thesis, our focus is mainly on the quality of separation, and discussions on computational efficiency are mostly superficial.

We find that in particular one of the algorithms we have constructed shows promise. Its performance is on par with existing methods, and further examination of its properties might be in order.

Acknowledgements

I would like to thank my supervisor, Professor Torbjørn Eltoft, for his never ending patience with a student who has procrastinated somewhat more than what is considered normal on completing his thesis.

Many friends and colleagues have been of invaluable help during my work on this thesis. Stian Anfinsen has made a substantial contribution to the code development. His and Anthony Doulgeris' proofing of this text, has also helped me immensely. The project has been a collaboration with the Norwegian Centre for Telemedicine, and I would like to thank in particular Vedad Hadziavdic for meaningful discussions, as well as his help and support.

Finally a massive bear hug of gratitude to the many unnamed individuals who deserve it.

Contents

Chapter 1

Introduction

1.1 Introduction

The object of study in this thesis will be the blind source separation (BSS) problem, where one seeks to recover some original sources from the available observations. The blindness refers to the fact that neither the sources nor the mixing is known. Specifically, we will concern ourselves with the basic linear mixture model, where L unknown and statistically independent source signals have been mixed by a unknown $L' \times L$ matrix $\mathbf{A}$ to yield the L' observations $\mathbf{x}$, possibly corrupted by additive noise.

$$\mathbf{x} = \mathbf{A}\mathbf{s} + \boldsymbol{\nu} \tag{1.1}$$

A wide range of problems can be formulated in such a manner. Typical examples include mixtures of simultaneous speech signals that have been recorded by several microphones, interfering radio signals arriving at a mobile phone, and even brain waves recorded by multiple sensors. For real world problems, the model in equation (1.1) will often be inadequate in its basic form. The observations will typically be corrupted by propagation delays and reverberations, and the usual way to deal with this is through a convolutive model, defining the model as in equation (1.1), but with each element of $\mathbf{A}$ representing a *filter* instead of a scalar. However, as shown in [16], it is possible to perform an *embedding* of the observations, and reformulate the problem within the confines of the basic linear model. The embedding consists of replacing the observation vector $\mathbf{x}(t) = [x_1(t), \ldots, x_{L'}(t)]^T$ by a vector $\tilde{\mathbf{x}}(t)$ consisting of a desired number M of time delayed versions of every observervation, i.e.

$$\begin{aligned}\tilde{\mathbf{x}}(t) = [&x_1(t), x_1(t-1), \ldots, x_1(t-M+1), x_2(t), x_2(t-1), \ldots, x_2(t-M+1), \\ &\ldots, x_{L'}(t), x_{L'}(t-1), \ldots, x_{L'}(t-M+1)]^T\end{aligned} \tag{1.2}$$

Defining $\tilde{\mathbf{s}}$ similarly, a convolutive model can be written as

$$\tilde{\mathbf{x}} = \tilde{\mathbf{A}}\tilde{\mathbf{s}} + \boldsymbol{\nu}' \tag{1.3}$$

where the elements of the matrix $\tilde{\mathbf{A}}$ are the coefficients of the filters (or some approximation of them) that describe the system, in some suitable order.

Although our focus in this thesis will be on BSS, it deserves mention that the techniques under study are equally applicable within the framework of *analysis* of multivariate data, where one seeks to decompose the data into independent components or factors, not necessarily representing actual physical sources. As such, the techniques can can be seen as an extension of principal component analysis and factor analysis [21].

Despite its deceptively simple problem statement, it was not until the mid 1990s that a satisfactory solution to the BSS problem was found [6][7]. Since then a myriad of different solutions have been proposed, some general and some highly specialised. A major part of the work on this thesis, has been the study and implementation of some of the existing techniques. The insights gained from these studies, has been applied to constructing a source separation algorithm where the sources are modelled as scale mixtures of Gaussians [2]. Scale mixtures of Gaussians are well suited at modelling the super-Gaussian probability densities that characterise many real world signals, speech being perhaps the most popular example. The construction of this algorithm is the main contribution of the thesis.

The performance of our algorithm will be compared to a selection of existing techinques.

1.2 Structure of thesis

Part I of this thesis is concerned with the theory underlying the BSS problem. It starts off with a chapter which is meant as an exposition of the Expectation Maximisation (EM) algorithm. This is included since the EM algorithm will be used for deriving the learning rules of the algorithms in chapters 4 through 6. Next, chapter 3 concerns Independent Component Analysis (ICA), perhaps the most popular class of BSS algorithms. There does not presently appear to be a consensus as to the presise definition of ICA, and in many texts ICA and BSS are used interchangeably. In this thesis ICA will simply refer to noiseless BSS.

Chapter 4 is devoted to Independent Factor Analysis, where the independent sources are modelled as mixtures of Gaussians (MOGs). The original

contribution of this thesis is presented in chapter 4, where we model the sources as 1-dimensional scale mixture of Gaussians and derive a source separation algorithm for such a case.

The theory part of the thesis is concluded with a chapter about Denoising Source Separation (DSS), a general framework for source separation built around denoising principles.

In part II of the thesis, the algorithms that have been presented/developed in part I, are tested on real audio signals that have been mixed artificially and then corrupted by noise. This will include a discussion around the performance of the algorithms.

Part I

Theory

Chapter 2

The Expectation-Maximisation Algorithm

2.1 Introduction

An expectation-maximisation (EM) algorithm is an algorithm for finding maximum likelihood estimates of incomplete data, originally proposed by Dempster et. al. [8]. Typically, EM is used when finding the maximum likelihood estimator of parameter vector θ based on only the observed data Y is difficult. In many applications it is possible to augment the the observed data Y with additional data Z such that the augmented likelihood $L(\theta|Y, Z)$ is easier to maximize. EM proceeds by performing an expectation (E) step, which computes an expectation of the augmented likelihood by including the latent variables Z as if they were observed, and a maximisation (M) step, which computes the maximum likelihood estimates of the parameters by maximizing the expected likelihood found on the E step. The parameters found on the M step are then used to begin another E step, and the process is repeated until convergence. In this presentation of the algorithm, I will start by presenting a proof of Jensen's inequality (to be defined), since this inequality is used both in proving that the EM algorithm increases the likelihood, and later in chapter 4 to show that the complete data likelihood is bounded from below by a chosen error function. I will then proceed to develop the EM algorithm, before briefly discussing its convergence properties.

2.2 Jensen's Inequality

Before stating and proving Jensen's Inequality, it is necessary to define what is meant by a convex and concave function. Specifically we will need to

demonstrate that the natural logarithm is a concave function, so I will print without proof a theorem to aid us in this task. (The proof is straightforward, and of no particular interest to us.)

Definition 1 *Let f be a real valued function defined on an interval $I = [a, b]$. f is said to be convex on I if $\forall x_1, x_2 \in I, \lambda \in [0, 1]$,*

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2)$$

f is said to be strictly convex if the inequality is strict.

Definition 2 *f is concave (strictly concave) if $-f$ is convex (strictly convex).*

Theorem 1 *If f(x) is twice differentiable on $[a, b]$ and $f''(x) \geq 0$ on $[a, b]$ then f(x) is convex on $[a, b]$.*

With these definitions, and the above theorem in place, it is easy to prove that the natural logarithm is a concave function:

Proposition 1 *-log (x) is strictly convex and log (x) strictly concave on (0, ∞)*

Proof: With $f(x)$ = -log (x), we have $f''(x) = \frac{1}{x^2} > 0$ for $x \in (0, \infty)$. By theorem (1), -log (x) is strictly convex on $(0, \infty)$. Also, by definition (2) log (x) is strictly concave on $(0, \infty)$.

■

Jensen's inequality expands the notion of convexity to n points.

Theorem 2 (Jensen's inequality) *Let f be a convex function defined on an interval I. If $x_1, x_2, \ldots, x_n \in I$ and $\lambda_1, \lambda_2, \ldots, \lambda_n \geq 0$ with $\sum_{i=1}^{n} \lambda_i = 1$, then*

$$f\left(\sum_{i=1}^{n} \lambda_i x_i\right) \leq \sum_{i=1}^{n} \lambda_i f(x_i)$$

Proof: For $n = 1$, λ would have to equal 1, and the result follows. For $n = 2$ the result follows from the definition of convexity. To demonstrate that the theorem is valid for all natural numbers, we apply an induction argument. Assume that the theorem is true for some n. We then need to

prove that it is true for $n+1$. At least one of the λ_i is strictly positive, say λ_{n+1}. Then,

$$\begin{aligned}
f\left(\sum_{i=1}^{n+1}\lambda_i x_i\right) &= f\left(\lambda_{n+1}x_{n+1}+\sum_{i=1}^{n}\lambda_i x_i\right) \\
&= f\left(\lambda_{n+1}x_{n+1}+(1-\lambda_{n+1})\frac{1}{1-\lambda_{n+1}}\sum_{i=1}^{n}\lambda_i x_i\right) \\
&\leq \lambda_{n+1}f(x_{n+1})+(1-\lambda_{n+1})f\left(\frac{1}{1-\lambda_{n+1}}\sum_{i=1}^{n}\lambda_i x_i\right) \\
&= \lambda_{n+1}f(x_{n+1})+(1-\lambda_{n+1})f\left(\sum_{i=1}^{n}\frac{\lambda_i}{1-\lambda_{n+1}}x_i\right) \\
&\leq \lambda_{n+1}f(x_{n+1})+(1-\lambda_{n+1})\sum_{i=1}^{n}\frac{\lambda_i}{1-\lambda_{n+1}}f(x_i) \\
&= \lambda_{n+1}f(x_{n+1})+\sum_{i=1}^{n}\lambda_i f(x_i) \\
&= \sum_{i=1}^{n+1}\lambda_i f(x_i)
\end{aligned}$$

where the first inequality follows from the definition of convexity, and the second from the induction hypothesis, which can be used since $\sum_{i=1}^{n}\frac{\lambda_i}{1-\lambda_{n+1}} = 1$.

■

Corollary: Let X be an integrable real-valued random variable and φ a measurable convex function. Then:

$$\varphi\left(E\left[X\right]\right) \leq E\left[\varphi(X)\right]$$

The proof of the corollary follows immediately for the discrete case. Simply let $\lambda_i = P(X_i)$ and sum over all the entire sample space of X and the result follows. The proof for continuous X is slightly more elaborate and will be omitted.

2.3 Developing the EM Algorithm

As stated in the introduction to this chapter, the goal of the EM algorithm is to find maximum likelihood estimates of θ from incomplete data Y. That

is, we wish to find θ such that the likelihood $P(Y|\theta)$ is a maximum. Y is here a random vector, whose distribution is parameterized by θ. It is often convenient to work with the log likelihood function which is given by

$$L(\theta) = \log P(Y|\theta)$$

Since the natural logarithm is a strictly increasing function, the value of θ that maximizes $L(\theta)$ also maximizes $P(Y|\theta)$.
The EM Algorithm is an iterative procedure for maximizing $L(\theta)$. Hence, we are interested in at each iteration finding an improved estimate of θ given the current estimate θ_n calculated in the previous iteration. That is, we wish to find an updated estimate θ such that,

$$L(\theta) > L(\theta_n)$$

Stated in a slightly different way, we wish to find a positive difference

$$L(\theta) - L(\theta_n) = \log P(Y|\theta) - \log P(Y|\theta_n) \tag{2.1}$$

We seek in the EM algorithm a method for including unobserved or missing variables Z. Noting that the total probability $P(Y|\theta)$ can be written

$$P(Y|\theta) = \sum_Z P(Y, Z|\theta) = \sum_Z P(Y|Z, \theta)P(Z|\theta)$$

we can rewrite equation (2.1) as

$$\begin{aligned}
L(\theta) - L(\theta_n) &= \log\left(\sum_Z P(Y|Z,\theta)P(Z|\theta)\right) - \log P(Y|\theta_n) \\
&= \log\left(\sum_Z P(Y|Z,\theta)P(Z|\theta)\cdot\frac{P(Z|Y,\theta_n)}{P(Z|Y,\theta_n)}\right) - \log P(Y|\theta_n) \\
&= \log\left(\sum_Z P(Z|Y,\theta_n)\frac{P(Y|Z,\theta)P(Z|\theta)}{P(Z|Y,\theta_n)}\right) - \log P(Y|\theta_n) \\
&\geq \sum_Z P(Z|Y,\theta_n)\log\left(\frac{P(Y|Z,\theta)P(Z|\theta)}{P(Z|Y,\theta_n)}\right) - \log P(Y|\theta_n) \\
&= \sum_Z P(Z|Y,\theta_n)\log\left(\frac{P(Y|Z,\theta)P(Z|\theta)}{P(Z|Y,\theta_n)P(Y|\theta_n)}\right) \\
&= \sum_Z P(Z|Y,\theta_n)\log\left(\frac{P(Y,Z|\theta)}{P(Y,Z|\theta_n)}\right)
\end{aligned}$$

$$\triangleq \Delta(\theta|\theta_n) \tag{2.2}$$

where we have used Jensen's inequality when moving the logarithm into the sum. This is valid because, being a probability measure, $P(Z|Y,\theta_n) \geq 0$ and $\sum_Z P(Z|Y,\theta_n) = 1$. Also, $\log P(Y|\theta_n) = \sum_Z P(Z|Y,\theta_n) \log P(Y|\theta_n)$, so we are justified in moving the term into the summation.
We now define the quantity

$$l(\theta|\theta_n) \triangleq L(\theta_n) + \Delta(\theta|\theta_n) \tag{2.3}$$

so, from equation (2.2)

$$L(\theta) \geq l(\theta|\theta_n) \tag{2.4}$$

Consequently, $l(\theta|\theta_n)$ is bounded from above by the likelihood function $L(\theta)$. Furthermore we have from equation (2.2) that

$$\begin{aligned} l(\theta_n|\theta_n) &= L(\theta_n) + \Delta(\theta_n|\theta_n) \\ &= L(\theta_n) + \sum_Z P(Z|Y,\theta_n) \log\left(\frac{P(Y,Z|\theta_n)}{P(Y,Z|\theta_n)}\right) \\ &= L(\theta_n) + \sum_Z P(Z|Y,\theta_n) \log 1 \\ &= L(\theta_n) \end{aligned} \tag{2.5}$$

Our overall objective is still to find the values of θ that maximize $L(\theta)$. We have now found a function $l(\theta|\theta_n)$ which is bounded above by $L(\theta)$, and which is equal to $L(\theta)$ at the current estimate θ_n. Consequently, any θ that increases $l(\theta|\theta_n)$ will necessarily increase $L(\theta)$. The EM algorithm proceeds by choosing the θ that maximizes $l(\theta|\theta_n)$ as the updated estimate θ_{n+1}. This is illustrated for a scalar θ in figure (2.1). Note that while the curve for $L(\theta)$ will be the same for every iteration of the algorithm, there will be a new realization of l for each new iteration. Considering the next iteration in figure (2.1), the curve for $l(\theta|\theta_{n+1})$ will be such that $l(\theta|\theta_{n+1})$ is equal to $L(\theta_{n+1})$. From the reasoning above, each iteration of the EM algorithm will consist of updating our estimate for θ based on the current estimate θ_n, yielding:

$$\begin{aligned} \theta_{n+1} &= \arg\max_\theta \{l(\theta|\theta_n)\} \\ &= \arg\max_\theta \left\{\sum_Z P(Z|Y,\theta_n) \log\left(\frac{P(Y,Z|\theta)}{P(Y,Z|\theta_n)}\right)\right\} \end{aligned}$$

Dropping the terms that are constant w. r. t. θ:

$$= \arg\max_\theta \left\{\sum_Z P(Z|Y,\theta_n) \log\left(P(Y,Z|\theta)\right)\right\}$$

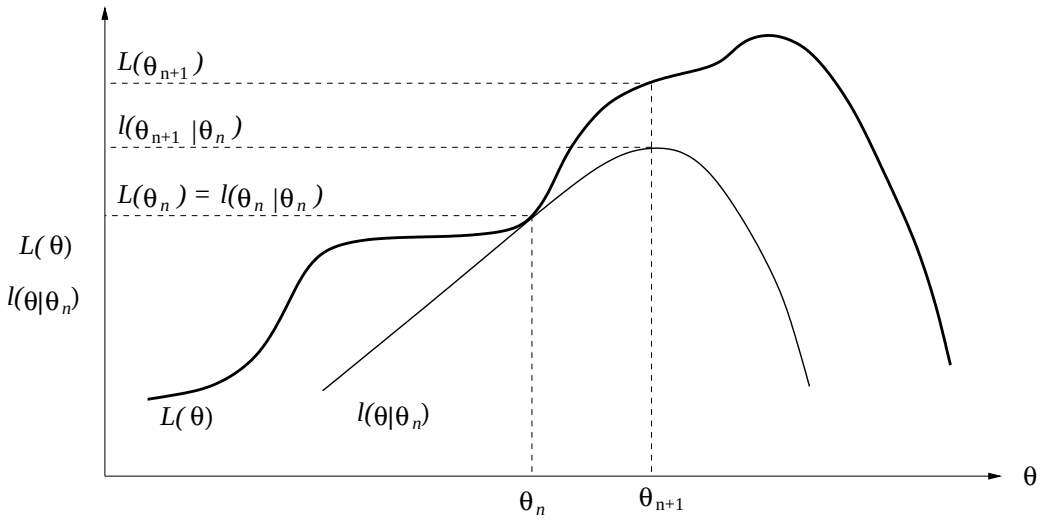


Figure 2.1: Illustration of a single iteration of the EM algorithm

$$= \arg\max_{\theta} \left\{ \mathrm{E}_{Z|Y,\theta_n} \left[\log P(Y, Z|\theta) \right] \right\} \tag{2.6}$$

We have arrived at an algorithm that has traded the direct maximisation of $L(\theta)$ for the maximisation of $l(\theta|\theta_n)$. The usefulness of such an approach lies in the fact that $l(\theta|\theta_n)$ takes into account the missing or unobserved data or variables Z. In problems where we also wish to estimate these variables, the EM algorithm thus provides a framework for doing so. Furthermore, there are situations where their introduction greatly simplifies the maximisation of $L(\theta)$. There may for example be situations where the direct maximisation of $L(\theta)$ is not feasible, whereas $l(\theta|\theta_n)$ can be made an analytically tractable function with an appropriate choice of Z.

The EM algorithm is summed up in the following section. The algorithm is just as applicable for the case of continuous random variables. To stress this, I will therefore use the continuous form.

2.4 Specification of the EM procedure

Let θ denote the quantity we wish to estimate, and let Y and Z be the observed and unobserved data, respectively. Then, given the current estimate θ_n of θ, define the function

$$l(\theta, \theta_n) = E_{Z|Y,\theta_n} \left[\log L(\theta|Y, Z) \right] = E_{Z|Y,\theta_n} \left[\log P(Y, Z|\theta) \right] \tag{2.7}$$

The algorithm then takes the form:

1. **The E-Step**: *Calculation of* $l(\theta|\theta_n)$
2. **The M-Step**: *Maximisation of* $l(\theta|\theta_n)$ *with respect to* θ

Implicit within the E-step is the estimation of the augmented data with respect to the current model estimate and the observed data Y. This estimation simply involves taking expectations with respect to the current model (conditional on the observed data).

In the preceding section I gave a slightly intuitive argument that EM always increases the likelihood $L(\theta|Y)$. A more formal proof can be given by utilizing the following inequality which holds for all densities f and g by a reformulation of Jensen's inequality:

$$E_g\left[\log\frac{f(Y)}{g(Y)}\right] \leqslant 0 \tag{2.8}$$

Letting f be $P(Z|\theta_{n+1}, Y)$ and g be $P(Z|\theta_n, Y)$, we get the following relation

$$\int_Z \log\left(\frac{P(Z|\theta_{n+1}, Y)}{P(Z|\theta_n, Y)}\right) P(Z|\theta_n, Y)\, dZ \leqslant 0 \tag{2.9}$$

The M-step of the EM algorithm consists of maximizing $l(\theta, \theta_n)$ with regard to θ. The value of θ which maximizes $l(\theta, \theta_n)$ is denoted θ_{n+1}). Hence $l(\theta_{n+1}, \theta_n) \geqslant l(\theta_n, \theta_n)$, so

$$\int_Z \log(L(\theta_{n+1}|Y, Z)P(Z|\theta_n, Y)\, dZ \geqslant \int_Z \log(L(\theta_n|Y, Z)P(Z|\theta_n, Y)\, dZ$$

$$\Downarrow$$

$$\int_Z \log\left(\frac{P(Y, Z|\theta_{n+1})}{P(Y, Z|\theta_n)}\right) P(Z|\theta_n, Y)\, dZ \geqslant 0 \tag{2.10}$$

Subtracting (2.9) from (2.10), we find

$$\int_Z \log\left(\frac{P(Y, Z|\theta_{n+1})}{P(Y, Z|\theta_n)}\frac{P(Z|\theta_n, Y)}{P(Z|\theta_{n+1}, Y)}\right) P(Z|\theta_n, Y)\, dZ \geqslant 0$$

$$\Downarrow$$

$$\int_Z \log\left(\frac{P(Z|\theta_{n+1}, Y)P(Y|\theta_{n+1})}{P(Z|\theta_{n+1}, Y)}\frac{P(Z|\theta_n, Y)}{P(Z|\theta_n, Y)P(Y|\theta_n)}\right) P(Z|\theta_n, Y)\, dZ \geqslant 0$$

$$\Downarrow$$

$$\int_Z \log\left(\frac{P(Y|\theta_{n+1}}{P(Y|\theta_n)}\right) P(Z|\theta_n, Y)\, dZ \geqslant 0$$

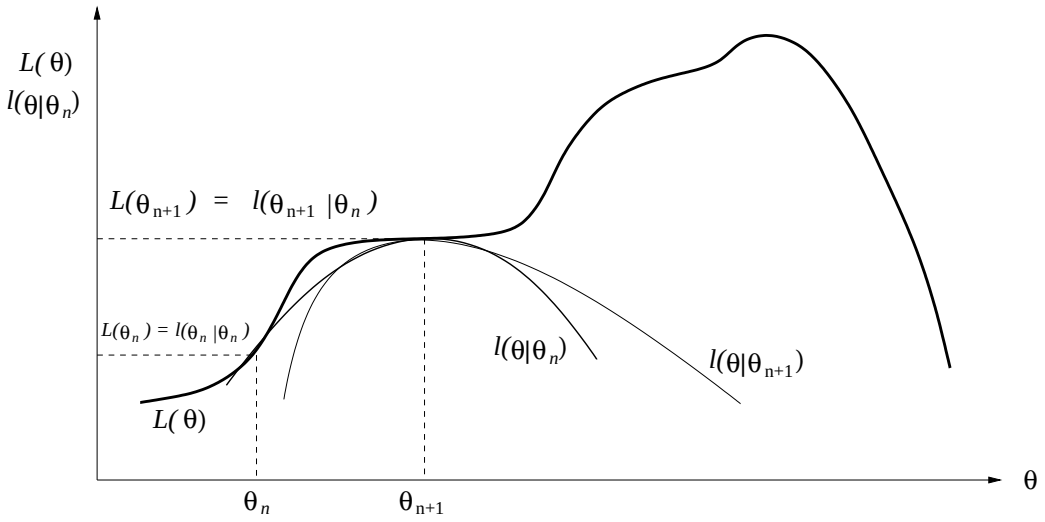


Figure 2.2: Illustration of a case where the EM algorithm converges to a saddle point of $L(\theta)$

Since the terms within the logarithm do not depend on Z, the above equation reduces to

$$\log P(Y|\theta_{n+1}) \geqslant \log P(Y|\theta_n) \tag{2.11}$$

which shows that the likelihood is nondecreasing for each step in the EM algorithm. When the algorithm reaches a fixed point for some θ_n, the value θ_n maximizes $l(\theta|\theta_n)$. Since l and L are equal at θ_n if l and L are differentiable at θ_n, then θ_n must be a stationary point of L. However, the stationary point need not be a local maximum. It is possible for the algorithm to converge to a local minimum or a saddle point in unusual cases. This situation is illustrated in figure (2.2) for a scalar θ. It is only for a unimodal $L(\theta)$ that convergence is guaranteed.

