Exploration of multivariate EEG /MEG signals using non-stationary models

Pierre Ablin

université PARIS-SACLAY

informatics mathematics Inria

TELECOM Paris

IP PARIS

NNT : 2019SACLT051

Thèse de doctorat

Exploration of multivariate M/EEG signals using non-stationary models

Thèse de doctorat de l'Université Paris-Saclay
préparée à Inria et Télécom ParisTech

Ecole doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Mathématiques et informatique

Thèse présentée et soutenue à Palaiseau, le 28/11/2019, par

PIERRE ABLIN

Composition du Jury :

Aapo Hyvärinen Professor, University of Helsinki (Department of Computer Science)	Président
Pierre-Antoine Absil Professor, Université Catholique de Louvain (Department of Mathematical Engineering)	Rapporteur
Pierre Comon Directeur de Recherche, CNRS, Université Grenoble Alpes (GIPSA-Lab)	Rapporteur
Émilie Chouzenoux Chargée de Recherche, Inria, Université Paris-Saclay, CentraleSupélec (Centre pour la Vision Numérique)	Examinateur
Jean-François Cardoso Directeur de Recherche, CNRS, Institut d'Astrophysique de Paris	Directeur de thèse
Alexandre Gramfort Directeur de Recherche, Université Paris-Saclay, Inria, CEA	Co-directeur de thèse

Contents

Acknowledgement

Je remercie tout d'abord mes deux directeurs de thèse, qui m'ont excellement encadré. Jean-François, merci de m'avoir enseigné la beauté de l'ICA. Je garde en mémoire nos stimulantes réunions, et j'espère qu'il y en aura encore de nombreuses. Alex, merci d'avoir été le moteur de ma thèse et de toujours m'avoir poussé vers l'avant. Merci aussi de m'avoir envoyé aux quatres coins du monde ! J'ai beaucoup appris à vos côtés. Then, I would like to warmly thank the jury for agreeing to review my work. Your comments have truly improved the quality of this thesis.

Je remercie ensuite mes premiers collègues, ceux de Télécom: Mathurin, pour m'avoir accompagné jusqu'au bout du Rer B (et dans bien d'autres endroits), Pierre, pour m'avoir accompagné jusqu'aux coins les plus reculés du Bangladesh (certes, dans un écran, mais quand même). Merci aux autres occupants du bureau E306, Eugène l'apprenti sorcier, Quentin le grand sportif et Kevin l'alpha. Merci à Mastane, toujours dans le game depuis la Courtine, Hamid, les cheveux les plus soyeux de tout le plateau, Alex, les chaussures les plus blanches de tout le plateau, et Simon, le conducteur fou de Langebaan. Merci aux anciens, Ray Brault, Anna (Dr. Love) et Albert pour avoir ouvert la voie, et aux plus jeunes, Anas et Guillaume.

Merci ensuite à toute l'équipe Parietal pour son accueil, et en particulier à son directeur, Bertrand, pour son oreille attentive. Merci aux deux précurseurs, Elvis et Arthur, merci à Thomas M. pour ton amitié et notre fructueuse collaboration. Merci aussi à Antonia et Hamza pour votre companie dans notre bureau, et à Thomas B., Jache, Hugo, Zacharie, Hugo, Patricio et Jérome pour tous ces verres pris ensemble. Merci à Olivier pour ces discussions, j'espère te croiser encore longtemps aux concerts de la capitale. David, merci pour tous ces bons moments, en particulier cette semaine intense avant la deadline NeurIPS. Big up à la relève, Quentin, Hicham, Valentin, Maeliss et Hubert. Merci à Loubna, Ana-Luisa et Kamalakar pour leur accueil tous les mardi. Merci aussi à tous ceux que j'oublie.

Merci à mes co-auteurs, Francis Bach, Cédric Févotte, Herwig Wendt, Dylan, David, Thomas, Mathurin, Denis et Gaël pour nos collaborations et ces captivants échanges scientifiques.

Je voudrais pour finir remercier mes parents pour leur présence pendant ces ~~25~~ 26 années. Merci à eux d'avoir toujours su me mettre sur les bons rails, pour arriver aujourd'hui à cette thèse. Vous m'avez donné toutes les chances de réussir.