2.5 The Generalized EM Algorithm

The Generalized EM Algorithm (GEM) is a variant of the EM Algorithm where, instead of choosing the value of θ that maximizes $l(\theta|\theta_n)$ as the updated value θ_{n+1}, one is content with choosing θ_{n+1} such that $l(\theta_{n+1}|\theta_n) \geq l(\theta_n|\theta_n)$. This approach can be useful when maximisation is difficult. In some cases the GEM algorithms may even converge faster than the standard EM formulation.

Chapter 3

Independent Component Analysis

3.1 Introduction

Independent Component Analysis (ICA) is, as its name implies, a technique for finding underlying components or factors from multivariate statistical data. There are a multitude of different techniques that are proud to march under the banner of ICA, and what is common for all of them is that they search for components that are both statistically independent and non-Gaussian.

What became known as ICA was first introduced in the early 1980s by J. Hérault, C. Jutten and B. Ans [15] [19], but it was not until the mid 1990s that the technique received widespread attention, especially after A. J. Bell and T. J. Sejnowski published their work on an infomax approach to the problem [6]. Since then a wide range of different approaches have been proposed, and connections have been established between the statistical optimization criteria guiding the different techniques. An important advance was made by S. Amari [1] (and independently by J. Cardoso [7] from a different starting point), when he showed that in the space of nonsingular square matrices the natural gradient gave the direction of steepest descent of a chosen error function. The development of a fixed-point algorithm, FastICA, by A. Hyvärinen and J. Karhunen [17] has since contributed to the use of ICA to large-scale problems due to its computational efficiency.

ICA has found use in a wide variety of applications. It is well suited as a tool for feature extraction, and it is well established as a technique within

the brain imaging community. Also within the field of telecommunications ICA has proven itself useful.

In this by no means exhaustive presentation of ICA, I will focus on some of the main results and will to a large extent follow the lead given in the excellent textbook by A. Hyvärinen et. al. [16]

3.2 The Basic Model

ICA in its simplest form, is a instantaneous linear mixing model, where we assume that N statistically independent signals s_i have been mixed by the $N \times N$ mixing matrix $\mathbf{A}$ to form the N mixtures x_i:

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{3.1}$$

Here $\mathbf{x}(t) = [x_1, \ldots, x_N]^T$ and $\mathbf{s} = [s_1, \ldots, s_N]^T$. Both x_i and s_i will be understood as random variables, making $\mathbf{x}$ and $\mathbf{s}$ multivariate random variables. Now, denoting the columns of $\mathbf{A}$ by $\mathbf{a}_i$ we may write:

$$\mathbf{x} = \sum_{i=1}^{N} \mathbf{a}_i s_i \tag{3.2}$$

In addition to the independence of the signals s_i we need to impose the restriction that at most one of the independent signals has a Gaussian distribution. For the purpose of simplicity it will also be assumed that the mixing matrix is square and invertible. While this is not strictly necessary for the identifiability of the model, it simplifies the situation considerably, since the task of ICA can now be formulated as estimating the mixing matrix $\mathbf{A}$, and obtaining the independent components as $\mathbf{s} = \mathbf{A}^{-1}\mathbf{x}$.

Inherent in the ICA model are the following two ambiguities.

1. The variances of the independent components cannot be determined. This is readily seen from the form in equation (3.2). Scaling of the sources by multiplication of a scalar α_i can always be cancelled by dividing the corresponding column in $\mathbf{A}$ by the same scalar,

$$\mathbf{x} = \sum_{i=1}^{N} (\frac{1}{\alpha_i}\mathbf{a}_i)(\alpha_i s_i) \tag{3.3}$$

since neither the mixing matrix nor the signals are known. The independent components will therefore be assumed to have unit variance, $var\,[s_i^2] = 1$.

2. The order of the independent components cannot be determined.

   Again this follows from the fact that both $\mathbf{s}$ and $\mathbf{A}$ are unknown. Hence rearranging the order of of the components can be cancelled by a suitable rearrangement of $\mathbf{A}$. Letting $\mathbf{P}$ denote a permutation matrix and $\mathbf{P}^{-1}$ its inverse, we find that $\mathbf{x} = \mathbf{A}\mathbf{P}^{-1}\mathbf{P}\mathbf{s}$. $\mathbf{P}\mathbf{s}$ consists of the elements s_i of $\mathbf{s}$ but in a different order, and the matrix $\mathbf{A}\mathbf{P}^{-1}$ is simply a new unknown mixing matrix to be estimated.

Finally, it will also be assumed that both the mixture variables and the independent components have zero mean. This can be done without loss of generality, since we always can center the observed variables, say $\mathbf{x}'$, by subtracting their mean

$$\mathbf{x} = \mathbf{x}' - E[\mathbf{x}'] \tag{3.4}$$

before doing ICA, thus also making the independent variables zero mean, since

$$E[\mathbf{s}] = \mathbf{A}^{-1}E[\mathbf{x}] \tag{3.5}$$

This does not affect the mixing matrix, so after having estimated $\mathbf{A}$ for the zero mean data, the original independent components can be reconstructed by adding $\mathbf{A}^{-1}E[\mathbf{x}']$ to the zero mean independent components.

3.3 Preprocessing

Apart from the previously mentioned centering of the observations, there are a few other preprocessing steps that can be usefull. A common one for many ICA algorithms is to *whiten* or *sphere* the data.

3.3.1 Whitening

A zero mean random vector $\mathbf{z} = [z_1 \dots z_N]^T$ is said to be white if its elements z_i are uncorrelated and have unit variances:

$$E[\mathbf{z}\mathbf{z}^T] = \mathbf{I}$$

In other words, to whiten, we seek a linear transformation matrix $\mathbf{V}$ of the data $\mathbf{x}$ such that the transformed data

$$\mathbf{z} = \mathbf{V}\mathbf{x}$$

is white. Perhaps the most popular method for achieving this is through eigenvalue decomposition of the covariance matrix $E[\mathbf{x}\mathbf{x}^T] = \mathbf{E}\mathbf{D}\mathbf{E}^T$, where

$\mathbf{E}$ is the orthogonal matrix of eigenvectors of $E[\mathbf{x}\mathbf{x}^T]$ and $\mathbf{D}$ is the diagonal matrix with the corresponding eigenvalues along the diagonal, $\mathbf{D} = diag(d_1, \ldots, d_N)$. Whitening is now performed by $\mathbf{V} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^T$

$$\mathbf{z} = \mathbf{V}\mathbf{x} = \mathbf{E}\mathbf{D}^{-\frac{1}{2}}\mathbf{E}^T\mathbf{x}$$

where $\mathbf{D}^{-1/2} = diag(d_1^{-1/2}, \ldots, d_N^{-1/2})$. That this transformation indeed produces whitened data is easily verified from

$$\begin{aligned} E[\mathbf{z}\mathbf{z}^T] &= \mathbf{V}E[\mathbf{x}\mathbf{x}^T]\mathbf{V}^T = \left(\mathbf{E}\mathbf{D}^{-\frac{1}{2}}\mathbf{E}^T\right)\left(\mathbf{E}\mathbf{D}\mathbf{E}^T\right)\left(\mathbf{E}\mathbf{D}^{-\frac{1}{2}}\mathbf{E}^T\right) \\ &= \mathbf{E}\mathbf{D}^{-\frac{1}{2}}\mathbf{D}\mathbf{D}^{-\frac{1}{2}}\mathbf{E}^T = \mathbf{I} \end{aligned}$$

It must be noted that this whitening matrix is by no means unique. In fact, multiplying a whitening matrix $\mathbf{V}$ by *any* orthogonal matrix $\mathbf{U}$ will produce another whitening matrix, as seen by

$$E[\mathbf{z}\mathbf{z}^T] = \mathbf{U}\mathbf{V}E[\mathbf{x}\mathbf{x}^T]\mathbf{V}^T\mathbf{U}^T = \mathbf{U}\mathbf{I}\mathbf{U}^T = \mathbf{I}$$

Having performed whitening of the data, the task of ICA can now be viewed as finding the modified matrix $\tilde{\mathbf{A}} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^T\mathbf{A}$ given the whitened $\mathbf{z}$, since

$$\mathbf{z} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^T\mathbf{x} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^T\mathbf{A}\mathbf{s} = \tilde{\mathbf{A}}\mathbf{s}$$

The usefulness of this lies in the fact that we can now restrict our search for mixing matrixes to orthogonal matrixes. Since the s_i are independent and with unit variance

$$E[\mathbf{z}\mathbf{z}^T] = \tilde{\mathbf{A}}E[\mathbf{s}\mathbf{s}^T]\tilde{\mathbf{A}}^T = \tilde{\mathbf{A}}\tilde{\mathbf{A}}^T = \mathbf{I}$$

Consequently the number of parameters to be estimated is greatly reduced, since instead of having to estimate the n^2 parameters that are the elements of the original matrix $\mathbf{A}$, we need only find the orthogonal matrix $\tilde{\mathbf{A}}$. An $n \times n$ orthogonal matrix has $n(n-1)/2$ degrees of freedom, so the number of parameters to be estimated is more than halved.

Concluding this section on whitening, the connection between whitening and Principal Component Analysis (PCA) is interesting to note. Firstly, PCA techniques are often used to find the eigenvectors and eigenvalues needed to do the whitening. Furthermore, PCA is commonly used to perform a dimension reduction when the number of observations is large. This is done through the standard technique of setting a limit below which the eigenvalues, and hence the principal components, are deemed insignificantly small and are hence disregarded. It is assumed that the reader is familiar with PCA, so the details of the process are omitted. For a overview of PCA see e.g. [16].

3.3.2 Further Preprocessing

For noisy data some filtering may be desirable. It turns out that linear filtering of the sensor signals $x_i(t)$ does not influence the ICA model. Letting $\mathbf{X}$ be a matrix having the observations $\mathbf{x}(1), \ldots, \mathbf{x}(N)$ as its columns, and letting $\tilde{\mathbf{X}}$ be its filtered version, the ICA model can now be represented as

$$\mathbf{X} = \mathbf{AS}$$

where $\mathbf{S}$ is a matrix consisting of the sources. Any linear filtering of $\mathbf{X}$ can be representet as a multiplication from the right by a filtering matrix $\mathbf{F}$. Accordingly

$$\tilde{\mathbf{X}} = \mathbf{XF} = \mathbf{ASF} = \mathbf{A}\tilde{\mathbf{S}}$$

so we see that the model still holds, with the same mixing matrix $\mathbf{A}$. In fact we retrieve a filtered version of the original sources.

3.4 The Natural Gradient

Before proceeding to present some of the techniques used to perform ICA, a few words about the optimization methods are warrented. Common to all methods of performing ICA is that they proceed by optimizing some objective function. This is normally performed by gradient techniques, attempting to increment the solution in the direction of steepest descent/ascent. In a Euclidian orthogonal coordinate system, the gradient indeed gives the direction of steepest descent, but as Amari [1] makes clear, the parameter space where the optimization takes place in ICA, is that of nonsingular square matrixes, of dimesion say $n \times n$. This space is *not* Euclidian, but rather has a Riemannian structure which leads to the calculation of what has been termed *the natural gradient*, to give the direction of steepest descent in such a space.

Now, in an optimization scheme for minimizing an objective function $\mathcal{J}$, we wish to find the direction for a small increment $\delta\mathbf{W}$ such that the value $\mathcal{J}(\mathbf{W} + \delta\mathbf{W})$ is minimized. $\mathbf{W}$ represents the point at the current iteration. We further require that the squared norm $\|\delta\mathbf{W}\|^2$ is constant, hence constraining the length of the step to be constant. The squared norm is defined as a weighed matrix inner product

$$\|\delta\mathbf{W}\|^2 = \langle \delta\mathbf{W}, \delta\mathbf{W} \rangle_{\mathbf{W}}$$

such that

$$\langle \delta\mathbf{W}, \delta\mathbf{W} \rangle_{\mathbf{I}} = \sum_{i,j=1}^{n} (\delta w_{ij})^2 = \mathrm{Tr}[\delta\mathbf{W}^T \delta\mathbf{W}]$$

In [18] Amari makes the case that because of the Riemannian structure of the matrix space, the following invariance should apply to this inner product:

$$\langle \delta\mathbf{W}, \delta\mathbf{W} \rangle_{\mathbf{W}} = \langle \delta\mathbf{W}\mathbf{M}, \delta\mathbf{W}\mathbf{M} \rangle_{\mathbf{W}\mathbf{M}}$$

for any matrix $\mathbf{M}$. Putting $\mathbf{M} = \mathbf{W}^{-1}$, gives

$$\langle \delta\mathbf{W}, \delta\mathbf{W} \rangle_{\mathbf{W}} = \langle \delta\mathbf{W}\mathbf{W}^{-1}, \delta\mathbf{W}\mathbf{W}^{-1} \rangle_{\mathbf{I}} = \mathrm{Tr}[(\mathbf{W}^T)^{-1}\delta\mathbf{W}^T\delta\mathbf{W}\mathbf{W}^{-1}]$$

Now, keeping the inner product thus defined constant, Amari showed that the largest increment in $\mathcal{J}(\mathbf{W} + \delta\mathbf{W})$ is in the direction of the natural gradient, as defined by:

$$\frac{\partial \mathcal{J}}{\partial \mathbf{W}}_{nat} = \frac{\partial \mathcal{J}}{\partial \mathbf{W}}\mathbf{W}^T\mathbf{W} \tag{3.6}$$

The natural gradient, as will become apparrant in the subsequent sections, not only gives the direction of steepest descent, but actually reduces the amount of calculations necessary to compute the update in many optimization schemes, since it renders unnecessary the calculation of inverses.

3.5 ICA by Maximisation of Non-Gaussianity

Maximation of non-Gaussianity is an intuitively pleasing method for doing ICA. The basic thought is that any mixing of signals will produce an output which is more Gaussian than the original signals, and hence the objective in achieving separation should be to maximize non-Gaussianity. Justification of this can be found in the Central Limit Theorem, which states that the distribution of a sum of independent random variables tends towards a Gaussian distribution.

From equation (3.1), it is obvious that the independent components can be written in terms of the observations as

$$\mathbf{s} = \mathbf{A}^{-1}\mathbf{x} \tag{3.7}$$

Estimating an independent component can thus be seen as the search for an appropriate linear combination of the observations, which we will denote $y = \mathbf{b}^T\mathbf{x}$, where $\mathbf{b}$ is a vector to be determined. This may of course be written in terms of the sources as $y = \mathbf{b}^T\mathbf{A}\mathbf{s}$. Accordingly, y is a linear combination of the s_i, with coefficients given by $\mathbf{q}^T = \mathbf{b}^T\mathbf{A}$, so

$$y = \mathbf{b}^T\mathbf{x} = \mathbf{q}^T\mathbf{s} = \sum_i q_i s_i \tag{3.8}$$

The goal of ICA can be stated as finding $\mathbf{b}$ such that y equals one of the sources. This is obviously achieved with $\mathbf{b}$ equal to a row in the inverse of $\mathbf{A}$, making $\mathbf{q}$ a vector with one of its elements equal to one, and the rest equal to zero.
Regarding now the distribution of $y = \mathbf{q}^T\mathbf{s}$, the basic idea for achieving separation, is that the sum of independent variables will be more Gaussian than the original variables. Hence y will be maximally non-Gaussian when $\mathbf{q}$ is a vector of zeros and a single 1, corresponding to y being equal to one of the sources. Accordingly, the goal should be to look for the $\mathbf{b}$ that maximises the non-Gaussianity of $y = \mathbf{b}^T\mathbf{x}$.

3.5.1 Measures of Non-Gaussianity

Having justified the maximisation of non-Gaussianity as a reasonable course of action for performing ICA, we now need to look at some quantative measures of non-Gaussianity. The perhaps most common measure in this respect is the *kurtosis*

Kurtosis For a zero mean random variable y, kurtosis is defined as

$$\text{kurt}(y) = E[y^4] - 3(E[y^2])^2 \tag{3.9}$$

Assuming that y has been normalized such that its variance is equal to one, (3.9) may be simplified further to $E[y^4] - 3$. For a Gaussian random variable kurtosis is equal to zero, whilst for most non-Gaussian variables it is nonzero. Kurtosis can be both positive and negative. Random variables that have a negative kurtosis are called sub-Gaussian, and those with positive kurtosis are called super-Gaussian. Super-Gaussian random variables will typically have a sharply peaked probability density function (pdf) with heavy tails, whilst the pdf of a sub-Gaussian random variable will typically be flat, in the sense that it is rather constant for some band near zero and drops sharply to near zero for larger values.

Kurtosis is a perfectly viable measure of non-Gaussianity, and could be used as a basis of developing algorithms for performing ICA (as indeed has been done, see e.g. [27]). However, there are some practical drawbacks to using kurtosis stemming from the fact that, in any practical implementation, it has to be estimated from a measured sample. The main problem is that moment based estimates of kurtosis are extremely sensitive to outliers, meaning that a single large (possibly erroneous) value will greatly affect the estimate. This lack of robustness leads us to consider another measure og non-Gaussianity, namely *negentropy*.

Negentropy This is an information theoretic measure, based on the concept of differential entropy. Stated simply, the (differential) entropy of a random variable is related to the amount of information that the observation of the variable gives. The more random a variable is, the larger the entropy. For a more comprehensive overview of information theoretic concepts see e.g. [14] or [16]. Formally, the entropy H of a random variable is defined as:

$$H(\mathbf{y}) = -\int p_y(\boldsymbol{\eta}) \log p_y(\boldsymbol{\eta})\, d\boldsymbol{\eta} \tag{3.10}$$

Of all random variables of equal variance, it turns out that a Gaussian variable has the largest entropy. Accordingly the negentropy J of a random variable $\mathbf{y}$, which is defined as

$$J(\mathbf{y}) = H(\mathbf{y}_{gauss}) - H(\mathbf{y}) \tag{3.11}$$

will be nonnegative, and zero only if $\mathbf{y}$ has a Gaussian distribution. $\mathbf{y}_{gauss}$ is a Gaussian random variable with the same covariance matrix as $\mathbf{y}$.
The superiority of negentropy as a measure of non-Gaussianity is argued for convincingly in [16], and here we will content ourselves with presenting the estimators of negentropy used in aforementioned reference. Here negentropy is approximated as

$$J(y) \propto [E[G(y)] - E[G(\nu)]]^2 \tag{3.12}$$

where ν is a Gaussian variable of zero mean and unit variance, and G can be practically any nonquadratic function, chosen so as to give robust estimators. This entails choosing a G that does not grow to fast, and the following two choises of G are suggested:

$$G_1(y) = \frac{1}{a_1} \log \cosh a_i y \tag{3.13}$$

$$G_2(y) = -\exp(-y^2/2) \tag{3.14}$$

3.5.2 A Fixed Point Algorithm

In this section it will be shown how a *fixed point* algorithm (to be defined) can be used to perform ICA. It will be assumed throughout that the observations have been centered and whitened. The starting point will be a standard gradient technique.

As argued in the introduction to section 3.5, our goal is to find the linear conbination of the observations that maximises non-Gaussianity, that is, the $\mathbf{w}$ that maximises the non-Gaussianity of $\mathbf{w}^T\mathbf{x}$. Since, by assumption $\mathbf{x}$ is

white, we can immediately see that $E[\mathbf{w}^T\mathbf{x}] = \|\mathbf{w}\|^2$. Since we desire the separated components to have unit variance, this expectation must equal unity (see section (3.3.1)). Accordingly, we have a constrained optimisation problem, of maximising equation (3.12) w.r.t. $\mathbf{w}$ under the constraint that $\|\mathbf{w}\|^2 = 1$.

Now, taking the gradient of the approximation of negentropy in (3.12), yields the following algorithm

$$\Delta\mathbf{w} \propto \gamma E[\mathbf{x}g(\mathbf{w}^T\mathbf{x})] \tag{3.15}$$

$$\mathbf{w} \leftarrow \mathbf{w}/\|\mathbf{w}\| \tag{3.16}$$

where $\gamma = E[G(y)] - E[G(\nu)]$, ν still being a standardized gaussian random variable. g is the derivative of G. The normalization ensures that the constraint $\|\mathbf{w}\| = 1$ is fulfilled.

It is of possible to use this algortihm to obtain the independent components, but a more effective way is by performing *fixed point iteration*. A fixed point x of a function f is a point where $f(x) = x$. Fixed point iteration is a method for finding stable fixed points, which starts with an initial value x_0, and iterates to find $x_{n+1} = f(x_n)$. To derive a fixed point iteration scheme for our problem, note that at a stable point of the gradient algorithm for finding $\mathbf{w}$, the gradient must point in the direction of $\mathbf{w}$. In other words the gradient must equal $\mathbf{w}$ multiplied by some scalar constant. This is evident from the fact that when converged, then adding the gradient to $\mathbf{w}$ does not change its direction. After normalisation the gradient will actually be equal to $\mathbf{w}$ (with a possible and irrelevant change of sign). Hence a natural candidate for fixed point iteration is

$$\mathbf{w} \leftarrow E[\mathbf{x}g(\mathbf{w}^T\mathbf{x})] \tag{3.17}$$

In [16] however, they find that updating $\mathbf{w}$ as

$$\mathbf{w} \leftarrow E\left[\mathbf{x}g(\mathbf{w}^T\mathbf{x}) - E[g'(\mathbf{w}^T\mathbf{x}]\right] \tag{3.18}$$

has vastly improved convergence properties, and this is the update used in the FastICA package, which is of interest in this thesis. Equation (3.18) is arrived at by noting that one can add $\mathbf{w}$ multiplied by some constant α to both sides of equation (3.17) without changing the fixed points. They then proceed by finding the update in terms of $\mathbf{w}$ by way of Newton's method, using some approximations.

So far we have showed how to find *one* independent component (IC). To find further ICs we utilize the fact that the vectors $\mathbf{w}_i$ corresponding to

different ICs, are orthogonal. Thus, to estimate several ICs, we need to run the one-unit algorithm several times, and for each new $\mathbf{w}_i$ being iterated on, ensure that it is orthogonal to the previously found $\mathbf{w}$. This is easily achieved using the Gram-Schmidt method (see e.g. [24]). Finally, to conclude this chapter, the FastICA algorithm can be summed up neatly in the following table from [16]:

1. Center the data to make its mean zero
2. Whiten the data
3. Choose m, the number of ICs to estimate. Set counter $p \leftarrow 1$
4. Choose an initial (random) value of unit norm for $\mathbf{w}_p$.
5. Let $\mathbf{w}_p \leftarrow E\left[\mathbf{x}g(\mathbf{w}^T\mathbf{x}) - E[g'(\mathbf{w}^T\mathbf{x}]\right]$
6. Do the following orthogonalisation:
$$\mathbf{w}_p \leftarrow \mathbf{w}_p - \sum_{j=1}^{p-1}(\mathbf{w}_p^T\mathbf{w}_j)\mathbf{w}_j \tag{3.19}$$
7. Let $\mathbf{w}_p \leftarrow \mathbf{w}_p/\|\mathbf{w}\|$.
8. If $\mathbf{w}_p$ has not converged, go back to step 5.
9. Set $p \leftarrow p + 1$. If $p \leq m$, go back to step 4.

3.6 The Maximum Likelihood Approach

The well known maximum likelihood (ML) method has been applied successfully in the ICA problem. While providing an elegant solution to ICA, a drawback with the technique is that it requires some knowledge or assumptions of the probability densities of the independent components. It turns out, however, that approximation of the densities can be done by a family of densities that are spesified by a limited number of parameters.

3.6.1 The likelihood function

Given the ICA model in equation (3.1) it is a well known result from probability theory (see e.g. [16]) that the density of this linear transformation is given by
$$p_{\mathbf{x}}(\mathbf{x}) = \frac{1}{|\mathbf{A}|}p_{\mathbf{s}}(\mathbf{A}^{-1}\mathbf{x}) \tag{3.20}$$

Denoting the inverse of the mixing matrix as $\mathbf{W} = \mathbf{A}^{-1}$ and letting p_i denote the density of the ith independent component, we may write

$$p_{\mathbf{x}}(\mathbf{x}) = |\mathbf{W}| \prod_i p_i(\mathbf{w}_i^T \mathbf{x}) \tag{3.21}$$

where $\mathbf{w}_i$ is the ith row of $\mathbf{W}$. Now assuming we have T observations of $\mathbf{x}$ which we denote by $\mathbf{x}(1), \ldots, \mathbf{x}(T)$, then the likelihood is given by

$$L(\mathbf{W}) = \prod_{t=1}^{T} p_{\mathbf{x}}(\mathbf{x}(t)) = \prod_{t=1}^{T} \prod_{i=1}^{N} p_i(\mathbf{w}_i^T \mathbf{x}(t)) |\mathbf{W}| \tag{3.22}$$

where we have made the standard assumption that the observations at any two different time instances are independent of each other. Finally, denoting the log-likelihood $l(\mathbf{W})$, we find that

$$l(\mathbf{W}) = \sum_{t=1}^{T} \sum_{i=1}^{N} \log p_i(\mathbf{w}_i^T \mathbf{x}(t)) + T \log |\mathbf{W}| \tag{3.23}$$

which may be rewritten

$$\frac{1}{T} l(\mathbf{W}) = E\left[\sum_{i=1}^{N} \log p_i(\mathbf{w}_i^T \mathbf{x}(t)) \right] + \log |\mathbf{W}| \tag{3.24}$$

where $E[\cdot]$ here denotes the sample average. Before using this expression in an ICA scheme, some assumptions need to be made on the densities.

3.6.2 Estimation of the Densities

As shown in [16] it suffices to use only two approximations of the density of an independent component, one for supergaussian densities and one for subgaussian densities. The justification of this comes from the following theorem also from [16]:

Theorem 3 *Denote by $\tilde{p}_i$ the assumed densities of the independent components, and*

$$g_i(s_i) = \frac{\partial}{\partial s_i} \log \tilde{p}_i(s_i) = \frac{\tilde{p}_i'(s_i)}{\tilde{p}_i(s_i)} \tag{3.25}$$

Constrain the estimates of the independent components $y_i = \mathbf{w}_i^T \mathbf{x}$ to be uncorrelated and to have unit variance. Then the ML estimator is locally consistent, if the assumed densities $\tilde{p}_i$ fulfill

$$E[s_i g_i(s_i) - g'(s_i)] > 0 \tag{3.26}$$

for all i.

This theorem tells us that small misspecifications in the densities $\tilde{p}_i$ do not affect the local consistency of the ML estimator, since sufficiently small changes do not change the sign in equation (3.26). Also the theorem aids us in the construction of the two densities needed. Simply choose two densities such that the condition in theorem 3 is always fulfilled for only one of them. As an example, consider the following log-densities:

$$\log \tilde{p}_i^+ = \alpha_1 - 2\log\cosh(s) \tag{3.27}$$

$$\log \tilde{p}_i^- = \alpha_2 - [s^2/2 - \log\cosh(s)] \tag{3.28}$$

where α_1, α_2 are normalizing parameters that ensure that $\tilde{p}_i^+$ and $\tilde{p}_i^-$ are probability densities. Here $\tilde{p}_i^+$ represents a super-Gaussian density, and $\tilde{p}_i^-$ a sub-Gaussian density. Inserted into equation (3.26) we find for $\tilde{p}_i^+$

$$2E[-\tanh(s_i)s_i + (1 - \tanh(s_i)^2)] \tag{3.29}$$

and for $\tilde{p}_i^-$

$$E[\tanh(s_i)s_i + (1 - \tanh(s_i)^2)] \tag{3.30}$$

Clearly the signs of these expressions are always opposite (except when both equal zero), so for practically any distribution of the s_i, one of these distributions fulfill the condition in theorem 3. Hence, in a practical implementation, one can estimate the densities of the sources as being either $\tilde{p}_i^+$ or $\tilde{p}_i^-$.

3.6.3 A Gradient Algorithm

The obvious method for maximizing the likelihood is by a gradient method. Computing the gradient of (3.24) yields the following expression:

$$\frac{1}{T}\frac{\partial l}{\partial \mathbf{W}} = [\mathbf{W}^T]^{-1} + E[\mathbf{g}(\mathbf{W}\mathbf{x})\mathbf{x}^T] \tag{3.31}$$

where $\mathbf{g}(\mathbf{y}) = (g_i(y_i), \ldots, g_N(y_N))$ is a component-wise vector function that consists of the *score* functions g_i of the distributions of s_i, defined as

$$g_i = (\log p_i)' = \frac{p_i'}{p_i} \tag{3.32}$$

This immediately gives rise to the following update for the de-mixing matrix $\mathbf{W}$:

$$\Delta\mathbf{W} \propto [\mathbf{W}^T]^{-1} + E[\mathbf{g}(\mathbf{W}\mathbf{x})\mathbf{x}^T] \tag{3.33}$$

Now, remembering that the direction of steepest descent is given by the natural gradient, this algorithm may be modified using equation (3.6), to yield

$$\Delta\mathbf{W} \propto (\mathbf{I} + E[\mathbf{g}(\mathbf{W}\mathbf{x})\mathbf{x}^T])\mathbf{W} \tag{3.34}$$

Here the benefit of using the natural gradient becomes clear. While calculation of the standard gradient requires the calculation of the inverse matrix at each update, this is avoided when using the natural gradient. In addition, of course, comes the faster convergence.

To complete the spesification of the algorithm, one must choose two densities, one representing super-Gaussian sources and one representing sub-Gaussian. Here we choose the densities from equations (3.27) and (3.28), giving us the following score functions:

$$g^+(y) = -2\tanh(y) \tag{3.35}$$
$$g^-(y) = \tanh(y) - y \tag{3.36}$$

To decide which of the two nonlinearities in (3.35) and (3.36) to use, one can calculate the expectation

$$E[-\tanh(s_i)s_i + (1 - \tanh(s_i)^2)] \tag{3.37}$$

If this is positive, then use (3.35), otherwise use (3.36). The expectation must of course be calculated using the estimate of the sources available at the current iteration. The ML approach to ICA may finally be summed up in the following points:

1. Center and whiten the data
2. Choose an initial separating matrix $\mathbf{W}$, and choose initial values of $\gamma_i, i = 1, \ldots, N$, either randomly or using prior information. Set the learning rates μ and μ_γ.
3. Compute $\mathbf{y} = \mathbf{W}\mathbf{x}$
4. If the nonlinearities are not fixed a priori
   (a) update $\gamma_i = (1 - \mu_\gamma)\gamma_i + \mu_\gamma E[-\tanh(y_i)y_i + (1 - \tanh(y_i)^2)]$.
   (b) if $\gamma_i > 0$, define g_i as in (3.35), otherwise define it as in (3.36).
5. Update the separating matrix by
   $$\mathbf{W} \leftarrow \mathbf{W} + \mu(\mathbf{I} + E[\mathbf{g}(\mathbf{W}\mathbf{x})\mathbf{x}^T])\mathbf{W} \tag{3.38}$$
   where $\mathbf{g}(\mathbf{y}) = (g_i(y_i), \ldots, g_N(y_N))^T$.
6. If not converged, go back to step 3.

3.7 Other Methods

In this presentation we have focused on two common techniques for performing ICA. It must be noted however, that there exist a vast number of other approaches to the problem. Indeed, the techniques presented in the following chapters, can readily be seen as falling under the ICA umbrella.

Of the methods not mentioned in this thesis, perhaps the most popular is ICA by minimisation of mutual information. This technique is appealing, since mutual information is a natural information-theoretic measure of dependence, and hence its minimisation is a natural objective for achieving separation. Also it is a usefull framework for comparison of the various ICA methods (see e.g. [22]).

In short, the field of ICA is large and growing. Among the main challenges is the development of ICA scemes that take into account the presence of additive noise. The techniques presented i the following chapters are an attempt at achieving this.

Chapter 4

Independent Factor Analysis

4.1 Introduction

Independent factor analysis (IFA) is yet another method for recovering independent sources from their observed mixtures. The method was proposed by H. Attias in his article Independent Factor Analysis [4]. Its starting point is the standard linear mixing model, but in contrast with standard ICA methods, the noise is explicitly taken into account through estimation of the noise covariance. As its name implies, IFA can be reduced to either factor analysis (FA), principal component analysis (PCA) or ICA: When the sources are Gaussian, IFA performs ordinary FA, whilst in the zero-noise limit standard IFA reduces to an EM algorithm for PCA. A separate EM algorithm has therefore been derived for the zero noise case, which basically performs ICA with the nonlinearity in the learning rule for the separating matrix being a sum of the state posteriors, scaled by the data and the model parameters.

The IFA algorithm consists of two separate stages. In the first stage an EM algorithm is employed to perform unsupervised learning of both the mixing matrix and the parameters of the probabilistic model describing the noise and the signals being mixed. The sources are modeled as a mixture of Gaussians, allowing all the probabilistic calculations to be performed analytically. The second stage is the reconstruction of the sources by an optimal nonlinear estimator, in practise a Least Mean Square (LMS) or Maximum A posteriori (MAP) estimate, based on the data and the parameters learned in the first stage.

Besides the blind separation application, IFA has been proposed for modeling multidimensional data by a highly constrained mixture of Gaussians and

as a tool for nonlinear signal encoding.

4.2 The Model

The basic model we are considering is again the case of L independent signals $\mathbf{s}$ being mixed linearly by $\mathbf{A}$ (not necessarily square) and corrupted by additive noise $\boldsymbol{\nu}$, producing the L' mixtures $\mathbf{x}$. $\mathbf{s}$ are referred to as the source signals and $\mathbf{x}$ the sensor signals.

$$\mathbf{x} = \mathbf{A}\mathbf{s} + \boldsymbol{\nu} \tag{4.1}$$

We are interested in finding an expression for the probability density of the sensor signals $p(\mathbf{x})$. To do so we model the sources x_i as L independent random variables with arbitrary distributions $p(x_i|\theta)$, where θ_i is the parameter set parameterising the ith source density. Furthermore we assume that the noise is Gaussian with zero mean and covariance matrix Λ. Λ is not necessarily a diagonal matrix, thus allowing correlations between sensors. Such correlations may be present due to source noise or propagation noise.Therefore,

$$p(\boldsymbol{\nu}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}) \tag{4.2}$$

The model sensor densities can be written in terms of the source densities and their parameters, in addition to the mixing matrix $\mathbf{A}$ and the noise covariance Λ. These terms will henceforth be referred to collectively as W

$$W = (\mathbf{A}, \boldsymbol{\Lambda}, \theta) \tag{4.3}$$

$$\begin{aligned} p(\mathbf{x}|W) &= \int p(\mathbf{x}|\mathbf{s})p(\mathbf{s})\, d\mathbf{s} \\ &= \int \mathcal{N}(\mathbf{A}\mathbf{s}, \boldsymbol{\Lambda}) \prod_{i=1}^{L} p(s_i|\theta_i)\, d\mathbf{s} \end{aligned} \tag{4.4}$$

The goal of IFA is to adapt the parameters W to minimise an error function that measures the distance between $p(\mathbf{x}|W)$ and the observed sensor densities. To facilitate in these calculations, the probability density functions of the sources $p(s_i|\theta_i)$ are modeled as mixtures of Gaussians (MOG). This has several advantages, the most gratifying being that all calculations can be done analytically. Also, such a MOG model can approximate any source density with an appropriate number of mixture elements.

4.2.1 The Source Model

Formally, the source model is defined as follows: For each source s_i, the density is described as a mixture of n_i Gaussians $q_i = 1, \ldots, n_i$ with means μ_{i,q_i}, variances ν_{i,q_i}, and mixing proportions w_{i,q_i}:

$$p(s_i|\theta_i) = \sum_{q_i=1}^{n_i} w_{i,q_i}\mathcal{N}(\mu_{i,q_i}, \nu_{i,q_i}), \qquad \theta_i = \{w_{i,q_i}, \mu_{i,q_i}, \nu_{i,q_i}\} \tag{4.5}$$

For this description to describe a genuine probability density function (pdf) the mixing proportions for each source must sum to unity: $\sum_{q_i=1}^{n_i} w_{i,q_i} = 1$. To gain some insight into the generative probabilistic model in question, I refer to figure (4.1). To generate signal s_i, pick a state q_i with probability w_{i,q_i}, and then draw a number from the corresponding Gaussian density $\mathcal{N}(\mu_{i,q_i}, \nu_{i,q_i})$.

An important insight is gained when one studies the joint source density $p(\mathbf{s})$. By the assumption of independence we have that

$$\begin{aligned}
p(\mathbf{s}|\boldsymbol{\theta}) &= \prod_{i=1}^{L} p(s_i|\theta_i) = \prod_{i=1}^{L}\sum_{q_i=1}^{n_i} w_{i,q_i}\mathcal{N}(\mu_{i,q_i}, \nu_{i,q_i}) \\
&= (w_{1,1}\mathcal{N}(\mu_{1,1}, \nu_{1,1}) + w_{1,2}\mathcal{N}(\mu_{1,2}, \nu_{1,2}) + \cdots + w_{1,n}\mathcal{N}(\mu_{1,n_1}, \nu_{1,n_1})) \\
&\quad \times (w_{2,1}\mathcal{N}(\mu_{2,1}, \nu_{2,1}) + w_{2,2}\mathcal{N}(\mu_{2,2}, \nu_{2,2}) + \cdots + w_{2,n}\mathcal{N}(\mu_{2,n_2}, \nu_{2,n_2})) \\
&\qquad\qquad \vdots \\
&\quad \times (w_{L,1}\mathcal{N}(\mu_{L,1}, \nu_{L,1}) + w_{L,2}\mathcal{N}(\mu_{L,2}, \nu_{L,2}) + \cdots + w_{L,n}\mathcal{N}(\mu_{L,n_L}, \nu_{L,n_L})) \\
&= \sum\left(\prod_{i=1}^{L} w_{i,q_i}\prod_{i=1}^{L}\mathcal{N}(\mu_{i,q_i}, \nu_{i,q_i})\right)
\end{aligned}$$

The summation above is to be understood as a sum over all the $\prod_{i=1}^{L} n_i$ combinations of the individual Gaussians. To help clarify matters the following quantity is introduced:

$$\mathbf{q} = (q_1, \ldots, q_L) \tag{4.6}$$

$\mathbf{q}$ is a vector whose entries are all the possible combinations of the individual source states q_i. As such $\mathbf{q}$ will have $\prod_{i=1}^{L} n_i$ different realizations. Each $\mathbf{q}$ realises a different combination of the individual μ_{i,q_i}, ν_{i,q_i}, and w_{i,q_i}. Define:

$$\begin{aligned}
w_{\mathbf{q}} &= \prod_{i=1}^{L} w_{i,q_i} = w_{1,q_1} \times \cdots \times w_{L,q_L}, \qquad \boldsymbol{\mu}_{\mathbf{q}} = (\mu_{1,q_1}, \ldots, \mu_{L,q_L}) \\
\mathbf{V}_{\mathbf{q}} &= \mathrm{diag}(\nu_{1,q_1}, \ldots, \nu_{L,q_L})
\end{aligned} \tag{4.7}$$

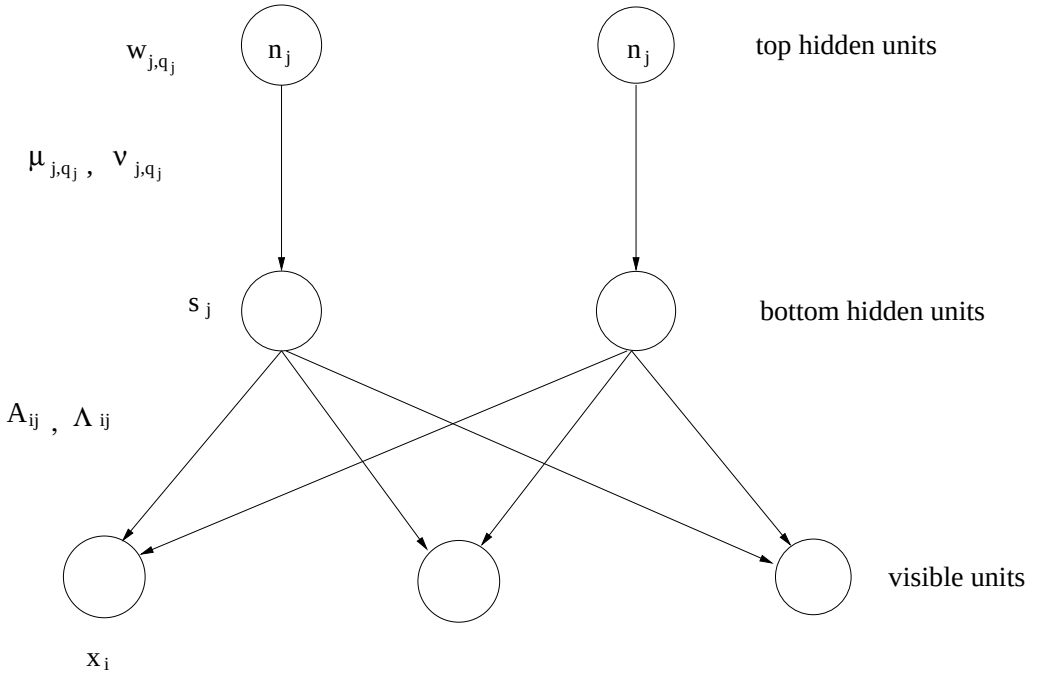


Figure 4.1: Generative model

With these definitions in place, it is trivial to show that the joint source densities can be written in the following convenient form:

$$p(\mathbf{s}|\boldsymbol{\theta}) = \sum_{\mathbf{q}} w_{\mathbf{q}} \mathcal{N}(\boldsymbol{\mu}_{\mathbf{q}}, \mathbf{V}_{\mathbf{q}}) \tag{4.8}$$

From (4.8) it is evident that, viewed in L-dimensional space, the joint source density $p(\mathbf{s})$ is itself a MOG, with mixing proportions $w_{\mathbf{q}}$, mean $\boldsymbol{\mu}_{\mathbf{q}}$, and covariance matrix $\mathbf{V}_{\mathbf{q}}$. There is however an important difference between this L-dimensional MOG model, and the standard MOG fitting scheme. This is evident from the fact that modifying the mean and variance of a single-source state q_i results in a modification of all the $n^{(L-1)}$ different $\mathbf{q}$ (for the case of all $n_i = n$) which q_i is an element in, whereas in ordinary MOG estimation the Gaussians are free to adapt independently. Our source model is a mixture of coadaptive Gaussians.

4.2.2 The Sensor Model

Having a description of the source model, we are now ready to proceed with a probabilistic description of the sensor model. Specifically we are interested in finding an analytically tractable description of $p(\mathbf{x}|W)$, since later on we will

be interested in finding the W that minimises the distance between $p(\mathbf{x}|W)$ and the observed distribution $p(\mathbf{x})^0$.