Abstract

Independent Component Analysis (ICA) models a set of signals as linear combinations of independent sources. This analysis method plays a key role in electroencephalography (EEG) and magnetoencephalography (MEG) signal processing. Applied on such signals, it allows to isolate interesting brain sources, locate them, and separate them from artifacts. ICA belongs to the toolbox of many neuroscientists, and is a part of the processing pipeline of many research articles. Yet, the most widely used algorithms date back to the 90's. They are often quite slow, and stick to the standard ICA model, without more advanced features.

The goal of this thesis is to develop practical ICA algorithms to help neuroscientists. We follow two axes. The first one is that of speed. We consider the optimization problems solved by two of the most widely used ICA algorithms by practitioners: Infomax and FastICA. We develop a novel technique based on preconditioning the L-BFGS algorithm with Hessian approximation. The resulting algorithm, Picard, is tailored for real data applications, where the independence assumption is never entirely true. On M/EEG data, it converges faster than the 'historical' implementations.

Another possibility to accelerate ICA is to use incremental methods, which process a few samples at a time instead of the whole dataset. Such methods have gained huge interest in the last years due to their ability to scale well to very large datasets. We propose an incremental algorithm for ICA, with important descent guarantees. As a consequence, the proposed algorithm is simple to use and does not have a critical and hard to tune parameter like a learning rate.

In a second axis, we propose to incorporate noise in the ICA model. Such a model is notoriously hard to fit under the standard non-Gaussian hypothesis of ICA, and would render estimation extremely long. Instead, we rely on a spectral diversity assumption, which leads to a practical algorithm, SMICA. The noise model opens the door to new possibilities, like finer estimation of the sources, and use of ICA as a statistically sound dimension reduction technique. Thorough experiments on M/EEG datasets demonstrate the usefulness of this approach.

All algorithms developed in this thesis are open-sourced and available online. The Picard algorithm is included in the largest M/EEG processing Python library, MNE and Matlab library, EEGlab.

Notation

General

I_p	Identity matrix in $R^{p\times p}$
$M^\top$	Transpose of a real matrix M
$\lvert M\rvert$	The determinant of a matrix M
M^{-1}	The inverse of a matrix M
GL_p	The set of invertible matrices of size $p \times p$
$\mathcal{O}_p$	The set of orthogonal matrices of size $p \times p$; such that $MM^\top = I_p$
$\mathrm{KL}(p, q)$	The Kullback-Leibler divergence between the two distributions p and q, equal to $\int_x \log(\frac{p(x)}{q(x)})p(x)dx$
$\langle x, y\rangle$	The inner product between two elements of an Euclidean space, $x, y \in E$. It is always the canonical inner product, unless specified otherwise. If $E = \mathbb{R}^p$, $\langle x, y\rangle = \sum_{i=1}^p x_i y_i \in \mathbb{R}$
$\langle x, M, y\rangle$	The inner product with respect to M, where M is a symmetric linear operator. $\langle x, M, y\rangle = \langle x, My\rangle$
$x \otimes y$	The outer product between two elements of an Euclidean space, $x, y \in E$. It is a linear map from E to E. If $E = \mathbb{R}^p$, $x \otimes y$ is the $p \times p$ matrix of entries $(x \otimes y)_{ij} = x_i y_j$
$\lVert x\rVert$	The norm of x. If x lives in a Euclidean space, unless specified otherwise, it is the ℓ_2 norm
∇f	The gradient of f
$\nabla^2 f$	The Hessian of f
$\exp(M)$	The exponential of M: $\exp(M) = \sum_{k=0}^{+\infty} \frac{M^k}{k!}$

1 Motivation and contribution

Contents

Background

This thesis gathers three years of work on the problem of Independent Component Analysis (ICA), and its application to brain signal processing. In this section, we review some useful mathematical principles which will guide us throughout this manuscript. The ICA problem is the following. Let $A \in \mathrm{GL}_p$, the set of $p \times p$ invertible matrices, and $\mathbf{s} \in \mathbb{R}^p$ a random vector with independent entries, drawn from a density $d(\mathbf{s}) = \prod_{i=1}^{p} d_i(s_i)$. Assume that we observe the random vector $\mathbf{x} \in \mathbb{R}^p$ such that:

$$\mathbf{x} = A\mathbf{s} \tag{1.1}$$

The matrix A is called the *mixing matrix*, $\mathbf{s}$ the *sources*, and $\mathbf{x}$ the *observations*. The independent component analysis of $\mathbf{x}$ consists in recovering A and $\mathbf{s}$ from samples of $\mathbf{x}$.