Now, referring again to figure(4.1), the basic generative model is as follows: For every source s_i, pick a particular realization of q_i each with probability w_{i,q_i}. The probability of a particular $\mathbf{q}$ is then

$$p(\mathbf{q}) = w_{\mathbf{q}} \tag{4.9}$$

Given q_i for every source, each unit will produce a sample s_i from a Gaussian density with mean μ_{i,q_i} and variance ν_{i,q_i}. It is straightforward to demonstrate that the probability of generating a particular vector $\mathbf{s}$ is

$$p(\mathbf{s}|\mathbf{q}) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{q}}, \mathbf{V}_{\mathbf{q}}) \tag{4.10}$$

Now, given a particular realization of $\mathbf{s}$, it follows directly from (4.1) and (4.2) that

$$p(\mathbf{x}|\mathbf{s}, W) = \mathcal{N}(\mathbf{A}\mathbf{s}, \boldsymbol{\Lambda}) \tag{4.11}$$

Next we note from equation (4.4) that $p(\mathbf{x}|W)$ can be written as

$$p(\mathbf{x}|W) = \int p(\mathbf{x}|\mathbf{s}, W)p(\mathbf{s}|W)\, d\mathbf{s} = \int p(\mathbf{x}|\mathbf{s}, \mathbf{A}, \boldsymbol{\Lambda})p(\mathbf{s}|\boldsymbol{\theta})\, d\mathbf{s} \tag{4.12}$$

where $p(\mathbf{x}|\mathbf{s}, W) = p(\mathbf{x}|\mathbf{s}, \mathbf{A}, \boldsymbol{\Lambda})$, since the identity of the state that created $\mathbf{s}$ has no bearing on the the probability of $\mathbf{x}$ once $\mathbf{s}$ has been created. Similarly, the parameters describing the relationship between $\mathbf{s}$ and $\mathbf{x}$ have no bearing on the the probability of generating a particular $\mathbf{s}$ to begin with. As such the IF layers form a top-down first-order Markov chain, meaning that the probability of a certain outcome in the next layer only depends on the outcome in the present. Consequently, equation (4.12) is valid, and we can write (dropping the explicit reference to $\mathbf{A}$ and $\boldsymbol{\Lambda}$)

$$\begin{aligned} p(\mathbf{x}|W) &= \int p(\mathbf{x}|\mathbf{s})p(\mathbf{s}|\boldsymbol{\theta})\, d\mathbf{s} = \int p(\mathbf{x}|\mathbf{s}) \sum_{\mathbf{q}} p(\mathbf{q})p(\mathbf{s}|\mathbf{q})\, d\mathbf{s} \\ &= \sum_{\mathbf{q}} p(\mathbf{q}) \int p(\mathbf{s}|\mathbf{q})p(\mathbf{x}|\mathbf{s})\, d\mathbf{s} = \sum_{\mathbf{q}} p(\mathbf{q})p(\mathbf{x}|\mathbf{q}) \end{aligned} \tag{4.13}$$

where the last equality follows because $\int p(\mathbf{s}|\mathbf{q})p(\mathbf{x}|\mathbf{s})\, d\mathbf{s} = p(\mathbf{x}|\mathbf{q})$. To find the conditional distribution $p(\mathbf{x}|\mathbf{q})$ one may of course evaluate the integral, which will yield an analytical solution since both quantities being integrated

over are Gaussian. A less cumbersome alternative is to first find the conditional distribution of $\mathbf{y} = \mathbf{As}$ and then $\mathbf{x} = \mathbf{y} + \boldsymbol{\nu}$ for a given $\mathbf{q}$. The characteristic function of $p(\mathbf{y}|\mathbf{q})$ is

$$\begin{aligned}\phi_{\mathbf{y}|\mathbf{q}}(\boldsymbol{\omega}) &= E_{\mathbf{y}|\mathbf{q}}\left[\exp(-j\boldsymbol{\omega}^T\mathbf{y})\right] = E_{\mathbf{s}|\mathbf{q}}\left[\exp(-j\boldsymbol{\omega}^T\mathbf{As})\right] = \int \exp(-j\boldsymbol{\omega}^T\mathbf{As})p(\mathbf{s}|\mathbf{q})\,d\mathbf{s} \\ &= \int |2\pi\mathbf{V_q}|^{-\frac{1}{2}}\exp\big(-j\boldsymbol{\omega}^T\mathbf{As} - \frac{1}{2}(\mathbf{s}-\boldsymbol{\mu}_\mathbf{q})^T\mathbf{V_q}^{-1}(\mathbf{s}-\boldsymbol{\mu}_\mathbf{q})\big)\,d\mathbf{s}\end{aligned}$$

By completing the square in the exponent, we can write

$$\begin{aligned}\phi_{\mathbf{y}|\mathbf{q}}(\boldsymbol{\omega}) &= \int |2\pi\mathbf{V_q}|^{-\frac{1}{2}}\exp\big(-j\boldsymbol{\omega}^T\mathbf{As} - \frac{1}{2}(\mathbf{s}-\boldsymbol{\mu}_\mathbf{q}+j\mathbf{V_q}\mathbf{A}^T\boldsymbol{\omega})^T\mathbf{V_q}^{-1}(\mathbf{s}-\boldsymbol{\mu}_\mathbf{q}+j\mathbf{V_q}\mathbf{A}^T\boldsymbol{\omega}) \\ &\quad +j\boldsymbol{\omega}^T(\mathbf{As}-\mathbf{A}\boldsymbol{\mu}_\mathbf{q}) - \frac{1}{2}\boldsymbol{\omega}^T\mathbf{AV_q}\mathbf{A}^T\boldsymbol{\omega}\big)d\mathbf{s} \\ &= \exp(j\boldsymbol{\omega}^T\mathbf{A}\boldsymbol{\mu}_\mathbf{q} - \frac{1}{2}\boldsymbol{\omega}^T\mathbf{AV_q}\mathbf{A}^T\boldsymbol{\omega})\int |2\pi\mathbf{V_q}|^{-\frac{1}{2}}\exp\big(-\frac{1}{2}(\mathbf{s}-\boldsymbol{\mu}_\mathbf{q}+j\mathbf{V_q}\mathbf{A}^T\boldsymbol{\omega})^T \\ &\quad \times\mathbf{V_q}^{-1}(\mathbf{s}-\boldsymbol{\mu}_\mathbf{q}+j\mathbf{V_q}\mathbf{A}^T\boldsymbol{\omega})\big)d\mathbf{s} \\ &= \exp(j\boldsymbol{\omega}^T\mathbf{A}\boldsymbol{\mu}_\mathbf{q} - \frac{1}{2}\boldsymbol{\omega}^T\mathbf{AV_q}\mathbf{A}^T\boldsymbol{\omega})\end{aligned}$$

It is a well known result (see e.g [29]) that the characteristic function of a multivariate Gaussian is given by

$$\phi(\boldsymbol{\omega}) = \exp(j\boldsymbol{\omega}^T\boldsymbol{\mu} - \frac{1}{2}\boldsymbol{\omega}^T\boldsymbol{\Sigma}\boldsymbol{\omega})$$

where $\boldsymbol{\mu}$ is its mean, and Σ its covariance matrix. Since the characteristic function of a random variable completely defines its pdf (they are paired by the Fourier transform), we conclude that $p(\mathbf{y}|\mathbf{q}) = \mathcal{N}(\mathbf{A}\boldsymbol{\mu}_\mathbf{q}, \mathbf{AV_q}\mathbf{A}^T)$. Now, adding the noise $\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda})$ which is independent of $\mathbf{y}$, we find the characteristic function for $\mathbf{x} = \mathbf{y} + \boldsymbol{\nu}$:

$$\begin{aligned}\phi_{\mathbf{x}|\mathbf{q}}(\boldsymbol{\omega}) &= \phi_{\mathbf{y}|\mathbf{q}+\boldsymbol{\nu}}(\boldsymbol{\omega}) \\ &= \phi_{\mathbf{y}|\mathbf{q}}(\boldsymbol{\omega})\phi_{\boldsymbol{\nu}}(\boldsymbol{\omega}) = \exp(-j\boldsymbol{\omega}^T\mathbf{A}\boldsymbol{\mu}_\mathbf{q} - \frac{1}{2}\boldsymbol{\omega}^T\mathbf{AV_q}\mathbf{A}^T\boldsymbol{\omega})\cdot\exp(-\frac{1}{2}\boldsymbol{\omega}^T\boldsymbol{\Lambda}\boldsymbol{\omega}) \\ &= \exp\left(-j\boldsymbol{\omega}^T\mathbf{A}\boldsymbol{\mu}_\mathbf{q} - \frac{1}{2}\boldsymbol{\omega}^T(\mathbf{AV_q}\mathbf{A}^T+\boldsymbol{\Lambda})\boldsymbol{\omega}\right)\end{aligned}$$

Here we have used the well known fact that the pdf of a sum of two random variables is the convolution of their respective pdfs. Since the characteristic function is Fourier paired to its pdf, the convolution reduces to a multiplication in the above equation, and we can safely conclude that

$$p(\mathbf{x}|\mathbf{q}) = \mathcal{N}(\mathbf{A}\boldsymbol{\mu}_\mathbf{q}, \mathbf{AV_q}\mathbf{A}^T + \boldsymbol{\Lambda}) \tag{4.14}$$

Consequently the sensor density can be written

$$p(\mathbf{x}|W) = \sum_{\mathbf{q}} p(\mathbf{x}|\mathbf{q})p(\mathbf{q}) = \sum_{\mathbf{q}} w_{\mathbf{q}} \mathcal{N}(\mathbf{A}\boldsymbol{\mu}_{\mathbf{q}}, \mathbf{A}\mathbf{V}_{\mathbf{q}}\mathbf{A}^T + \boldsymbol{\Lambda}) \tag{4.15}$$

and we can conclude that the sensor density model is a mixture of coadaptive Gaussians.

4.3 Deriving The Learning Rules

As a starting point for learning the parameters in the IF model, we need an error function that measures the distance between the model sensor density $p(\mathbf{y}|w)$ and the observed density $p^o(\mathbf{y})$. The goal will then be to adapt the parameters W iteratively to minimise this error. A natural candidate for the error function is the Kullback-Leibler (KL) distance, defined by

$$\varepsilon(W) = \int p^o(\mathbf{x}) \log \frac{p^o(\mathbf{x})}{p(\mathbf{y}|W)}\, d\mathbf{x} = -E\left[\log p(\mathbf{x}|W)\right] - H_{p^o} \tag{4.16}$$

where the expectation is over the observed $\mathbf{y}$ and H_{p^o} is the output entropy. The entropy is independent of W and will henceforth be dropped, since its presence does not affect the minimisation with respect to W. Consequently our task is that of minimising $-E\left[\log p(\mathbf{y}|W)\right]$. We recognise the expectation as the log-likelihood of the observed signals given the model parameters W. Hence, minimising the KL distance is equivalent to maximising the likelihood of the data with respect to the model.

Rather than using a gradient based technique for minimising equation (4.16), a EM approach is chosen, since the former approach can result in exceedingly slow learning. As discussed in chapter 2, the EM algorithm is well suited to finding maximumlikelihoodd estimates in probabilistic models when part of the data is missing. In our model the missing data are the sources $\mathbf{s}$ and the states $\mathbf{q}$, and W are theparameterss to be estimated. From the formulation of the EM algorithm in chapter 2 we proceed by two steps:

1. **The E-Step**: Calculate $l(W|W')$

$$\begin{aligned} l(W|W') &= E_{\mathbf{q},\mathbf{s}|\mathbf{x},W'}\left[\log p(\mathbf{q},\mathbf{s},\mathbf{x}|W)\right] \\ &= \sum_{\mathbf{q}} \int p(\mathbf{q},\mathbf{s}|\mathbf{x},W') \log p(\mathbf{q},\mathbf{s},\mathbf{x}|W)\, d\mathbf{s} \end{aligned} \tag{4.17}$$

for each observed $\mathbf{x}$. The result is then averaged over all observations.

2. **The M-Step**: Maximise $l(W|W')$. That, is find the updated estimate W'' such that

$$W'' = \arg\max_{W} \left[l(W|W') \right] \tag{4.18}$$

Now, returning to the joint density of the visible layer and the two hidden layers (see figure 4.1), it is easy to show that because of the Markov property

$$p(\mathbf{q}, \mathbf{s}, \mathbf{x}|W) = p(\mathbf{q})p(\mathbf{s}|\mathbf{q})p(\mathbf{x}|\mathbf{s}) \tag{4.19}$$

Inserted into equation (4.17) we obtain,

$$\begin{aligned} l(W|W') &= \sum_{\mathbf{q}} \int p(\mathbf{q}, \mathbf{s}|\mathbf{x}, W') \log \left[p(\mathbf{q})p(\mathbf{s}|\mathbf{q})p(\mathbf{x}|\mathbf{s}) \right] \, d\mathbf{s} \\ &= \underbrace{E_{\mathbf{q},\mathbf{s}|\mathbf{x},W'} \left\{ \log p(\mathbf{q}) \right\}}_{l_T} + \underbrace{E_{\mathbf{q},\mathbf{s}|\mathbf{x},W'} \left\{ \log p(\mathbf{s}|\mathbf{q}) \right\}}_{l_B} + \underbrace{E_{\mathbf{q},\mathbf{s}|\mathbf{x},W'} \left\{ \log p(\mathbf{x}|\mathbf{s}) \right\}}_{l_V} \end{aligned} \tag{4.20}$$

so we see that, because of the Markov property, we can split the problem of maximisation into three parts: The visible layer l_V, the bottom hidden layer l_B, and the top hidden layer l_T. Writing out the individual terms we find that

$$\begin{aligned} l_V(W', \mathbf{A}, \mathbf{\Lambda}) &= \sum_{\mathbf{q}} \int p(\mathbf{q}, \mathbf{s}|\mathbf{x}, W') \log p(\mathbf{x}|\mathbf{s}) \, d\mathbf{s} \\ &= \int p(\mathbf{s}|\mathbf{x}, W') \log p(\mathbf{x}|\mathbf{s}) \, d\mathbf{s} \\ l_B(W', \{\mu_{i,q_i}, \nu_{i,q_i}\}) &= \sum_{\mathbf{q}} \int p(\mathbf{q}, \mathbf{s}|\mathbf{x}, W') \log p(\mathbf{s}|\mathbf{q}) \, d\mathbf{s} \\ &= \sum_{\mathbf{q}} \int p(\mathbf{s}|\mathbf{q}, \mathbf{x}, W')p(\mathbf{q}|\mathbf{x}, W') \log p(\mathbf{s}|\mathbf{q}) \, d\mathbf{s} \\ &= \sum_{i=1}^{L} \sum_{q_i=1}^{n_i} p(q_i|\mathbf{x}, W') \int p(s_i|q_i, \mathbf{x}, W') \log p(s_i|q_i) \, ds_i \\ l_T(W', \{w_{i,q_i}\}) &= \sum_{\mathbf{q}} \int p(\mathbf{q}, \mathbf{s}|\mathbf{x}, W') \log p(\mathbf{q}) \, d\mathbf{s} \\ &= \sum_{i=1}^{L} \sum_{q_i=1}^{n_i} p(q_i|\mathbf{x}, W') \log p(q_i) \end{aligned} \tag{4.21}$$

where it is made explicit which of the parameters in W the different layers depend on. Inserting the densities from equations (4.9), (4.10) and (4.11) into the expressions we find:

$$
\begin{aligned}
&l_V(W', \mathbf{A}, \mathbf{\Lambda}) \\
&= \int p(\mathbf{s}|\mathbf{x}, W') \log \left\{ |2\pi\mathbf{\Lambda}|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(\mathbf{x} - \mathbf{A}\mathbf{s})^T \mathbf{\Lambda}^{-1} (\mathbf{x} - \mathbf{A}\mathbf{s}) \right] \right\} d\mathbf{s} \\
&= -\frac{1}{2} \log |2\pi\mathbf{\Lambda}| - \frac{1}{2} \int p(\mathbf{s}|\mathbf{x}, W') \left[(\mathbf{x} - \mathbf{A}\mathbf{s})^T \mathbf{\Lambda}^{-1} (\mathbf{x} - \mathbf{A}\mathbf{s}) \right] d\mathbf{s} \\
&= -\frac{1}{2} \log |2\pi\mathbf{\Lambda}| - \frac{1}{2} \int p(\mathbf{s}|\mathbf{x}, W') \mathrm{Tr}\left[\mathbf{x}^T \mathbf{\Lambda}^{-1} \mathbf{x} - 2(\mathbf{A}\mathbf{s})^T \mathbf{\Lambda}^{-1} \mathbf{x} + \mathbf{s}^T \mathbf{A}^T \mathbf{\Lambda}^{-1} \mathbf{A}\mathbf{s} \right] d\mathbf{s} \\
&= -\frac{1}{2} \log |2\pi\mathbf{\Lambda}| - \frac{1}{2} \left\{ \mathrm{Tr}\left[\mathbf{\Lambda}^{-1} \mathbf{x}\mathbf{x}^T \right] - \mathrm{Tr}\left[2\mathbf{\Lambda}^{-1} \mathbf{x} \langle \mathbf{s}^T | \mathbf{x}, W' \rangle \mathbf{A}^T \right] \right. \\
&\quad \left. + \mathrm{Tr}\left[\mathbf{\Lambda}^{-1} \mathbf{A} \langle \mathbf{s}\mathbf{s}^T | \mathbf{x}, W' \rangle \mathbf{A}^T \right] \right\} \\
&= -\frac{1}{2} \log |2\pi\mathbf{\Lambda}| - \frac{1}{2} \mathrm{Tr}\left[\mathbf{\Lambda}^{-1} \left(\mathbf{x}\mathbf{x}^T - 2\mathbf{x} \langle \mathbf{s}^T | \mathbf{x}, W' \rangle \mathbf{A}^T + \mathbf{A} \langle \mathbf{s}\mathbf{s}^T | \mathbf{x}, W' \rangle \mathbf{A}^T \right) \right]
\end{aligned}
\tag{4.22}
$$

$$
\begin{aligned}
&l_B(W', \{\mu_{i,q_i}, \nu_{i,q_i}\}) \\
&= \sum_{i=1}^{L} \sum_{q_i=1}^{n_i} p(q_i|\mathbf{x}, W') \\
&\quad \times \int p(s_i|q_i, \mathbf{x}, W') \log \left[(2\pi\nu_{i,q_i})^{-\frac{1}{2}} \exp\left(-\frac{1}{2\nu_{i,q_i}} (s_i - \mu_{i,q_i})^2 \right) \right] ds_i \\
&= \sum_{i=1}^{L} \sum_{q_i=1}^{n_i} p(q_i|\mathbf{x}, W') \\
&\quad \times \left[-\frac{1}{2} \log(2\pi\nu_{i,q_i}) - \frac{1}{2\nu_{i,q_i}} \left(\langle s_i^2 | q_i, \mathbf{x}, W' \rangle - 2\langle s_i | q_i, \mathbf{x}, W' \rangle \mu_{i,q_i} + \mu_{i,q_i}^2 \right) \right]
\end{aligned}
\tag{4.23}
$$

$$
l_T(W', \{w_{i,q_i}\}) = \sum_{i=1}^{L} \sum_{q_i=1}^{n_i} p(q_i|\mathbf{x}, W') \log(w_{i,q_i}) \tag{4.24}
$$

The expectations in equations (4.22) and (4.23) deserve some special attention. We need to calculate the following

$$
\langle m(\mathbf{s}) | \mathbf{x}, W' \rangle = \int m(\mathbf{s}) p(\mathbf{s}|\mathbf{x}, W') \, d\mathbf{s} \tag{4.25}
$$

$$
\langle m(s_i) | q_i, \mathbf{x}, W' \rangle = \int m(s_i) p(s_i|q_i, \mathbf{x}, W') \, ds_i \tag{4.26}
$$

where $m(\mathbf{s}) = \mathbf{s}, \mathbf{s}\mathbf{s}^T$ and $m(s_i) = s_i, s_i^2$. I will in the following, for notational convenience, drop the explicit reference to the parameters W'. It should however be kept in mind that all the averages in equations (4.27) to (4.42) are based on the parameters arrived at in the previous iteration W'. We could at this stage of course try to find an expression for $p(\mathbf{s}|\mathbf{x})$, but this would be extremely cumbersome involving the quotient of two MOGs, since $p(\mathbf{s}|\mathbf{x}) = p(\mathbf{s})p(\mathbf{x}|\mathbf{s})/p(\mathbf{x})$ with the relevant densities given in equations (4.8), (4.11) and (4.15). A more productive course, is to first calculate the expectations for a given $\mathbf{q}$, and then sum over the posteriors $p(\mathbf{q}|\mathbf{x})$. That is, find

$$\langle m(\mathbf{s})|\mathbf{q},\mathbf{x}\rangle = \int m(\mathbf{s})p(\mathbf{s}|\mathbf{q},\mathbf{x})\,d\mathbf{s} \tag{4.27}$$

and sum over the posteriors $p(\mathbf{q}|\mathbf{x})$ for all states $\mathbf{q}$, since

$$\begin{aligned}
&\sum_{\mathbf{q}} p(\mathbf{q}|\mathbf{x})\langle m(\mathbf{s})|\mathbf{q},\mathbf{x}\rangle \\
&= \sum_{\mathbf{q}} p(\mathbf{q}|\mathbf{x}) \int m(\mathbf{s})p(\mathbf{s}|\mathbf{q},\mathbf{x})\,d\mathbf{s} \\
&= \int m(\mathbf{s}) \sum_{\mathbf{q}} p(\mathbf{q}|\mathbf{x})p(\mathbf{s}|\mathbf{q},\mathbf{x})\,d\mathbf{s} = \int m(\mathbf{s})p(\mathbf{s}|\mathbf{x})\,d\mathbf{s} \\
&= \langle m(\mathbf{s})|\mathbf{x}\rangle
\end{aligned}$$

which is the desired result. Thus we need to find the density for $\mathbf{s}$ with $\mathbf{x}$ and $\mathbf{q}$ given. Now, since

$$p(\mathbf{s}|\mathbf{q},\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{s})p(\mathbf{s}|\mathbf{q})}{p(\mathbf{x}|\mathbf{q})} = \frac{\mathcal{N}(\mathbf{A}\mathbf{s},\mathbf{\Lambda})\mathcal{N}(\boldsymbol{\mu}_{\mathbf{q}},\mathbf{V}_{\mathbf{q}})}{\mathcal{N}(\mathbf{A}\boldsymbol{\mu}_{\mathbf{q}},\mathbf{A}\mathbf{V}_{\mathbf{q}}\mathbf{A}^T+\mathbf{\Lambda})} \tag{4.28}$$

we can write out the terms in the exponential in equation (4.28) involving $\mathbf{s}$. Gathering terms, these are found to be:

$$-\frac{1}{2}\mathbf{s}^T(\mathbf{A}^T\mathbf{\Lambda}^{-1}\mathbf{A}+\mathbf{V}_{\mathbf{q}}^{-1})\mathbf{s} + \mathbf{s}^T(\mathbf{A}^T\mathbf{\Lambda}^{-1}\mathbf{x}+\mathbf{V}_{\mathbf{q}}^{-1}\boldsymbol{\mu}_{\mathbf{q}})$$

Recognising this as a Gaussian in canonical form, we can immediately conclude that the covariance matrix for $\mathbf{s}|\mathbf{q},\mathbf{x}$ is

$$\Sigma_{\mathbf{q}} = (\mathbf{A}^T\mathbf{\Lambda}^{-1}\mathbf{A}+\mathbf{V}_{\mathbf{q}}^{-1})^{-1} \tag{4.29}$$

Denoting its mean by $\boldsymbol{\rho}_{\mathbf{q}}(\mathbf{x})$ we also find that

$$\begin{aligned}
\mathbf{s}^T\Sigma_{\mathbf{q}}^{-1}\boldsymbol{\rho}_{\mathbf{q}}(\mathbf{x}) &= \mathbf{s}^T(\mathbf{A}^T\mathbf{\Lambda}^{-1}\mathbf{x}+\mathbf{V}_{\mathbf{q}}^{-1}\boldsymbol{\mu}_{\mathbf{q}}) \\
&\Downarrow \\
\boldsymbol{\rho}_{\mathbf{q}}(\mathbf{x}) &= \Sigma_{\mathbf{q}}(\mathbf{A}^T\mathbf{\Lambda}^{-1}\mathbf{x}+\mathbf{V}_{\mathbf{q}}^{-1}\boldsymbol{\mu}_{\mathbf{q}})
\end{aligned} \tag{4.30}$$

Hence,

$$p(\mathbf{s}|\mathbf{q},\mathbf{x}) = \mathcal{N}(\boldsymbol{\rho}_{\mathbf{q}}(\mathbf{x}), \Sigma_{\mathbf{q}}) \tag{4.31}$$

and it follows that

$$\langle \mathbf{s}|\mathbf{q},\mathbf{x}\rangle = \boldsymbol{\rho}_{\mathbf{q}}(\mathbf{x}), \qquad \langle \mathbf{s}\mathbf{s}^T|\mathbf{q},\mathbf{x}\rangle = \Sigma_{\mathbf{q}} + \boldsymbol{\rho}_{\mathbf{q}}(\mathbf{x})\boldsymbol{\rho}_{\mathbf{q}}(\mathbf{x})^T \tag{4.32}$$

If you are of a particularly masochistic persuasion, you can of course arrive at the same result through meticulous manipulation of equation (4.28). We now need to find the posteriors $p(\mathbf{q}|\mathbf{x})$. These are, however, readily attained through equations (4.9) and (4.14), since

$$p(\mathbf{q}|\mathbf{x}) = \frac{p(\mathbf{q})p(\mathbf{x}|\mathbf{q})}{\sum_{\mathbf{q}'} p(\mathbf{q}')p(\mathbf{x}|\mathbf{q}')} \tag{4.33}$$

and we have all we need to calculate

$$\langle m(\mathbf{s})|\mathbf{x}\rangle = \sum_{\mathbf{q}} p(\mathbf{q}|\mathbf{x})\langle m(\mathbf{s})|\mathbf{q},\mathbf{x}\rangle \tag{4.34}$$

Most of the work is now done, also for the calculation of the individual source averages in equation (4.26). We note from equation (4.32) that $\langle s_i|\mathbf{q},\mathbf{x}\rangle$ are simply the elements of $\langle \mathbf{s}|\mathbf{q},\mathbf{x}\rangle$ and $\langle s_i^2|\mathbf{q},\mathbf{x}\rangle$ the diagonal elements of $\langle \mathbf{s}\mathbf{s}^T|\mathbf{q},\mathbf{x}\rangle$. Returning to equation (4.23), we see that we actually need to calculate $p(q_i|\mathbf{x})\langle m(s_i)|q_i,\mathbf{x}\rangle$. A closer look reveals that

$$p(q_i|\mathbf{x})\langle m(s_i)|q_i,\mathbf{x}\rangle = \sum_{\substack{\mathbf{q} \\ \text{containing} \\ q_i}} p(\mathbf{q}|\mathbf{x})\langle m(s_i)|\mathbf{q},\mathbf{x}\rangle \tag{4.35}$$

where the summation is to be understood as follows: Each q_i can take n_i different values. For one value of q_i, there will be $\prod_{j\neq i}^{L} n_j$ different $\mathbf{q}$, where q_i is an element. These are the $\mathbf{q}$ that are to be summed over. The same applies for finding the individual state posteriors, as seen by:

$$p(q_i|\mathbf{x}) = \sum_{\substack{\mathbf{q} \\ \text{containing} \\ q_i}} p(\mathbf{q}|\mathbf{x}) \tag{4.36}$$

This completes the expectation step of the EM algorithm, since we are now able to calculate $l(W|W')$. What remains is the maximisation of $l(W|W')$ with respect to W. This is done by computing its gradient $\partial l/\partial W$ layer for layer. Through standard rules of matrix calculus it is easily verified that:

$$\frac{\partial l_V}{\partial \mathbf{A}} = -\boldsymbol{\Lambda}^{-1}\mathbf{x}\langle \mathbf{s}^T|\mathbf{x}\rangle + \boldsymbol{\Lambda}^{-1}\mathbf{A}\langle \mathbf{s}\mathbf{s}^T|\mathbf{x}\rangle$$

$$\frac{\partial l_V}{\partial \mathbf{\Lambda}} = \frac{1}{2}\mathbf{\Lambda}^{-1} - \frac{1}{2}\mathbf{\Lambda}^{-1}\left(\mathbf{s}\mathbf{s}^T - 2\mathbf{x}\langle \mathbf{s}^T|\mathbf{x}\rangle \mathbf{A}^T + \mathbf{A}\langle \mathbf{s}\mathbf{s}^T|\mathbf{x}\rangle \mathbf{A}^T\right)\mathbf{\Lambda}^{-1} \tag{4.37}$$

For the bottom hidden layer we find

$$\begin{aligned}\frac{\partial l_B}{\partial \mu_{i,q_i}} &= \frac{1}{\nu_{i,q_i}} p(q_i|\mathbf{x})(\langle s_i|q_i,\mathbf{x}\rangle - \mu_{i,q_i}) \\ \frac{\partial l_B}{\partial \nu_{i,q_i}} &= \frac{1}{\nu_{i,q_i}^2} p(q_i|\mathbf{x})\left(\langle s_i^2|q_i,\mathbf{x}\rangle - 2\langle s_i|q_i,\mathbf{x}\rangle \mu_{i,q_i} + \mu_{i,q_i}^2 - \nu_{i,q_i}\right)\end{aligned} \tag{4.38}$$

In the maximisation of l_T with respect to the mixing proportions w_{i,q_i} we need to enforce the constraint that $\sum_{q_i=1}^{n_i} w_{i,q_i} = 1$, and ensure that all the w_{i,q_i} are nonnegative, since $w_{i,q_i} = p(q_i)$ are probabilities. This could be done by the method of Lagrange multipliers, but for reasons not fully understood, it is more practical to enforce the constraints automatically by working with new parameters $\overline{w}_{i,q_i}$, related to w_{i,q_i} through

$$w_{i,q_i} = \frac{e^{\overline{w}_{i,q_i}}}{\sum_{q_i'} e^{\overline{w}_{i,q_i}}} \tag{4.39}$$

Taking the gradient with respect to the new parameters we find

$$\frac{\partial l_T}{\partial \overline{w}_{i,q_i}} = p(q_i|\mathbf{x}) - w_{i,q_i} \tag{4.40}$$

What remains is to set these equations equal to zero, and to solve them with regard to the parameters in W. Finally, one averages over all the $\mathbf{x}$, and arrives at the following rules for updating W:

$$\begin{aligned}\mathbf{A}^{new} &= E\left[\mathbf{x}\langle \mathbf{s}^T|\mathbf{x}\rangle\right]\left(E\left[\langle \mathbf{s}\mathbf{s}^T|\mathbf{x}\rangle\right]\right)^{-1} \\ \mathbf{\Lambda}^{new} &= E\left[\mathbf{x}\mathbf{x}^T\right] - E\left[\mathbf{x}\langle \mathbf{s}^T|\mathbf{x}\rangle\right]\mathbf{A}_{new}^T\end{aligned} \tag{4.41}$$

and, for the MOG parameters θ

$$\begin{aligned}\mu_{i,q_i}^{new} &= \frac{E\left[p(q_i|\mathbf{x})\langle s_i|q_i,\mathbf{x}\rangle\right]}{E\left[p(q_i|\mathbf{x})\right]} \\ \nu_{i,q_i}^{new} &= \frac{E\left[p(q_i|\mathbf{x})\langle s_i^2|q_i,\mathbf{x}\rangle\right]}{E\left[p(q_i|\mathbf{x})\right]} - \left(\mu_{i,q_i}^{new}\right)^2 \\ w_{i,q_i}^{new} &= E\left[p(q_i|\mathbf{x})\right]\end{aligned} \tag{4.42}$$

where the expectation operator is over all the observations, i.e. for any function $\mathbf{F}$ of $\mathbf{x}$, $E\left[\mathbf{F}(\mathbf{x})\right] = \frac{1}{T}\sum_{t=1}^{T}\mathbf{F}(\mathbf{x}^{(t)})$.

While we could be content with the learning rules as they stand, it is practical to enforce a scaling on the parameters. This is because in the Blind Signal Separation (BSS) problem, the sources are defined only within an order permutation and scaling as discussed in chapter 3. In order to prevent the parameters from acquiring arbitrarily large values, we perform a scaling to ensure that the variance of each source is kept equal to unity, and compensate the mixing matrix accordingly. This scaling takes the following form:

$$\sigma_j^2 = \sum_{q_j}^{n_j} w_{j,q_j}(\nu_{j,q_j} + \mu_{j,q_j}^2) - \left(\sum_{q_j}^{n_j} w_{j,q_j}\mu_{j,q_j} \right)^2$$

$$\mu_{j,q_j} \rightarrow \frac{\mu_{j,q_j}}{\sigma_j}, \qquad \nu_{j,q_j} \rightarrow \frac{\nu_{j,q_j}}{\sigma_j^2}, \qquad A_{ij} \rightarrow A_{ij}\sigma_j \tag{4.43}$$

Chapter 5

Scale Mixture Source Separation

5.1 Introduction

The goal of this chapter is to develop a source separation algorithm where the sources are modelled as scale mixtures of Gaussians [2]. Several different approaches to the problem are studied, some making assumptions on the prior distributions, and some not. The approach in itself is not entirely novel. In particular J. A. Palmer has concerned himself with the topic [26], where many aspects of the problem at hand are discussed. Scale mixtures of Gaussians are well suited in modeling typically peaky densities, which characterise many acoustic recordings, speech being perhaps the most common variant. It is hoped that the algorithm will prove general enough to separate a wide variety of signals.

5.2 The Model

Again we examine the case of an instantaneous mixture of L source signals, creating L' sensor signals consisting of the mixture plus noise.

$$\mathbf{x} = \mathbf{A}\mathbf{s} + \boldsymbol{\nu}, \qquad p(\boldsymbol{\nu}) = \mathscr{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\nu}}) \tag{5.1}$$

Each of the sources is modelled as a 1-dimensional scale mixture of Gaussian

$$s_i = \sqrt{z_i} y_i, \qquad p(y_i) = \mathscr{N}(0, 1) \tag{5.2}$$

The sources as well as the mixtures are assumed zero mean, without losing generality, for the reasons explained in chapter 3.2. The generative model is

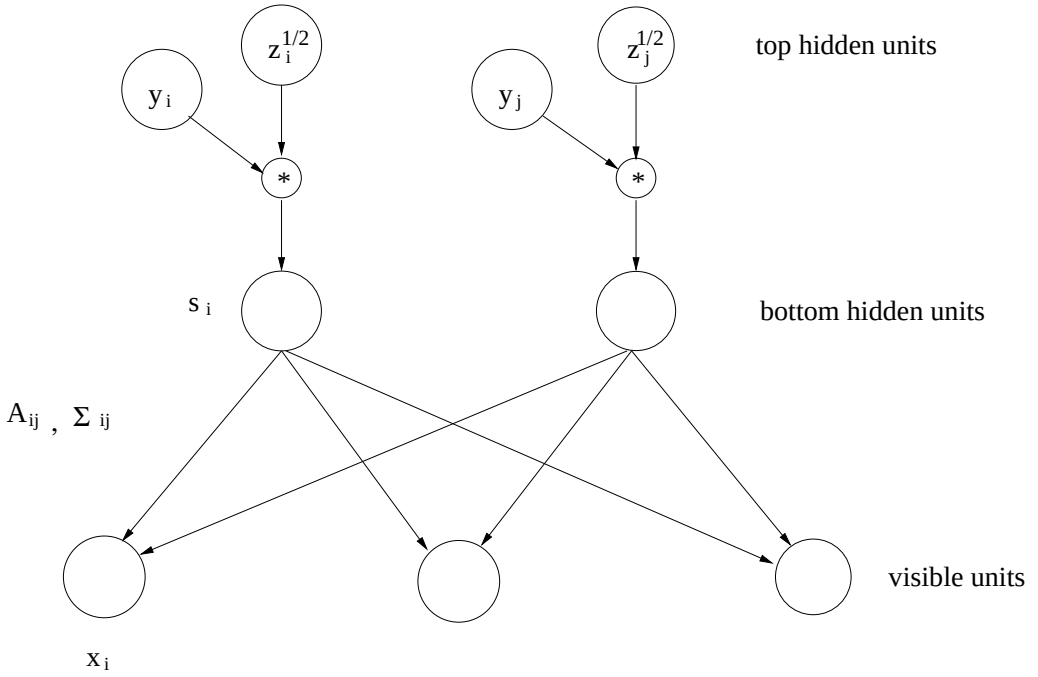


Figure 5.1: Generative model. To generate a source s_i, draw z_i from a distribution $p(z)$ (e.g. from equations (5.8) or (5.10)) and multiply with y_i which is drawn from a Normal distribution of zero mean and unit variance. The observations x_i then are generated as a mixture of all s_i and with noise added.

illustrated in figure (5.1). The random variables z_i will be termed the *hyper prior*. Different distributions of the hyper prior will give rise to different algorithms, since their distributions define the distributions of the sources s_i. In this chapter, two distribution families will be considered, and a source separation algorithm will be developed in detail for one of them.

Continuing with the specification of the model, the assumption of independence means that we must have

$$p(\mathbf{s}) = \prod_{i=1}^{L} p(s_i) \tag{5.3}$$

Now let $\mathbf{z} = [z_1, \ldots, z_L]^T$. From the independence of the s_i and the fact that when z_i is given, each of the s_i is distributed as $\mathcal{N}(0, z_i)$, we may write

$$p(\mathbf{s}|\mathbf{z}) = \prod_{i=1}^{L} p(s_i|z_i)$$

$$
\begin{aligned}
&= \prod_{i=1}^{L} (2\pi z_i)^{-\frac{1}{2}} \exp\left(-\frac{1}{2z_i} s_i^2\right) \\
&= \left(\prod_{i=1}^{L} (2\pi z_i)^{-\frac{1}{2}}\right) \exp\left(-\frac{1}{2}\sum_{i=1}^{L} \frac{1}{z_i} s_i^2\right) \\
&= |2\pi\mathbf{\Lambda}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\mathbf{s}^T\mathbf{\Lambda}^{-1}\mathbf{s}\right) \\
&= \mathcal{N}(\mathbf{0}, \mathbf{\Lambda})
\end{aligned} \tag{5.4}
$$

where

$$
\mathbf{\Lambda} = \begin{bmatrix} z_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & z_L \end{bmatrix} \tag{5.5}
$$

The overall generative setting has many similarities with the MOG model dealt with in chapter 4. Importantly, the Markov property still holds, and we may write the joint density of the visible and the two hidden layers as

$$
p(\mathbf{z}, \mathbf{s}, \mathbf{x}|W) = p(\mathbf{z})p(\mathbf{s}|\mathbf{z})p(\mathbf{x}|\mathbf{s}) \tag{5.6}
$$

The form of $p(\mathbf{s}|\mathbf{z})$ was found in equation (5.4) and, by the same reasoning as in chapter 4.2.2, $p(\mathbf{x}|\mathbf{s})$ is given by

$$
p(\mathbf{x}|\mathbf{s}) = \mathcal{N}(\mathbf{As}, \mathbf{\Sigma_\nu}) \tag{5.7}
$$

The density $p(\mathbf{z})$ is simply the product of the individual densities $p(z_i)$ since they are independent in our model. In this thesis we look at two different distributions for the individual i.i.d. z_i:

1. Γ distributed z_i

$$
p(z_i) = \frac{\lambda^{\alpha+1} z_i^{\alpha}}{\Gamma(\alpha+1)} \exp(-\lambda z_i) \qquad z > 0, \lambda > 0, \alpha > -1 \tag{5.8}
$$

   This gives us K distributed s_i:

$$
p(s_i) = \frac{2}{\sqrt{2\pi}} \frac{\lambda^{\alpha+1}}{\Gamma(\alpha+1)} \left(\sqrt{\frac{s_i^2}{2\lambda}}\right)^{\alpha+\frac{1}{2}} K_{\alpha+\frac{1}{2}}\left(\sqrt{2\lambda s_i^2}\right) \tag{5.9}
$$

2. Inverse Gaussian (IG) distributed z_i

$$
p(z_i) = \frac{\delta}{\sqrt{2\pi}} e^{\delta\gamma} z^{-\frac{3}{2}} \exp\left(-\frac{1}{2}\left(\frac{\delta^2}{z_i} + \gamma^2 z\right)\right) \qquad z > 0, \delta > 0, \gamma > 0 \tag{5.10}
$$

This gives us normal inverse Gaussian (NIG) distributed s_i:

$$p(s_i) = \frac{\delta\gamma e^{\delta\gamma}}{\pi\sqrt{\delta + s_i^2}} K_1\left(\gamma\sqrt{\delta^2 + s_i^2}\right) \tag{5.11}$$

λ and α in the Γ distribution are known respectively as the width and shape parameter from the way they affect the distribution. For the IG distribution γ and δ serve the same purpose. $\Gamma(\cdot)$ denotes the standard Gamma function[1]. $K_\nu(z)$ denotes the modified Bessel function of the second kind[2].

5.3 Deriving The Learning Rules

5.3.1 No Prior Assumptions

Contrary to the situation in chapter 4, where modelling the sources as MOGs allowed for the analytic calculation of all the quantities involved, we will now be forced to take some shortcuts in the derivation. As a starting point, we derive an algorithm without making any assumptions on the prior distribution of the z_i. As such, they can, in this context, be viewed simply as variational parameters, under which we seek to optimise the likelihood of the observations. The basic EM algorithm for achieving this is:

1. **The E-Step**: Calculate $l(W|W')$

$$\begin{aligned} l(W|W') &= E_{\mathbf{s}|\mathbf{x},W'}\left[\log p(\mathbf{s},\mathbf{x}|W)\right] \\ &= \int p(\mathbf{s}|\mathbf{x},W')\log p(\mathbf{s},\mathbf{x}|W)\, d\mathbf{s} \end{aligned} \tag{5.12}$$

   for each observed $\mathbf{x}$. The result is then averaged over all the observed $\mathbf{x}$. Here $W = \left\{\mathbf{A}, \mathbf{\Sigma_\nu}, \{\mathbf{z}_n\}_{n=1}^N\right\}$. N denotes the total number of samples.

2. **The M-Step**: Maximise $l(W|W')$. That is find the updated estimate W'' such that

$$W'' = \arg\max_W \left[l(W|W')\right] \tag{5.13}$$

   W' denotes the values of W from the previous iteration and W'', naturally, the updated estimate.

[1] The Gamma function: $\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}\, dt$

[2] $K_\nu(z)$ are solutions to Bessel's differential equation: $z^2\frac{d^2y}{dz^2} + z\frac{dy}{dz} - (z^2 + \nu^2) = 0$

Many of the results from the chapter on IFA still apply, among them the Markov property of the layers of the generative model. This means that, analogous to the IFA case, we may split the maximisation problem into parts, and we can write

$$\begin{aligned} l_V(W|W') &= \int p(\mathbf{s}|\mathbf{x}, W') \log p(\mathbf{x}|\mathbf{s})\, d\mathbf{s} \\ l_B(W|W') &= \int p(\mathbf{s}|\mathbf{x}, W') \log p(\mathbf{s}|W)\, d\mathbf{s} \end{aligned} \tag{5.14}$$

For the visible layer we apply directly the results from chapter 4.3, so

$$\begin{aligned} l_V(W|W') = &-\frac{1}{2}\log|2\pi\boldsymbol{\Sigma}_{\boldsymbol{\nu}}| \\ &-\frac{1}{2}\mathrm{Tr}\left[\boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-1}\left(\mathbf{x}\mathbf{x}^T - 2\mathbf{x}\langle\mathbf{s}^T|\mathbf{x}, W'\rangle\mathbf{A}^T + \mathbf{A}\langle\mathbf{s}\mathbf{s}^T|\mathbf{x}, W'\rangle\mathbf{A}^T\right)\right] \end{aligned} \tag{5.15}$$

where,

$$\langle m(\mathbf{s})|\mathbf{x}, W'\rangle = \int m(\mathbf{s}) p(\mathbf{s}|\mathbf{x}, W')\, d\mathbf{s} \tag{5.16}$$

with $m(\mathbf{s}) = \mathbf{s}, \mathbf{s}\mathbf{s}^T$. This of course yields the exact same update rules as in chapter 4.3 equation (4.41) , though the estimation of the required means will be slightly different. Accordingly:

$$\mathbf{A}^{new} = E\left[\mathbf{x}\langle\mathbf{s}^T|\mathbf{x}\rangle\right]\left(E\left[\langle\mathbf{s}\mathbf{s}^T|\mathbf{x}\rangle\right]\right)^{-1} \tag{5.17}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{new} = E\left[\mathbf{x}\mathbf{x}^T\right] - E\left[\mathbf{x}\langle\mathbf{s}^T|\mathbf{x}\rangle\right]\mathbf{A}_{new}^T \tag{5.18}$$

For the bottom hidden layer we find

$$\begin{aligned} l_B(W|W') &= \int p(\mathbf{s}|\mathbf{x}, W') \log\left\{|2\pi\boldsymbol{\Lambda}|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}\mathbf{s}^T\boldsymbol{\Lambda}^{-1}\mathbf{s}\right]\right\} d\mathbf{s} \\ &= -\frac{1}{2}\log|2\pi\boldsymbol{\Lambda}| - \frac{1}{2}\int p(\mathbf{s}|\mathbf{x}, W')\mathrm{Tr}[\boldsymbol{\Lambda}^{-1}\mathbf{s}\mathbf{s}^T]\, d\mathbf{s} \\ \frac{\partial l(W|W')}{\partial\boldsymbol{\Lambda}} &= -\frac{1}{2}\boldsymbol{\Lambda}^{-1} + \frac{1}{2}\int p(\mathbf{s}|\mathbf{x}, W')\boldsymbol{\Lambda}^{-1}\mathbf{s}\mathbf{s}^T\boldsymbol{\Lambda}^{-1}\, d\mathbf{s} \\ &= 0 \\ &\Downarrow \\ \boldsymbol{\Lambda} &= \int p(\mathbf{s}|\mathbf{x}, W')\mathbf{s}\mathbf{s}^T\, d\mathbf{s} \end{aligned} \tag{5.19}$$

or equivalently, in terms of $z_i(n)$ belonging to source i at time n

$$z_i(n) = E[s_i^2|\mathbf{x}_n, W'] \tag{5.20}$$

Now, following the same procedure as in chapter 4.3 it can be shown that

$$p(\mathbf{s}|\mathbf{x}, W') = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{s}|\mathbf{x},W'}, \boldsymbol{\Sigma}_{\mathbf{s}|\mathbf{x},W'}) \tag{5.21}$$

where

$$\begin{aligned} \boldsymbol{\Sigma}_{\mathbf{s}|\mathbf{x},W'} &= (\mathbf{A}^T\boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-1}\mathbf{A} + \boldsymbol{\Lambda}^{-1})^{-1} \\ &= \boldsymbol{\Lambda} - \boldsymbol{\Lambda}\mathbf{A}^T(\mathbf{A}\boldsymbol{\Lambda}\mathbf{A}^T + \boldsymbol{\Sigma}_{\boldsymbol{\nu}})^{-1}\mathbf{A}\boldsymbol{\Lambda} \qquad (5.22) \\ \boldsymbol{\mu}_{\mathbf{s}|\mathbf{x},W'} &= \boldsymbol{\Sigma}_{\mathbf{s}|\mathbf{x},W'}\mathbf{A}^T\boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-1}\mathbf{x} \\ &= \boldsymbol{\Lambda}\mathbf{A}^T(\mathbf{A}\boldsymbol{\Lambda}\mathbf{A}^T + \boldsymbol{\Sigma}_{\boldsymbol{\nu}})^{-1}\mathbf{x} \qquad (5.23) \end{aligned}$$

which completes the specification of the algorithm. The algorithm is summed up in table 5.1

5.3.2 Modifying the learning rules

As stated earlier, we seek a method in which to incorporate our assumptions on the distribution of the hyper prior. The rigorous approach for achieving this would be to employ the EM algorithm with the hyper prior included in the log-likelihood, i.e.

$$l(W|W') = E_{\mathbf{z},\mathbf{s}|\mathbf{x},W'}\left[\log p(\mathbf{z}, \mathbf{s}, \mathbf{x}|W)\right]$$

This, however, leads to hideously complicated integrals which no right thinking individual should need be subjected to. Instead, we shall use a result from [5], which states that the common *a posteriori* pdf $f_{Z|S}(z|s)$, of the mixture model in equation (5.2) is a Generalised Inverse Gaussian (GIG) distribution

$$\text{GIG:} \qquad f_Z(z|\theta, \delta, \gamma) = \left(\frac{\gamma}{\delta}\right)^{\theta} \frac{1}{2K_\theta(\delta\gamma)} z^{\theta-1} \exp\left(-\frac{1}{2}(\frac{\delta^2}{z} + \gamma^2 z)\right) \tag{5.24}$$

and its k^{th}-order moments are

$$E[z_i^k|s_i] = \left(\frac{\delta}{\gamma}\right)^k \frac{K_{\theta+k}(\delta\gamma)}{K_\theta(\delta\gamma)} \tag{5.25}$$

Table 5.1: Algorithm for scale mixture source separation with no assumptions made on priors

1. Center the observations

2. Initialise $\mathbf{z}_n$ for all n, as well as $\mathbf{A}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\nu}}$.