Many methods have been proposed in the literature to perform ICA. In this thesis, we follow the *maximum likelihood* principle: we view the ICA model (1.1) as a generative model with parameters A and $\mathbf{s}$. Maximum likelihood offers a principled framework for recovering the parameters of the model, with important statistical guarantees. A brief introduction to maximum-likelihood estimation is made in section 1.1.

Maximum likelihood allows to write the inference of A and $\mathbf{s}$ as an optimization problem. The log-likelihood of ICA writes:

$$\ell(A, \mathbf{s}, \mathbf{x}) = -\log|A| + \sum_{i=1}^{p} \log(d_i(s_i)) \ ,$$

which can be rewritten only in terms of A since we have $\mathbf{s} = A^{-1}\mathbf{x}$:

$$\ell(A, \mathbf{x}) = -\log|A| + \sum_{i=1}^{p} \log(d_i([A^{-1}\mathbf{x}]_i)) \ .$$

First, we consider the case of *non-Gaussian ICA*, where samples $X = [\mathbf{x}_1, \cdots, \mathbf{x}_n] \in \mathbb{R}^{p\times n}$ are assumed to be independent identically distributed. The log-likelihood then writes as a sum over samples:

$$L(A, X) = -n\log|A| + \sum_{j=1}^{n}\sum_{i=1}^{p} \log(d_i([A^{-1}X]_{ij})) \ .$$

Assuming that the source densities d_i are known, the estimation of A amounts to solving the optimization problem:

$$A \in \underset{A\in \mathrm{GL}_p}{\arg\max} -n\log|A| + \sum_{j=1}^{n}\sum_{i=1}^{p} \log(d_i([A^{-1}X]_{ij})) \ . \tag{1.2}$$

This is an optimization problem over the *manifold* GL_p. In section 1.2, we review the main optimization techniques that are going to be useful for our purpose, and in section 1.3 we give a brief introduction to manifolds and to optimization on it. Finally, section 1.4 is devoted to a more in-depth review of the principles of ICA.

The two first contributions of this thesis consist in developing fast algorithms for solving the problem (1.2). The Preconditioned ICA for Real Data (Picard) algorithm leverages the particular structure of the problem to find good and cheap to compute Hessian approximations, and uses them as a preconditioning for the L-BFGS algorithm. In a different approach, the Majorization-Minimization ICA (MMICA) uses the fact that the cost function writes as a sum over samples to obtain fast and safe stochastic algorithms for ICA, extremely fast when the number of observed samples n is large.

The last contribution of this thesis is the application of the Spectral Matching ICA algorithm on M/EEG data. It employs a different route to ICA than the non-Gaussian one: the samples are no longer supposed to be independently distributed. Instead, the sources are colored and show spectral diversity. There is also a noise model, enabling estimation of a different number of sources than observations. This algorithm is fit using the expectation-maximization principle derived in section 1.1.

Readers familiar with these subjects can directly go to section 1.4 for an introduction to ICA. Finally, section 1.5 summarizes and motivates the contributions of this thesis.

1.1 Statistical principles

According to the Cambridge Dictionary, statistics is the *science of using information discovered from collecting, organizing, and studying numbers.* As such, it is a core building of many modern sciences, including neuroscience, machine learning or signal processing. Many methods presented in this dissertation are built on strong statistical ground. We begin with a few statistical principles which will be useful later.