3. For all n, find $\langle \mathbf{s}|\mathbf{x}_n \rangle = \boldsymbol{\mu}_{\mathbf{s}|\mathbf{x}_n}$ by means of equation (5.23), using the current estimates of $\boldsymbol{\Lambda}_n = \text{diag}(\mathbf{z}_n)$, $\mathbf{A}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\nu}}$

4. For all n, find estimate of $\langle \mathbf{s}\mathbf{s}^T|\mathbf{x}_n \rangle$ by means of equation (5.22) and $\boldsymbol{\mu}_{\mathbf{s}|\mathbf{x}_n}$ from the previous step:

$$\langle \mathbf{s}\mathbf{s}^T|\mathbf{x}_n \rangle = \boldsymbol{\Sigma}_{\mathbf{s}|\mathbf{x}_n} + \boldsymbol{\mu}_{\mathbf{s}|\mathbf{x}_n}\boldsymbol{\mu}_{\mathbf{s}|\mathbf{x}_n}^T$$

   From equation (5.20), update $\mathbf{z}_n$ as

$$\mathbf{z}_n = \text{diag}\left(\langle \mathbf{s}\mathbf{s}^T|\mathbf{x}_n \rangle\right)$$

   where the diag$(\cdot)$ operator picks the diagonal terms of the matrix argument.

5. Update $\mathbf{A}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\nu}}$ using equations (5.17) and (5.18). The expectations are calculated as

$$E\left[\mathbf{x}\langle \mathbf{s}^T|\mathbf{x} \rangle\right] = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n \langle \mathbf{s}^T|\mathbf{x}_n \rangle$$

$$E\left[\langle \mathbf{s}\mathbf{s}^T|\mathbf{x} \rangle\right] = \frac{1}{N}\sum_{n=1}^{N} \langle \mathbf{s}\mathbf{s}^T|\mathbf{x}_n \rangle$$

6. If not converged, return to step 3.

We will consider spesifically the case of a Gamma distributed hyper prior, in which case the a posteriori pdf is a GIG$\left\{\alpha+\frac{1}{2},\sqrt{s_i^2},\sqrt{2\lambda}\right\}$, so its moments are given by

$$E[z_i^k|s_i]=\left(\sqrt{\frac{s_i^2}{2\lambda}}\right)^k\frac{K_{\alpha+\frac{1}{2}+k}(\sqrt{2\lambda s_i^2})}{K_{\alpha+\frac{1}{2}}(\sqrt{2\lambda s_i^2})} \tag{5.26}$$

The results for an IG distributed hyper prior are completely analogous, with the posterior pdf given as a GIG$\left\{-1,\sqrt{\delta^2+s_i^2},\gamma\right\}$, and will be omitted.

We are now ready to proceed with developing the algorithm. The overall idea is that we proceed within the same framework as in the case where the priors are viewed as variational parameters, and use the same updating scheme for the mixing matrix and noise covariance. Now, as stated above, we will use equation (5.26) to update our estimate of the z_i. Since the sources are not directly available, we will use their current estimate in said equation, in other words

$$\begin{aligned}z_i^{new}&=E[z_i|s_i=\langle s_i|\mathbf{x}\rangle]\\&=\sqrt{\frac{\mu^2_{s_i|\mathbf{x}}}{2\lambda}}\frac{K_{\alpha+\frac{3}{2}}(\sqrt{2\lambda\mu^2_{s_i|\mathbf{x}}})}{K_{\alpha+\frac{1}{2}}(\sqrt{2\lambda\mu^2_{s_i|\mathbf{x}}})}\end{aligned} \tag{5.27}$$

To update the parameters α_i and λ_i needed in the above calculation, we will use a simple moment based estimator of the form (see e.g. [11])

$$\hat{\alpha}_i=\frac{1}{\frac{\overline{z_i^2}}{\overline{z}_i^2}-1}-1 \tag{5.28}$$

$$\hat{\lambda}_i=\frac{\hat{\alpha}_i+1}{\overline{z}_i} \tag{5.29}$$

Here $\overline{z}_i$ and $\overline{z_i^2}$ are calculated from the current estimate of s_i, using the following relations which are easily derived from equation (5.2), utilising the independence of z and y:

$$\overline{z}_i=E[z_i]=E[s_i^2]=\frac{1}{N}\sum_{i=1}^N s_i^2 \tag{5.30}$$

$$\overline{z_i^2}=E[z_i^2]=\frac{E[s_i^4]}{3}=\frac{1}{3N}\sum_{i=1}^N s_i^4 \tag{5.31}$$

The algorithm thus proceeds by alternately setting z_i to its posterior mean, and setting $s_i=E[s_i|\mathbf{x},z_i]$.

Table 5.2: Algorithm for scale mixture source separation with Gamma distributed hyper prior

1. Center the observations
2. Initialise $\mathbf{z}_n$ for all n, as well as $\mathbf{A}$, $\boldsymbol{\Sigma_\nu}$, $\boldsymbol{\lambda}$ and $\boldsymbol{\alpha}$
3. For all n, find $\langle \mathbf{s}|\mathbf{x}_n \rangle = \boldsymbol{\mu}_{\mathbf{s}|\mathbf{x}_n}$ by means of equation (5.23), using the current estimates of $\boldsymbol{\Lambda}_n = \mathrm{diag}(\mathbf{z}_n)$, $\mathbf{A}$ and $\boldsymbol{\Sigma_\nu}$
4. Update $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ based on estimate of $\mathbf{s}$ from previous step, by way of equations 5.28 through 5.31
5. For all n, update $z_i(n)$ from equation (5.27)
6. Update the estimate of $\mathbf{s}$, again using equation (5.23), with the updated estimates of $\mathbf{z}$
7. Update $\mathbf{A}$ and $\boldsymbol{\Sigma_\nu}$ using equations (5.17) and (5.18).
8. If not converged, return to step 3.

For clarity, the steps are summed up in table (5.2). The notation is to be understood as follows: $\boldsymbol{\lambda}$ and $\boldsymbol{\alpha}$ are L by 1 vectors consisting of shape and scale parameters for the hyper priors, $\mathbf{x}_n = [x_1(n), \ldots, x_{L'}(n)]$ and $\mathbf{z}_n = [z_1(n), \ldots, z_L(n)]$.

5.4 Comments

The derivation of the algorithm in table 5.2 has been of a slightly ad hoc nature. The results, however, are consistent with existing techniques. Considering the work of J. A Palmer [26] [25] [20] we make use of the following relationship, which applies to all Gaussian scale mixture densities of the form of equation (5.2).

$$\begin{aligned} p'(s) &= \frac{\partial}{\partial s} \int_0^\infty p(s|z)p(z)\,dz = -\int_0^\infty \frac{1}{z} s\, p(s,z)\,dz \\ &= -s\, p(s) \int_0^\infty \frac{1}{z} p(z|s)\,dz \end{aligned}$$

Hence, we see that

$$E[\frac{1}{z_i}|s_i] = \int_0^\infty \frac{1}{z_i} p(z_i|s_i)\, dz = -\frac{p'(s_i)}{s_i\, p(s_i)} = \frac{f'(s_i)}{s_i} \tag{5.32}$$

where $f(s_i) = -\log p(s_i)$. Now, calculating $f(s_i)$ for the case of a Gamma distributed prior (in other words a K-distributed s_i) we find

$$\begin{aligned} f(s_i) &= -\log p(s_i) \\ &= -(\alpha + \frac{1}{2}) \log |s_i| - \log K_{\alpha+\frac{1}{2}} \left(\sqrt{2\lambda s_i^2} \right) + \text{const w.r.t } s_i \end{aligned} \tag{5.33}$$

so

$$\begin{aligned} f'(s_i) &= -(\alpha + \frac{1}{2}) \frac{1}{s_i} + \frac{\sqrt{2\lambda s_i^2}}{s_i} \left(K_{\alpha-\frac{1}{2}} \left(\sqrt{2\lambda s_i^2} \right) \right. \\ &\quad \left. + \frac{\alpha + \frac{1}{2}}{\sqrt{2\lambda s_i^2}} K_{\alpha+\frac{1}{2}} \left(\sqrt{2\lambda s_i^2} \right) \right) \frac{1}{K_{\alpha+\frac{1}{2}} \left(\sqrt{2\lambda s_i^2} \right)} \\ &= \frac{\sqrt{2\lambda s_i^2}}{s_i} \frac{K_{\alpha-\frac{1}{2}} \left(\sqrt{2\lambda s_i^2} \right)}{K_{\alpha+\frac{1}{2}} \left(\sqrt{2\lambda s_i^2} \right)} \end{aligned} \tag{5.34}$$

and

$$\begin{aligned} E[\frac{1}{z_i}|s_i] &= \frac{f'(s_i)}{s_i} \\ &= \sqrt{\frac{2\lambda}{s_i^2}} \frac{K_{\alpha-\frac{1}{2}} \left(\sqrt{2\lambda s_i^2} \right)}{K_{\alpha+\frac{1}{2}} \left(\sqrt{2\lambda s_i^2} \right)} \end{aligned} \tag{5.35}$$

corresponding nicely with equation (5.26) for the case of $k = -1$. In the work of Palmer, the z_i are updated as the reciprocal of equation (5.35)(actually, due to a slightly different definition of the scale mixture model, he estimates z_i, and its reciprocal make out the diagonal elements of $\mathbf{\Lambda}$).

The problem we have examined in this chapter, has been approached in a number of different ways in existing literature. Worth mentioning, apart from Palmer's research, are the approaches taken in e.g. [12] [3], where they approximate the joint posterior (in our case $p(\mathbf{s}, \mathbf{z}|\mathbf{x})$) as a *variational posterior* $q(\mathbf{s}, \mathbf{z}|\mathbf{x})$ which is restricted to the factorised form

$$q(\mathbf{s}, \mathbf{z}|\mathbf{x}) = q(\mathbf{s}|\mathbf{x}) q(\mathbf{z}|\mathbf{x}) \tag{5.36}$$

This approximation renders it possible to maximise the likelihood through the EM algorithm. This is referred to as a variational Bayesian framework. Also worthy of mention, is the variational approach taken by M. Girolami in [13]. He proceeds by expressing the assumed Laplacian source densities as functions of an additional parameter ξ (which corresponds to the hyper prior z in our case). Due to the convexity of the source densities, he utilises some topological concepts that allow him to find a function that is bounded from below by the source densities. This, in turn, allows him to formulate an EM algorithm for source separation. Interestingly, the algorithm thus derived, differs from that in table 5.1 only in its estimate of z_i, which in his case is estimated as $z_i = \sqrt{\langle s_i^2 | \mathbf{x} \rangle}$, compared to $z_i = \langle s_i^2 | \mathbf{x} \rangle$ in our algorithm. The performance of the respective algorithms will be compared in chapter 7.

Finally it must be mentioned that, though not explicitly taken into account, our algorithms are expected to achieve separation only for super-Gaussian sources. This stems from the fact that only in this case is the EM algorithm guaranteed to increase the likelihood. For a thorough discussion on why this is so, I refer again to [26].

Chapter 6

Denoising Source Separation

6.1 Introduction

Denoising source separation (DSS) is an algorithmic framework for source separation introduced by J. Särelä and H. Valpola [28]. The framework is meant to facilitate in the development of source separation algorithms which are optimized for specific problems, the main idea being to construct the algorithms around denoising principles. This very limited presentation of DSS will mainly concern itself with the derivation of the formulae constituting the basic building blocks of the algorithms constructed within said framework, since some of the steps omitted in [28] nontrivial. Since DSS will occupy only a minor part of the experimental part of this thesis, we will not concern ourselves with any in-depth discussion around practical denoising schemes.

6.2 The Model

The basic model we are considering is again an instantaneous mixture model:

$$\mathbf{X} = \mathbf{AS} + \boldsymbol{\nu} \tag{6.1}$$

where

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_M \end{bmatrix} \quad , \quad \mathbf{S} = \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \\ \vdots \\ \mathbf{s}_N \end{bmatrix}$$

The N vectors $\mathbf{s}_i$ represent the source signals and the M vectors $\mathbf{x}_i$ represent their mixtures. $\mathbf{s}_i = [s_i(1) \dots s_i(t) \dots s_i(T)]$ and

$\mathbf{x}_i = [x_i(1) \dots x_i(t) \dots x_i(T)]$ are both of length T. In the model the columns of $\mathbf{A}$, $\mathbf{a}_i = [a_{1i}a_{2i} \cdots a_{Mi}]^T$, are called the mixing vectors since the mixtures $\mathbf{x}(t) = [x_1(t)x_2(t) \dots x_M(t)]^T$ at a given time instance t can be written as a linear combination of them (see 3.2). In the following we assume that the sources, the noise and consequently the mixtures, are of zero mean. This can be done without losing generality since the mean always can be subtracted from the data. In general the independent noise will be assumed normally distributed with covariance matrix $\mathbf{\Sigma}_\nu$.

6.3 Deriving the learning rules

Again the starting point for deriving the learning rules is the EM algorithm. In this case we use the formulation

1. E-step: compute

$$q(\mathbf{S}) = p(\mathbf{S}|\mathbf{A}, \mathbf{X}) = \frac{p(\mathbf{X}, \mathbf{A}|\mathbf{S})p(\mathbf{S})}{p(\mathbf{X}, \mathbf{A})} = \frac{p(\mathbf{X}|\mathbf{A}, \mathbf{S})p(\mathbf{S})}{p(\mathbf{X}|\mathbf{A})} \tag{6.2}$$

2. M-step: find

$$\mathbf{A}_{new} = \arg\max_{\mathbf{A}} \mathrm{E}_{q(\mathbf{S})}[\log p(\mathbf{S}, \mathbf{X}|\mathbf{A})] \tag{6.3}$$

This maximisation has already been dealt with in slightly different formulation in previous chapters, yielding, with our present notation

$$\mathbf{A}_{new} = \mathbf{C}_{\mathbf{XS}}\mathbf{C}_{\mathbf{SS}}^{-1} \tag{6.4}$$

where

$$\mathbf{C}_{\mathbf{XS}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{x}(t)E[\mathbf{s}(t)^T|\mathbf{X}, \mathbf{A}] \tag{6.5}$$

$$\mathbf{C}_{\mathbf{SS}} = \frac{1}{T}\sum_{t=1}^{T} E[\mathbf{s}(t)\mathbf{s}(t)^T|\mathbf{X}, \mathbf{A}] \tag{6.6}$$

We will now assume that the data have been whitened such that $\mathbf{C}_{\mathbf{XX}} = \mathbf{I}$. Also, we will follow the standard procedure of fixing the variance of the sources to unity. So, together with the assumption of independence, $\mathbf{C}_{\mathbf{SS}} = \mathbf{I}$. If we also assume that $\mathbf{\Sigma}_\nu \propto \mathbf{I}$ we have that $\mathbf{C}_{\mathbf{XX}} = \mathbf{A}\mathbf{C}_{\mathbf{SS}}\mathbf{A}^T + \mathbf{\Sigma}_\nu = \mathbf{A}\mathbf{A}^T + \sigma_\nu^2\mathbf{I} = \mathbf{I}$, which implies that $\mathbf{A}\mathbf{A}^T \propto \mathbf{I}$. Consequently the mixing matrix $\mathbf{A}$ is orthogonal (up to scaling) for

whitened data (in the noiseless case it is of course orthogonal, as shown in 3.3.1).
A closer look at the likelihood of $\mathbf{S}$ reveals that it can be factorised with respect to the individual sources ($\tilde{\mathbf{a}}_i$ denotes row i of $\mathbf{A}$):

$$\begin{aligned}
L(\mathbf{S}) = p(\mathbf{X}|\mathbf{A},\mathbf{S}) &= \prod_{t=1}^{T} p(\mathbf{x}(t)|\mathbf{A},\mathbf{s}(t)) \\
&= \prod_{t=1}^{T} \frac{1}{(2\pi)^{\frac{M}{2}}|\mathbf{\Sigma}_\nu|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}(t)-\mathbf{A}\mathbf{s}(t))^T\mathbf{\Sigma}_\nu^{-1}(\mathbf{x}(t)-\mathbf{A}\mathbf{s}(t))\right) \\
&= \frac{1}{(2\pi)^{\frac{MT}{2}}|\mathbf{\Sigma}_\nu|^{\frac{T}{2}}} \exp\left(-\frac{1}{2\sigma_\nu^2}\sum_{t=1}^{T}\sum_{i=1}^{M}\left(x_i(t)-\tilde{\mathbf{a}}_i\mathbf{s}(t)\right)^2\right) \\
&= \frac{1}{(2\pi)^{\frac{MT}{2}}|\mathbf{\Sigma}_\nu|^{\frac{T}{2}}} \exp\left(-\frac{1}{2\sigma_\nu^2}\sum_{i=1}^{M}(\mathbf{x}_i-\tilde{\mathbf{a}}_i\mathbf{S})(\mathbf{x}_i-\tilde{\mathbf{a}}_i\mathbf{S})^T\right) \\
&= \frac{1}{(2\pi)^{\frac{MT}{2}}|\mathbf{\Sigma}_\nu|^{\frac{T}{2}}} \exp\left(-\frac{1}{2\sigma_\nu^2}\mathrm{Tr}\left((\mathbf{X}-\mathbf{A}\mathbf{S})(\mathbf{X}-\mathbf{A}\mathbf{S})^T\right)\right) \\
&= \frac{1}{(2\pi)^{\frac{MT}{2}}|\mathbf{\Sigma}_\nu|^{\frac{T}{2}}} \exp\left(-\frac{1}{2\sigma_\nu^2}\mathrm{Tr}\left(\mathbf{A}^T\mathbf{A}(\mathbf{S}-\mathbf{A}^{-1}\mathbf{X})(\mathbf{S}-\mathbf{A}^{-1}\mathbf{X})^T\right)\right) \\
&= \frac{1}{(2\pi)^{\frac{MT}{2}}|\mathbf{\Sigma}_\nu|^{\frac{T}{2}}} \exp\left(-\frac{1}{2}\sum_{i=1}^{N}\frac{\mathbf{a}_i^T\mathbf{a}_i}{\sigma_\nu^2}(\mathbf{s}_i-\mathbf{a}_i^{-1}\mathbf{X})(\mathbf{s}_i-\mathbf{a}_i^{-1}\mathbf{X})^T\right) \\
&= \frac{1}{(2\pi)^{\frac{MT}{2}}|\mathbf{\Sigma}_\nu|^{\frac{T}{2}}} \prod_{i=1}^{N} \exp\left(-\frac{1}{2}(\mathbf{s}_i-\mathbf{a}_i^{-1}\mathbf{X})\mathbf{\Sigma}_{\mathbf{s},\nu}^{-1}(\mathbf{s}_i-\mathbf{a}_i^{-1}\mathbf{X})^T\right) \\
&= C\prod_{i=1}^{N} L(\mathbf{s}_i) \qquad (6.7)
\end{aligned}$$

Here $\mathbf{\Sigma}_{\mathbf{s},\nu}$ is a $T \times T$ diagonal matrix with elements $\sigma_\nu^2/\mathbf{a}_i^T\mathbf{a}_i$ along the diagonal, and $\mathbf{a}_i^{-1}$ is the ith row of $\mathbf{A}^{-1}$.
This means that the sources are independent in the posterior $q(\mathbf{s})$ since all terms involving $\mathbf{S}$ in (6.2) factorise in terms of the individual sources $\mathbf{s}_i$. In turn, this implies that $\mathbf{C_{SS}}$ and $\mathbf{C_{SS}^{-1}}$ are diagonal, and multiplication by $\mathbf{C_{SS}^{-1}}$ in (6.4) reduces to a simple scaling of the individual sources, which can be ignored in the further development of the algorithm.
Noisy estimates of the sources can be recovered as the mode of the likelihood, i.e. $\mathbf{S} = \mathbf{A}^{-1}\mathbf{X}$. We have already shown that for whitened data and the appropriate assumptions, $\mathbf{A}^{-1} \propto \mathbf{A}^T$, and since the posterior $q(\mathbf{S})$ depends on the data only through the likelihood $L(\mathbf{S})$, the expectation

Table 6.1: The basic DSS algorithm

1. Centre and whiten the observations

2. Initialize $\mathbf{w}$ (e.g. randomly)

3. Find a noisy estimate of one source as the mode of the likelihood in equation (6.7)

$$\mathbf{s} = \mathbf{w}^T\mathbf{X} \tag{6.8}$$

4. Find the posterior expectation of $\mathbf{s}$. This can be viewed as performing denoising based on the model of the sources

$$\mathbf{s}^+ = \mathbf{f}(\mathbf{s}) \tag{6.9}$$

5. Calculate the new ML estimate of the mixing vector $\mathbf{w}$ from (6.5), normalising to prevent divergence (remember $\mathbf{A}_{new} = \mathbf{C_{XS}}\mathbf{C_{SS}^{-1}}$, and since $\mathbf{C_{SS}^{-1}}$ is a simple scaling it can be ignored)

$$\mathbf{w}^+ = \mathbf{X}\mathbf{s}^+ \tag{6.10}$$

$$\mathbf{w}_{new} = \frac{\mathbf{w}^+}{\|\mathbf{w}^+\|} \tag{6.11}$$

6. If not converged, return to step 3.

$E[\mathbf{S}|\mathbf{X}, \mathbf{A}]$ is a function of $\mathbf{A}^T\mathbf{X}$. Or, in terms of the individual sources, $E[\mathbf{s}_i|\mathbf{X}, \mathbf{A}] = \mathbf{f}(\mathbf{a}_i^T\mathbf{X})$. This leads to the one-unit source separation algorithm summed up in table 6.1 ($\mathbf{w} = \mathbf{a}$, following the notation in [28]).

The development of algorithms within the DSS framework will thus proceed by choosing the denoising function $\mathbf{f}(\mathbf{s})$ appropriately for the model under consideration. The case of linearly mixed Gaussian signals is discussed next.

6.3.1 Separating Gaussian Sources (Linear DSS)

We will in this section find the denoising function $\mathbf{f}(\mathbf{s})$, for the case of a Gaussian source with autocovariance matrix $\mathbf{\Sigma_{ss}}$. Its prior probability function is

$$p(\mathbf{s}) = \frac{1}{\sqrt{|2\pi\mathbf{\Sigma_{ss}}|}} \exp\left(-\frac{1}{2}\mathbf{s}\mathbf{\Sigma_{ss}^{-1}}\mathbf{s}^T\right). \tag{6.12}$$

We remind the reader that $\mathbf{s}$ is here to be understood as a single source $\mathbf{s} = [s(1), \ldots, s(T)]$, and is not to be confused with the random variable consisting of all sources, as the notation has been in earlier chapters. Accordingly, the autocovariance matrix $\mathbf{\Sigma_{ss}}$ is $T \times T$. The likelihood is obtained from equation (6.7) as

$$L(\mathbf{s}) = \exp\left(-\frac{1}{2}(\mathbf{s} - \mathbf{w}^T\mathbf{X})\mathbf{\Sigma_{\mathbf{s},\nu}^{-1}}(\mathbf{s} - \mathbf{w}^T\mathbf{X})^T\right), \tag{6.13}$$

where we have used the orthogonality of the mixing matrix to replace the inverse with the transpose. As argued for in the previous chapter, the posterior can be written in terms of individual sources. Multiplying $p(\mathbf{s})$ and $L(\mathbf{s})$, the only terms in equation (6.2) that involve $\mathbf{s}$, we obtain (after completing the square as in chapter (4.3)):

$$q(\mathbf{s}) = (\text{const. w.r.t } \mathbf{s}) \times \exp\left(-\frac{1}{2}(\mathbf{s} - \boldsymbol{\mu})\mathbf{\Sigma}^{-1}(\mathbf{s} - \boldsymbol{\mu})^T\right), \tag{6.14}$$

Where $\boldsymbol{\mu} = \mathbf{w}^T\mathbf{X}\left(\mathbf{I} + \sigma_\nu^2\mathbf{\Sigma_{ss}^{-1}}\right)^{-1}$, and $\mathbf{\Sigma}^{-1} = \frac{1}{\sigma_\nu^2} + \mathbf{\Sigma_{ss}^{-1}}$. Referring to equation (6.9), we see that, it in this case, takes the linear form

$$\mathbf{s}^+ = \mathbf{f}(\mathbf{s}) = \mathbf{s}(\mathbf{I} + \sigma_\nu^2\mathbf{\Sigma_{ss}^{-1}})^{-1} = \mathbf{s}\mathbf{D}. \tag{6.15}$$

In the simulations chapter, it will be demonstrated that two Gaussian signals that have been mixed linearly, can be separated through use of the DSS algorithm, using the denoising function above.

Part II

Computer Simulations

Chapter 7

Simulations

7.1 Introduction

In this chapter we will try to evaluate the performance of the different source separation techniques discussed in the theory part of this thesis. Most of the algorithms will be tested on the same data, with varying signal-to-noise ratio. One exception is our treatment of the DSS technique. Though, being a general framework for source separation, it could conceivably be used on the same test data as the other techniques. The obvious choice for the denoising function in this case would reduce it to the standard FastICA algorithm, which is already covered. Instead, therefore, we will demonstrate how it can be used to incorporate temporal information so as to achieve separation of linearly mixed Gaussian signals, something that standard ICA techniques cannot do. As such, it serves to demonstrate how additional knowledge (here in the form of the knowing the autocovariance matrix of the Gaussian signals) can render possible separation where ICA, which exclusively uses spatial information (as in information across the sensors/observations), fails.

7.2 The Data

In most of the simulations in this chapter, the data will consist of three audio signals, sampled at 22050 Hz. Since some of the algorithms to be tested are computationally quite expensive, we have used a modest 14000 samples, constituting around 0.6 seconds of sound. This should nonetheless be more than adequate to capture the relevant statistics of the signals in question. In any practical implementation, the algorithms would normally be performed in batch mode anyway, with the batch size being of the order

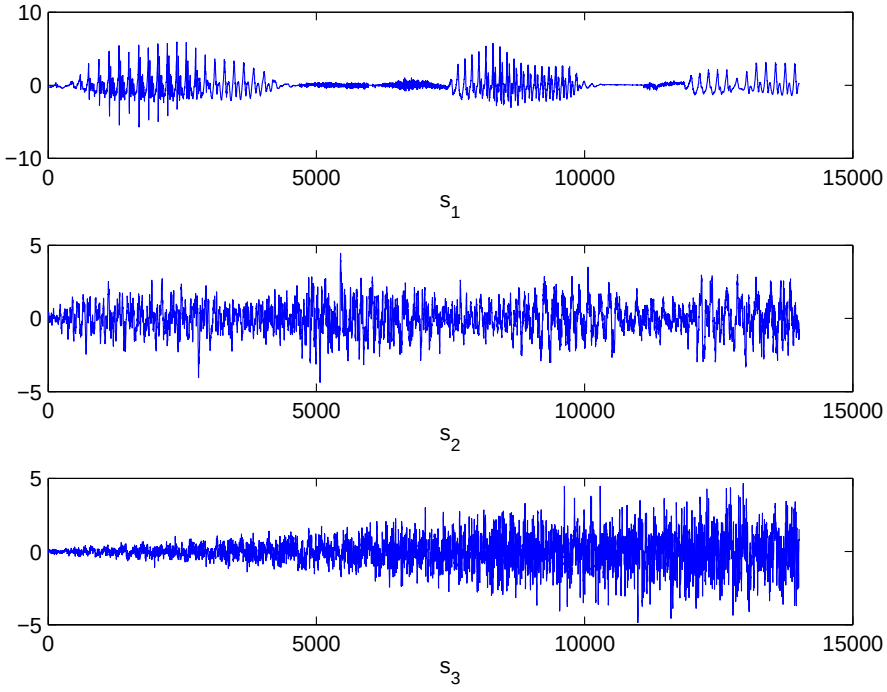


Figure 7.1: The centered sources

we have used, or smaller.
Due to the indeterminacies of the model, we have chosen to center and normalize the source signals. While not necessary to achieve separation, this makes the task of evaluating the performance of the algorithms slightly easier. The centered and normalized sources are shown in figure 7.1.

In keeping with the purpose of this thesis, namely the study of *instantaneous* linear mixture models, the mixing has been done artificially, through multiplication of a three by three mixing matrix, thus ensuring that the basic model holds. Any real world mixing of audio signals would of course be complicated by propagation delays and reverberations, necessitating a convolutive model (or an embedding of the signals, as discussed briefly in the introduction to this thesis). This, however, is beyond the scope of this thesis. To further control the mixing environment, the same mixing matrices have been used in the simulations for the different models. Specifically the instamix.m Matlab function (available for download from http://sound.media.mit.edu/ica-bench/) has been used,

which performs mixing with varying degrees of complexity. This function is parameterised by a condition number, b, where b=0 represents a singular matrix (consisting of only ones). The mixing matrix becomes proportional to the identity matrix with increasing values of b (which can be arbitrarily large). In our simulations we have used b=0.5.

Simulations have been performed at six different signal-to-noise ratios (SNR). The SNR is obtained by noting that the signal level at observation i is $E[(\sum_j A_{ij}s_j)^2] = \sum_j (A_{ij})^2$, since we have normalised the sources (they are of course also statistically independent, so $E[\mathbf{s}\mathbf{s}^T] = \mathbf{I}$). The corresponding noise level is $E[\nu_i^2] = \Sigma_{ii}$. Averaging over all the observations, we get

$$\text{SNR} = \frac{1}{L'} \sum_{i=1}^{L'} \left[\sum_{j=1}^{L} (A_{ij})^2 \right] / \Sigma_{ii} \tag{7.1}$$

For the purpose of our simulations, noise was added to each observation from a zero mean Gaussian distribution with variance

$$\Sigma_{ii} = \frac{1}{L'} \sum_{i=1}^{L'} \left[\sum_{j=1}^{L} (A_{ij})^2 \right] / \text{SNR}, \tag{7.2}$$

yielding the desired SNR. So as to avoid confusion, it must be stressed that we here are referring to the SNR of the *observations* x_i, which is not to be confused with the SNR of the recovered *sources*.

7.3 Performance Measures

There exist a multitude of different measures for evaluating the performance of source separation techniques, and the pros and cons of the various measures have been subject to much discussion (see e.g. [23] [10]). The most rigorous attempt at defining a general framework for performance measurement seems to be that of Vincent et. al. [10], and since they have provided downloadable Matlab software on their homepage, this will be used in this thesis.

The general idea in the above mentioned framework, is to decompose the estimated sources $\hat{s}_i$ as

$$\hat{s}_i = s_{\text{target}} + e_{\text{interf}} + e_{\text{noise}} + e_{\text{artif}} \tag{7.3}$$

where $s_{\text{target}} = f(s_i)$ is a version of the actual source s_i it is being compared with, modified by an allowed distortion, in our simple case a gain factor. The terms e_{interf}, e_{noise} and e_{artif} are, respectively, the interference, noise, and artifacts error term. Thus s_{target} represents the part of $\hat{s_i}$ perceived as coming from the wanted source s_i, e_{interf} the part coming from other unwanted sources $(s_j)_{j\neq i}$, e_{noise} the part coming from sensor noise, and e_{artif} from other causes. To perform this decomposition, one needs both the estimated sources as well as the original sources and noise. These are of course available for the simulated data under consideration in this chapter. The decomposition is performed by projecting orthogonally the estimated sources $\hat{s_i}$ onto a subspace spanned by the relevant vectors (For details I refer to [10]). From these decompositions, four performance criteria are defined. The *source-to-distortion* ratio is defined

$$\text{SDR}:=10\log_{10}\frac{\|s_{\text{target}}\|^2}{\|e_{\text{interf}}+e_{\text{noise}}+e_{\text{artif}}\|^2} \tag{7.4}$$

the *source-to-interferences* ratio

$$\text{SIR}:=10\log_{10}\frac{\|s_{\text{target}}\|^2}{\|e_{\text{interf}}\|^2} \tag{7.5}$$

the *source-to-noise* ratio

$$\text{SNR}:=10\log_{10}\frac{\|s_{\text{target}}+e_{\text{interf}}\|^2}{\|e_{\text{noise}}\|^2} \tag{7.6}$$

$$\tag{7.7}$$

and the *source-to-artifacts* ratio

$$\text{SNR}:=10\log_{10}\frac{\|s_{\text{target}}+e_{\text{interf}}+e_{\text{noise}}\|^2}{\|e_{\text{artif}}\|^2} \tag{7.8}$$

$$\tag{7.9}$$

We will content ourselves with calculating the first three of these measures. In order to have a measure for each models performance as a whole (as opposed to having one for every source), we will also include what Attias [4] refers to as the mixing matrix error

$$\epsilon = \left(\frac{1}{L^2-L}\sum_{i\neq j}J_{ij}^2\right)\left(\frac{1}{L}\sum_{i=1}^{L}J_{ii}\right)^{-1} \tag{7.10}$$

Here J_{ij} are the elements of the matrix $\mathbf{J} = \hat{\mathbf{A}}^{-1}\mathbf{A}$, where $\hat{\mathbf{A}}$ and $\mathbf{A}$ are the estimated and actual mixing matrix, respectively. Thus ϵ quantifies the distance between $\mathbf{J}$ and the identity matrix as the mean squared nondiagonal elements of $\mathbf{J}$, normalised by its mean squared diagonal elements. To compensate for any permutations that may have occurred in the separation algorithm, the columns of the estimated mixing matrix have been rearranged to ensure that $\mathbf{J}$ has its largest element along the diagonal.

7.4 Results

In this section, the performance of the following six algorithms will be presented.

1. The scale mixture of Gaussians algorithm with no assumptions on the hyper prior in table 5.1. This will be referred to as the *No prior* algorithm.

2. The scale mixture of Gaussians algorithm with Γ distributed hyper prior in table 5.2. This will be referred to in the tables as Γ *prior*.

3. The scale mixture of Gaussians algorithm for IG distributed hyper prior. This will be referred to as the *IG prior* algorithm.

4. Girolami's algorithm from [13]. This is identical to the *No prior* algorithm, except for the update of the z_i which is calculated as $z_i = \sqrt{\langle s_i^2 | \mathbf{x} \rangle}$, compared to $z_i = \langle s_i^2 | \mathbf{x} \rangle$ in the *No prior* algorithm.

5. The FastICA algorithm presented in chapter 3.5.2. We use the FastICA package for Matlab available for download from www.cis.hut.fi/projects/ica/fastica

6. The Independent Factor Analysis algorithm from chapter 4. We have modelled each source density by a 3-state MOG, ie $n_i = 3$ in equation (4.5)

The detailed results from running each of these algorithms once, are presented in tables 7.1 through 7.6. While there is undoubtedly a wealth of information to be obtained from the many measures that are included, we will in the discussion mainly concern ourselves with the global mixing error measure ϵ from equation 7.10. The other measures are useful when one wishes to evaluate the quality of separation and recovery for a single source, and we will in the discussion compare this with the quality of the estimated

pdfs for some selected sources. Apart from this use, the measures are tabulated for the interested reader since they are likely to become the standard performance measures for blind audio source separation. (Note: These measures give information about the reconstruction of the sources, whereas ϵ only concerns itself with the demixing part of the problem, ignoring the presence of noise).

In figure 7.2 the mixing matrix error ϵ is plotted for all algorithms and all SNRs.

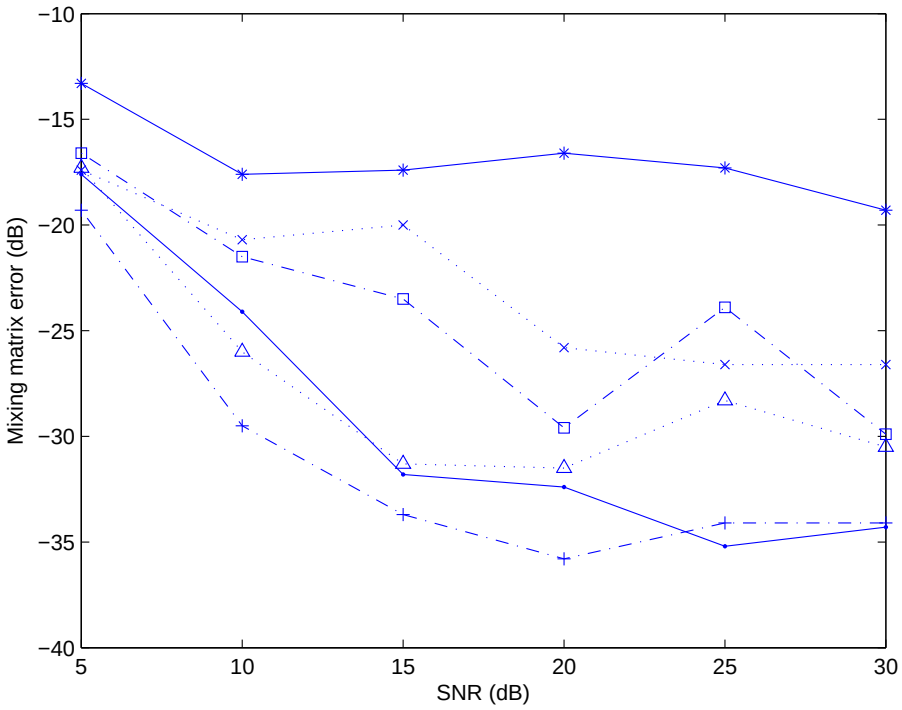


Figure 7.2: The mixing error of the various techniques at different signal to noise ratios. Included in the plot is Girolami's method (dash-dotted with squares), the IG distributed hyper prior method (dotted line with x's), Γ distributed hyper prior method (solid line with dots), non-random hyper prior method (solid line with stars), FastICA (dotted line with triangles), and IFA (dash-dotted with +'s).

Table 7.1: Performance of algorithms for SNR = 5 dB

Method	SDR (dB)			SIR (dB)			SNR (dB)			ϵ (dB)
	$\hat{s_1}$	$\hat{s_2}$	$\hat{s_3}$	$\hat{s_1}$	$\hat{s_2}$	$\hat{s_3}$	$\hat{s_1}$	$\hat{s_2}$	$\hat{s_3}$	
No prior	1.6	1.8	1.7	15.8	8.1	10.1	1.9	3.6	2.8	-13.3
Γ prior	2.0	1.4	1.5	12.9	15.9	17.6	2.6	1.7	1.7	-17.6
IG prior	1.6	1.2	1.8	18.0	19.7	10.7	1.8	1.3	2.8	-17.4
Girolami	1.0	1.9	1.3	24.0	9.9	17.5	1.0	3.1	1.5	-16.6
FastICA	1.9	1.1	1.3	11.4	31.4	15.1	2.7	1.1	1.6	-17.3
IFA	2.1	1.4	1.8	13.6	22.3	13.1	2.6	1.4	2.3	-19.3

Table 7.2: Performance of algorithms for SNR = 10 dB

Method	SDR (dB)			SIR (dB)			SNR (dB)			ϵ (dB)
	$\hat{s_1}$	$\hat{s_2}$	$\hat{s_3}$	$\hat{s_1}$	$\hat{s_2}$	$\hat{s_3}$	$\hat{s_1}$	$\hat{s_2}$	$\hat{s_3}$	
No prior	6.2	5.9	6.1	14.0	12.7	14.7	7.2	7.2	6.9	-16.8
Γ prior	6.4	5.8	6.4	20.3	30.3	19.3	6.6	5.9	6.6	-24.1
IG prior	6.1	6.2	6.4	24.0	14.4	20.3	6.2	7.1	5.8	-20.7
Girolami	6.1	5.7	6.1	20.6	20.9	15.5	6.4	5.4	6.8	-21.5
FastICA	6.2	5.9	6.0	21.0	43.6	21.1	6.4	5.9	6.2	-26.0
IFA	7.0	6.2	6.3	19.8	26.7	22.5	7.7	6.3	6.5	-29.5

Table 7.3: Performance of algorithms for SNR = 15 dB

Method	SDR (dB)			SIR (dB)			SNR (dB)			ϵ (dB)
	$\hat{s_1}$	$\hat{s_2}$	$\hat{s_3}$	$\hat{s_1}$	$\hat{s_2}$	$\hat{s_3}$	$\hat{s_1}$	$\hat{s_2}$	$\hat{s_3}$	
No prior	10.7	11.0	10.7	17.3	25.0	19.1	12.0	11.2	11.5	-22.6
Γ prior	11.1	10.6	11.0	35.2	29.2	26.2	11.1	10.7	11.2	-31.8
IG prior	10.2	9.3	10.7	16.5	15.2	23.7	11.4	10.6	11.0	-20.0
Girolami	10.9	10.5	10.2	35.0	17.4	20.7	10.9	11.6	10.6	-23.5
FastICA	10.8	11.0	10.8	33.6	25.3	30.5	10.8	11.2	10.8	-31.3
IFA	11.5	10.4	11.1	31.5	26.4	28.6	12.0	10.7	12.1	-33.7

Table 7.4: Performance of algorithms for SNR = 20 dB

Method	SDR (dB)			SIR (dB)			SNR (dB)			ϵ (dB)
	$\hat{s_1}$	$\hat{s_2}$	$\hat{s_3}$	$\hat{s_1}$	$\hat{s_2}$	$\hat{s_3}$	$\hat{s_1}$	$\hat{s_2}$	$\hat{s_3}$	
No prior	15.9	13.9	15.3	31.1	17.4	22.4	16.1	16.7	16.3	-23.9
Γ prior	15.9	15.1	16.0	40.7	25.4	31.4	16.0	15.6	16.1	-32.4
IG prior	15.9	14.3	15.1	44.3	20.7	21.7	15.9	15.4	16.2	-25.8
Girolami	15.9	15.1	15.5	35.3	24.5	25.0	16.0	15.7	16.1	-29.6
FastICA	15.4	15.8	15.6	26.9	28.1	29.7	15.7	16.1	15.8	-31.5
IFA	16.3	15.5	15.9	40.7	29.9	33.8	17.0	15.8	16.5	-35.8

Table 7.5: Performance of algorithms for SNR = 25 dB

Method	SDR (dB)			SIR (dB)			SNR (dB)			ϵ (dB)
	$\hat{s_1}$	$\hat{s_2}$	$\hat{s_3}$	$\hat{s_1}$	$\hat{s_2}$	$\hat{s_3}$	$\hat{s_1}$	$\hat{s_2}$	$\hat{s_3}$	
No prior	20.6	8.2	10.7	30.8	8.4	11.3	21.1	22.5	20.0	-14.5
Γ prior	20.7	19.8	21.0	35.6	28.0	38.7	20.9	20.7	21.1	-35.2
IG prior	19.4	17.4	20.4	24.9	20.7	28.6	20.9	20.5	21.1	-26.6
Girolami	19.6	16.5	17.9	26.1	18.7	20.6	20.7	20.5	21.35	-23.9
FastICA	20.0	18.4	20.2	26.5	22.7	29.4	21.2	20.5	20.8	-28.3
IFA	20.4	19.9	19.8	55.8	31.0	35.0	22.0	21.0	21.6	-34.1

Table 7.6: Performance of algorithms for SNR = 30 dB

Method	SDR (dB)			SIR (dB)			SNR (dB)			ϵ (dB)
	$\hat{s_1}$	$\hat{s_2}$	$\hat{s_3}$	$\hat{s_1}$	$\hat{s_2}$	$\hat{s_3}$	$\hat{s_1}$	$\hat{s_2}$	$\hat{s_3}$	
No prior	25.8	20.3	23.0	42.3	21.7	26.2	26.2	26.0	26.0	-28.4
Γ prior	25.3	23.2	25.7	34.7	27.5	39.5	25.9	25.7	26.0	-34.3
IG prior	23.8	19.6	22.3	27.7	20.9	25.1	26.1	25.6	25.6	-26.6
Girolami	25.4	21.0	25.0	34.9	22.8	32.3	25.9	25.7	25.9	-29.9
FastICA	22.1	24.8	24.3	24.8	30.8	29.5	25.6	26.0	25.8	-30.5
IFA	23.4	19.3	23.7	49.4	26.3	37.5	27.0	25.6	26.4	-34.1

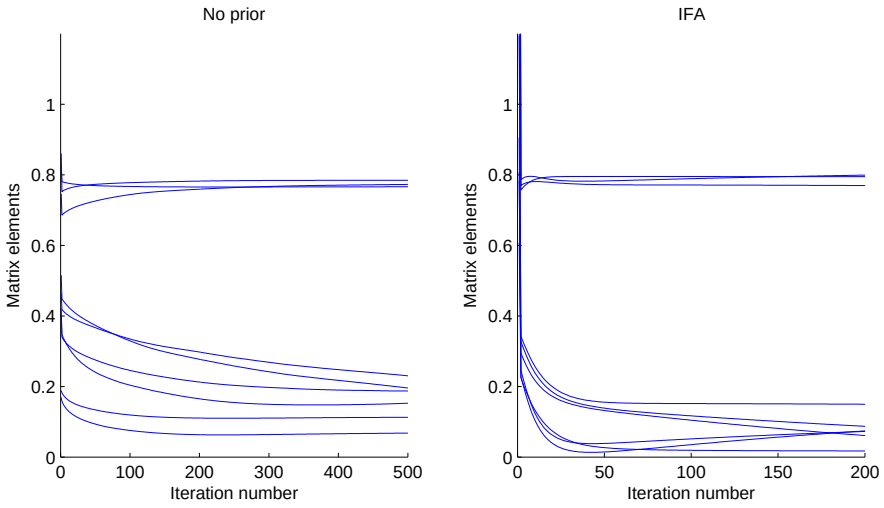


Figure 7.3: Illustration of the convergence of the No prior algorithm, and of IFA for SNR=5dB. Plotted are the matrix elements of the inverse estimated mixing matrix multiplied by the actual mixing matrix, $\mathbf{J} = \hat{\mathbf{A}}^{-1}\mathbf{A}$.