1.1.1 Likelihood, maximum-likelihood estimation

In the following, we consider a set $\mathcal{X}$ (e.g. a vector space), and a sample $x \in \mathcal{X}$ drawn from an unknown distribution over $\mathcal{X}$, ν^*. We are interested in recovering ν^*. In a *parametrized* framework, we consider a *parameter* space Θ which parametrizes a family of distributions $\nu(\cdot;\theta)$ for $\theta \in \Theta$. This can be seen as a model for x: a priori knowledge on x allows to narrow the set of possible distribution to a parametrized family.

Example 1.1 (Gaussian)**.** *For instance, assume $\mathcal{X} = \mathbb{R}$, and so that x is a scalar. We may impose a Gaussian model on x, parametrized by $\theta = (\mu, \sigma) \in \Theta = \mathbb{R} \times \mathbb{R}^+$, by taking:*

$$\nu(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(-\frac{(x-\mu)^2}{2\sigma^2}\Big) \ . \tag{1.3}$$

In this context, we define a model as a parametrized distribution family. We now start with a definition which is going to be extremely important for the rest of this thesis. We say that **the model holds** if there exists $\theta^* \in \Theta$ such that $\nu(\cdot;\theta^*) = \nu^*$.

As an important note, when it comes to real data (and to real generative processes ν^*), no model hold. It is the famous saying attributed to the statistician George Box: "All models are wrong". In example 1.1, the model holds if the distribution ν^* is also Gaussian. We can now define the likelihood and log-likelihood: The likelihood of $x \in \mathcal{X}$ is the function mapping θ to $\nu(x,\theta)$. The log-likelihood is the natural logarithm of this function:

$$\ell(\theta, x) = \log(\nu(x;\theta)) \in \mathbb{R} \ . \tag{1.4}$$

When the sample x is not ambiguous, we simply denote $\ell(\theta)$.

Example 1.2 (Gaussian)**.** *For a Gaussian model 1.3, we have $\ell(\mu,\sigma,x) = -\frac{(x-\mu)^2}{2\sigma^2} - \log(\sigma) - \frac{1}{2}\log(\pi)$.*

Everything said previously is in a setting where only *one* sample, x, is seen. In most cases, a statistician will deal with a *dataset* of $n > 1$ samples, $x_1, \cdots, x_n \in \mathcal{X}$. Luckily, we can treat this set of samples as a single sample $X = \{x_1, \cdots, x_n\}$, coming from the (potentially huge) product space $\mathcal{X}^n = \mathcal{X} \times \cdots \times \mathcal{X}$. The target density ν^* is then from a highly dimensional space. In order to reduce the dimension, it is customary to make some assumptions on the relationship between the samples. Independence and identical distribution are such assumptions. The samples are independent if the distribution factorizes:

$$\nu^*(x_1, \cdots, x_n) = \prod_{i=1}^{n} \nu_i^*(x_i) \ , \tag{1.5}$$

where ν_i^* is the i-th marginal of ν^*. and they are identically distributed if the marignals are all equal: $\nu_i^* = v^*$ for all i and a certain distribution v^* over $\mathcal{X}$. The combination of these two assumptions is called the independently, identically distributed (i.i.d.) assumption.

When using an i.i.d. model, the likelihood factorizes. Denoting $v(\cdot;\theta)$ the i-th marginal of $\nu(\cdot;\theta)$, the log-likelihood of the dataset X writes:

$$L(\theta, X) = \sum_{i=1}^{n} \ell(\theta, x_i) \ ,$$

where $\ell(\theta, x_i) = \log(v(x_i;\theta))$ is the log-likelihood of *one* sample.

Important note: In the following, under the identically distributed sample hypothesis, v will always denote the distribution of a sample (so it is a distribution over $\mathcal{X}$), while ν denotes the distribution of the whole dataset $X = (x_1, \cdots, x_n)$ (so it is a distribution over $\mathcal{X}^n$). In the following, we are mostly interested in recovering the sample distribution v, from which the dataset distribution ν follows under the i.i.d. hypothesis: $\nu(x_1, \cdots, x_n) = v(x_1) \times \cdots \times v(x_n)$.