7.5 Discussion

7.5.1 The performance of the algorithms

Before proceeding to discuss the specifics of the various algorithms' performances, it is important to state the fact that all the algorithms under consideration *do* achieve some degree of separation. This can be clearly seen from figure 7.3, where we have plotted the matrix elements of $\mathbf{J} = \hat{\mathbf{A}}^{-1}\mathbf{A}$ at each iteration of the 'no prior' algorithm for SNR = 5dB. This represents the worst case among our simulations, but even here we see that clearly three elements stand out as converging to a larger value than the rest. This is precisely what we want, and ideally we wish a single element from each column to converge to one and the rest to zero, representing complete separation. For comparison we have also plotted the best case for SNR = 5dB, the IFA algorithm.

The no prior algorithm What is immediately obvious from figure 7.2, is that the basic no prior algorithm consistently has the largest mixing matrix error of all the algorithms under consideration. This should come as no great surprise, since the other scale mixture models include more prior knowledge, namely that of the super-Gaussianity of the sources. Since the sources in our simulations *are* super-Gaussian, naturally this improves

separation. Also, the other scale mixture models implicitly employ an extra level of averaging through computation of the posterior expectations.

The Γ prior algorithm This algorithm actually performs quite well throughout the SNR range we have covered. It does not in all cases, however, do a particularly good job of estimating the source densities. This can be seen from the middle column of figure 7.4, where we have plotted the histogram of the sources along with the estimated source densities. Particularly for source s_2, we see that the estimated pdf misfits by a substantial margin. Referring to table 7.4, we see that this is reflected here, with s_2 scoring lowest in every performance measure. It is probably not coincidental that this algorithm has trouble with estimating s_2. The sources are modelled as the K-distribution (see chapter 5.2), which is particularly well suited for sharply peaked densities. Since s_2 is only slightly super-Gaussian, it would necessitate a rather large value of the shape parameter α to model it correctly. It would appear that our algorithm does not favour this.

The IG prior algorithm The performance of this algorithm is comparable to the Girolami algorithm, but noticeably inferior to the Γ prior algorithm. This is somewhat surprising, since both scale mixture models should be expected to have the same flexibility w.r.t. estimating the pdfs of the sources. Examining Figure 7.4, however, we see that the IG prior algorithm fails to capture the sharp peaks in the histogram for s_1 and s_3. It does do a slightly better job in fitting s_2 than the Γ prior algorithm, but also here the misfitting is substantial. The IG and Γ prior algorithms' failure probably lies in their parameter estimators sensitivity to outliers (for a discussion see e.g. [9]).
The main explanation for the inferior performance of this algorithm probably lies in the NIG distributions inability to represent the most sharply peaked densities.

The Girolami algorithm Girolami's algorithm was shown in [26] to be equivalent to a scale mixture model, where the source distributions are Laplacian. The Laplacian distribution is less flexible than both the K-distribution and the NIG-distribution that describe the sources in the IG and Γ prior algorithms. It is a sharply peaked density, characterised by a single scale parameter. It is therefore not surprising that source s_2 being the source with the least 'peaky' distribution (see figure 7.4), for the most part scores lowest in the performance measures in tables 7.1 through 7.6.

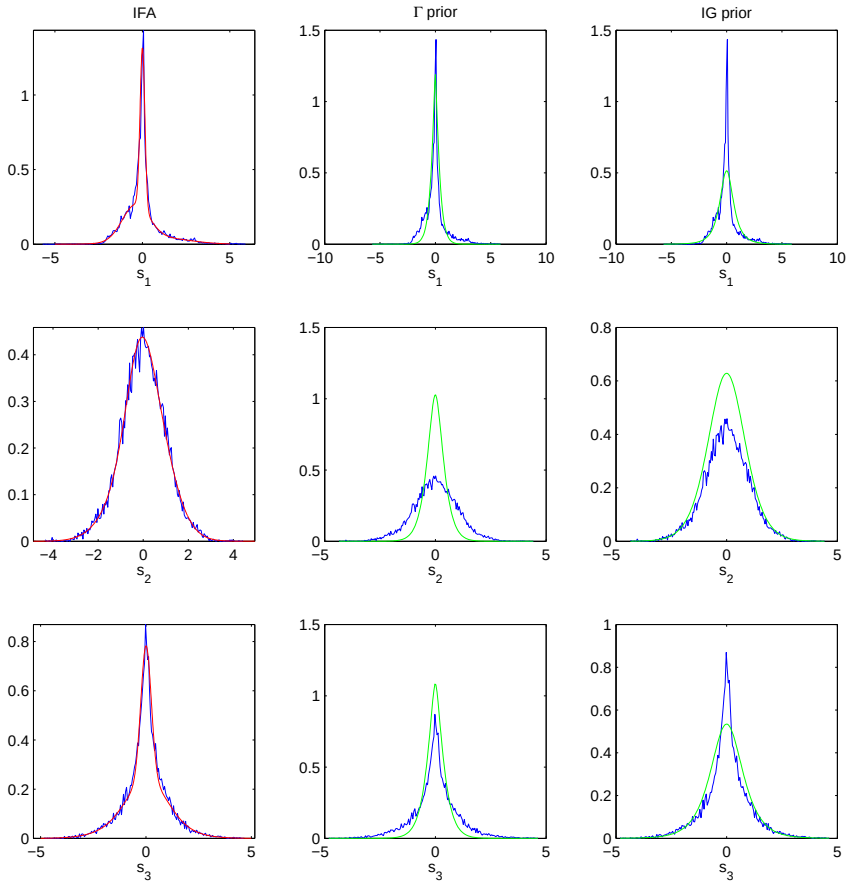


Figure 7.4: The estimated pdf of the sources are plotted together with the histogram of the original sources for SNR=20 dB. The left column shows the results obtained by IFA, the middle column shows the results for the Γ prior algorithm, and the right shows the results for the IG prior algorithm.

Also, its lack of flexibility explains why it does not perform as well as the Γ prior algorithm.

FastICA Slightly surprising, and very depressing, is the fact that FastICA, which has been developed through a noiseless model, outperforms most other algorithms even at the highest noise levels. At the lower noise levels its performance is comparable to the Γ prior algorithm. It would appear that the negentropy estimators are rather robust, and certainly are able to cope with the moderate noise levels we have used in our simulations. The fact that its performance seems to deteriorate slightly for the lower noise levels, is most likely a consequence of how they have defined the stopping criteria in the FastICA package. Allowing the algorithm to run a few more iterations would probably improve its performance compared to the other algorithms.

IFA The IFA algorithm's performance is, by a comfortable margin, superior to all the other algorithms we have considered, particularly at the higher noise levels. The flexibility of the MOG model, allows it to precisely estimate the source densities, as is apparent from from the first column of Figure 7.4. IFA is the only one of the algorithms we have considered that both explicitly takes the noise into account and can handle practically any type of source distribution, even highly skewed and multimodal distributions. A drawback with the algorithm, is that it in the no noise limit will converge to the PCA solution, and does not perform separation. Also, for low noise levels, its convergence speed will be exceedingly slow (There exists a separate IFA algorithm for the no noise limit, but we will not concern ourselves with it here), which is the likely explanation for the slightly deteriorating performance as the noise level increases above 20 dB.

7.5.2 Computational Efficiency

To complete this discussion, a few words on the computational efficiency of the various algorithms is warranted. Computational efficiency has not been an issue when working on this thesis, so this section will only touch on a very few issues and will not go into any great detail.

In a class of its own in terms of speed of convergence, was the FastICA algorithm. For the data sets used here, the results were practically instantly forthcoming. In the other end of the scale falls the IFA algorithm, being by far the slowest to converge. Even with our modest sample size and only

three sources, it took around 13 seconds to run one iteration of the EM algorithm. The scale mixture algorithms, in comparison, all spent approximately 4 seconds on one iteration, the ones requiring the calculation of Bessel functions being slightly slower. This was despite the fact that the IFA algorithm was optimized by implementing the more computationally demanding functions i C, whilst the scale mixture algorithms were rather naive Matlab implementations. The situation for IFA gets exponentially worse with an increasing number of sources. This comes from the way the conditional means are calculated in the E-step of the algorithm, where the summations are done over all the possible configurations of the source states. If each source is modelled as a n-dimensional MOG, then the number of states to be summed over is n^L, L being the number of sources. This is obviously impractical for large L.

7.6 Separation of Gaussians

This section is merely a reiteration of the demonstrations in [28] for the case of linear DSS. Its purpose is to demonstrate that Gaussians can in fact be separated, if one has available knowledge of the temporal structure in form of the covariance matrix.

Combining the steps in Table 6.1 with equation (6.15), we may write

$$\mathbf{w}^+ = \mathbf{X}\mathbf{s}^{+T} = \mathbf{X}\mathbf{D}\mathbf{s}^T = \mathbf{X}\mathbf{D}\mathbf{X}^T\mathbf{w} \tag{7.11}$$

Now, consider the eigenvalue decomposition of $\mathbf{D} = \mathbf{V}\mathbf{\Lambda}_D\mathbf{V}^T$, with $\mathbf{V}$ being the orthonormal matrix consisting of the eigenvectors of $\mathbf{D}$ and $\mathbf{\Lambda}_D$ the diagonal matrix with the corresponding eigenvalues. Writing $\mathbf{\Lambda}_D = \mathbf{\Lambda}_D^{\frac{1}{2}}\mathbf{\Lambda}_D^{\frac{1}{2}T} = \mathbf{\Lambda}^*\mathbf{\Lambda}^{*T}$ we may split the denoising matrix into two parts

$$\mathbf{D} = \mathbf{D}^*\mathbf{D}^{*T} \tag{7.12}$$

with $\mathbf{D}^* = \mathbf{V}\mathbf{\Lambda}^*\mathbf{V}^T$. Defining $\mathbf{Z} = \mathbf{X}\mathbf{D}^*$ we may write equation (7.11) as

$$\mathbf{w}^+ = \mathbf{Z}\mathbf{Z}^T\mathbf{w} \tag{7.13}$$

Seen from this perspective, the update scheme in linear DSS is simply the power method (see e.g. [24]) for finding the dominant eigenvalue of the unnormalised covariance matrix $\mathbf{Z}\mathbf{Z}^T$, and the algorithm converges to the fixed point $\mathbf{w}^*$ satisfying

$$\lambda\mathbf{w}^* = \mathbf{Z}\mathbf{Z}^T/T\,\mathbf{w}^* \tag{7.14}$$

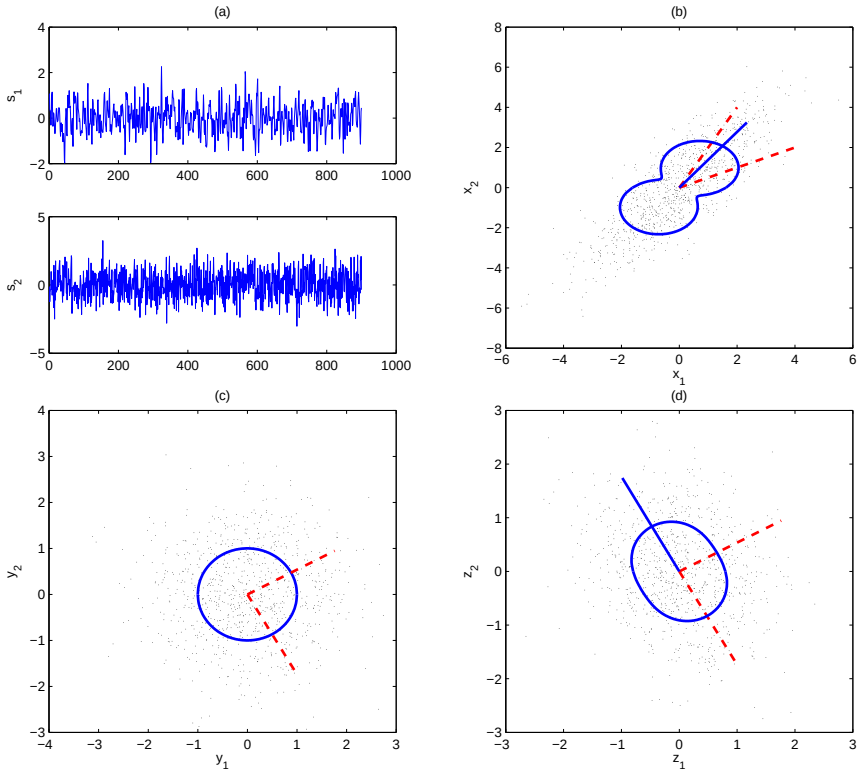


Figure 7.5: Linear DSS. Explanation in text

We thus seek a solution to the problem of separation, by finding the principle eigenvector of some transformed variables.
Now, considering Figure 7.5. The two Gaussian sources s_1 and s_2 depicted in (a), have been mixed linearly to produce the mixtures or observations x_1 and x_2, whose scatter plot is depicted in (b). In all the graphs, the dashed lines depict the mixing vectors, and the solid line depicts the largest eigenvector (the direction of maximum variance). The solid curves denote the standard deviation of the projection of the data in all directions.
The separation is achieved as follows: Firstly the observations are whitened, producing the transformed y_1 and y_2 in (c). Here it is plain to see that the mixing vectors have become orthogonal, a necessary condition for separation within this framework. The signals still cannot be separated by a principal eigenvector, since there is no principal eigenvector in the whitened space. Performing a further transformation by postmultiplication with $\mathbf{D}^*$ produces the variables in (d), where the principal eigenvector has lined up with one of the mixing vectors. Hence one source can be recovered by projecting the transformed variables onto the principal eigenvector. Obviously the second source can be recovered by projecting it on the second eigenvalue (orthogonal to the first), and separation is achieved.

Chapter 8

Summary

8.1 Conclusions

We have in this thesis studied several blind source separation algorithms, and have developed BSS algorithms based on the scale mixture of Gaussians model. The performance of the algorithms has been tested on audio signals that have been mixed artificially.

Among the algorithms we have developed, the scale mixture algorithm with the Γ distributed hyper prior has been shown to do the best job in recovering the sources. We surmise that the reason for this, is the ability of the K distribution to model the sharply peaked distributions that constituted the sources in our experiments. Plotting the histograms of the sources, along with the estimated distributions, seems to confirm this. It also highlights how the model can fail to fit data that is only slightly super-Gaussian. It must be stressed that the algorithms probably can be modified so as to improve their modelling capacity, through better estimators of the scale and width parameters.

We have learned that IFA is a powerful algorithm for source separation when the noise level is high, and it will consistently deliver the best results. When the number of sources increases, however, its computational extravagance limits its usefulness (There does exist a modified version which does not suffer from this defect, but this has not been studied in this thesis).

FastICA performs beautifully for the noise levels we have considered, and its near instant convergence makes it a very appealing algorithm.

The simulations we have done, have been of a somewhat limited nature, and it would not be prudent to infer too much from the results we obtained. We do, however, feel that the Γ prior algorithm shows sufficient promise so as to merit further examination. A more comprehensive simulation scheme should expand on the range of signal-to-noise ratio that the simulations are performed over, and the mixing matrix should also be varied. The effects of filtering the signals for noise removal should also be studied. Finally, each algorithm should be run several times under the same conditions to guard against spurious effects.

8.2 Suggestions to Further work

A natural extension of the scale mixture of Gaussians approach, would be to model each source as a mixture of scale mixtures of Gaussians, in the same manner that Attias has modelled the sources as simple MOGs. This has been done successfully for the noiseless case (see [20]), and it could be interesting to see if the noise covariance matrix could be estimated in any meaningful way.

Bibliography

[1] S. Amari, A. Cichocki, and H. Yang. A New Learning Algorithm for Blind Source Separation. In *Advances in Neural Information Processing Systems, 8*, pages 757–763. MIT Press, Cambridge, 1996.

[2] A. F. Andrews and C. L. Mallows. Scale mixtures of normal distributions. *Journal of the Royal Statistical Society. Series B*, 36(1):99–102, 1974.

[3] H. Attias. A variational Bayesian framework for graphical models. In T. et al Leen, editor, *Advances in Neural Information Processing Systems*. MIT Press, 2000.

[4] Hagai Attias. Independent factor analysis. *Neural Computation*, 11(4):803–851, 1999.

[5] O. E. Barndorff-Nielsen. Normal inverse gaussian distributions and stochastic volatility modelling. *Scandinavian Journal of Statistics*, 24(1):1–13, 1997.

[6] A. J. Bell and T. J. Sejnowski. An Information-Maximization Approach to Blind Separation and Blind Deconvolution. *Neural Computation*, 7:1129–1159, 1995.

[7] J.-F. Cardoso and B. H. Laheld. Equivariant adaptive source separation. *IEEE Transactions on Signal Processing*, 44(12):3017–3030, 1996.

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, B*, 39(1):1–38, 1977.

[9] Anthony P. Doulgeris. Statistical modelling of polarimetric sar data. Master's thesis, University of Tromsø, 2006.

[10] R Gribonval E Vincent and C Févotte. Performance measurement in blind audio source separation. *IEEE Trans. on Audio, Speech and Language Processing*, 14(4):1462–1469, 2006.

[11] T. Eltoft, T. Kim, and T.-W. Lee. Multivariate Scale-Mixture of Gaussians Modelling. In *Proceedings of International Conference on Independent Component Analysis*, pages 799–806, Charleston, SC, USA, 2006.

[12] Z. Ghahramani and M. Beal. Graphical models and variational methods, 2001.

[13] Mark Girolami. A variational method for learning sparse and overcomplete representations. *Neural Comput.*, 13(11):2517–2532, 2001.

[14] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, New Jersey, 2nd edition, 1999.

[15] J. Hérault and B. Ans. Circuits neuronaux à synapses modifiables: décodage de messages composites par apprentissage non supervisé. *C.-R de l'Académie des Sciences*, (299(III-13)):525–528, 1984.

[16] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, New York, 2001.

[17] A. Hyvärinen and E. Oja. A Fast Fixed-Point Algorithm for Independent Component Analysis. *Neural Computation*, 9:1483–1492, 1997.

[18] Shun ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

[19] C. Jutten J. Hérault and B. Ans. Détection de grandeurs primitives dans un message composite par une architecture de calcul neuromimétique en apprentissage non supervisé. In *Actes du Xéme colloque GRETSI*, pages 1017–1022, Nice, France, 1985.

[20] Kenneth Kreutz-Delgado Jason A. Palmer and Scott Makeig. Super-gaussian mixture source model for ica. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Proceedings of the 6th International Symposium on Independent Component Analysis*, pages 1059–1066. MIT Press, Cambridge, MA, 2006.

[21] Richard A. Johnson and Dean W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, 5 edition, 2002.

[22] Te-Won Lee, Mark Girolami, Anthony J. Bell, and Terrence J. Sejnowski. A unifying information-theoretic framework for independent component analysis. *Int. journal of computers and mathematics with applications*, 1999.

[23] A. Mansour, M. Kawamoto, and N. Ohnishi. A survey of performance indexes of ica algorithms. In *Proc. of the 21st IASTED Int. Conf.: Modelling, Identification and Control (MIC'02)*, pages 660–666, 2002.

[24] W. Keith Nicholson. *Linear Algebra With Applications*. International Thomson Publishing, third edition, 1990.

[25] Jason Palmer, David Wipf, Kenneth Kreutz-Delgado, and Bhaskar Rao. Variational em algorithms for non-gaussian latent variable models. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1059–1066. MIT Press, Cambridge, MA, 2006.

[26] Jason A. Palmer. *Variational and Scale Mixture Representations of Non-Gaussian Densities for Estimation in the Bayesian Linear Model: Sparse Coding, Independent Component Analysis, and Minimum Entropy Segmentation*. PhD thesis, University of California, San Diego, 2006.

[27] James Leblanc Phillip. Source separation of speech signals using kurtosis maximization. In *Proceedings of the 35th Allerton Conference on Communication, Control and Computing*, 1997.

[28] J. Särelä and H. Valpola. Denoising source separation. *Journal of Machine Learning Research*, 6:233–272, 2005.

[29] Louis L. Scharf. *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*. Addison Wesley, 1991.