We now discuss about a powerful technique for parameter estimation: the maximum-likelihood principle. For $X \in \mathcal{X}^n$ a dataset of n observations, and $\nu(\cdot;\theta)$ a model, the maximum-likelihood estimators are:

$$\theta^* \in \arg\max(L(\theta, X)) \ , \tag{1.6}$$

provided that the set of maximum values of the log-likelihood exists.

Example 1.3 (Gaussian)**.** *The log-likelihood for a Gaussian is* $\ell(\mu, \sigma, x) = -\frac{(x-\mu)^2}{2\sigma^2} - \log(\sigma) - \frac{1}{2}\log(\pi)$. *If the standard deviation* σ *is fixed and not a parameter of the model, it is maximized by setting* $\mu = x$. *If not, the maximum-likelihood problem is degenerate, since letting* $\mu = x$ *and* $\sigma \to 0$ *makes* ℓ *go to* $+\infty$.

On the contrary, provided a set of $n > 1$ *i.i.d. samples* $X = (x_1, \cdots, x_n)$, *the log-likelihood writes as a sum:*

$$L(\theta, X) = -\sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2} - n\log(\sigma) - \frac{n}{2}\log\pi \ ,$$

which is maximized for $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$, $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$. *We are here confronted to a sad truth: the maximum-likelihood estimates may be biased. Indeed, assuming that the i.i.d. samples in* X *are draw from Gaussian of mean* μ^* *and standard deviation* σ^*, *we find* $\mathbb{E}[\sigma^2] = \frac{n-1}{n}(\sigma^*)^2$.

We now derive a few properties of the maximum likelihood estimation. First, if $(x_1, \cdots, x_n)$ are i.i.d. samples drawn from a distribution v^*, and v is a model of log-likelihood ℓ, then:

$$\lim_{n\to+\infty} \frac{1}{n}\sum_{i=1}^{n} \ell(\theta, x_i) = \mathbb{E}_{x\sim v^*}[\ell(x,\theta)] \text{ almost everywhere} \quad , \tag{1.7}$$

and $\mathbb{E}_{x\sim v^*}[\ell(x,\theta)] \leq \mathbb{E}_{x\sim v^*}[\log(v^*(x))]$, with equality if and only if $v^* = v(x,\theta)$ almost everywhere. The difference between these two terms is the Kullback-Leibler divergence between v^* and $v(\cdot,\theta)$:

$$\mathrm{KL}(v^*, v) = \mathbb{E}_{x\sim v^*}\left[\log\left(\frac{v^*(x)}{v(x)}\right)\right] \geq 0 \ . \tag{1.8}$$

This proposition shows that (at least asymptotically) the average log-likelihood of the data with respect to the model is bounded by the *entropy* $\mathbb{E}_{x\sim v^*}[\log(v^*(x))]$. This is one justification for the maximum-likelihood approach: assuming that the model holds, with true parameter θ^* (we recall that it means $v^* = v(\cdot;\theta^*)$), we have, as n goes to infinity, $L(\theta) \leq L(\theta^*)$, with equality if and only if $v(\cdot;\theta) = v(\cdot;\theta^*)$.

This is where we first encounter the important notion of *identifiability*. We say that a model $(v(\cdot,\theta))_{\theta\in\Theta}$ is identifiable if:

$$\text{For all } \theta,\theta' \in \Theta,\ \ v(\cdot,\theta) = v(\cdot,\theta') \implies \theta = \theta' \ ,$$

where equality is understood as almost everywhere. Identifiability of a model means that two different parameters of the model cannot lead to the same distribution: there must be a way to tell them apart. This notion plays a key role in the rest of the thesis, and we will get back to it in greater length hereafter. Going back to the justification of the maximum likelihood approach, we find that for models that hold and that are identifiable, asymptotically, $L(\theta) \leq L(\theta^*)$ with equality if and only if $\theta = \theta^*$. Therefore, maximizing the *empirical* (finite sample) log-likelihood should give a reasonable estimate of θ^*. The following results proves that it is indeed the case: the maximum-likelihood estimator is asymptotically consistent: Assume that the model holds for a parameter θ^*, and that it is identifiable. Let $(x_1, \cdots, x_n)$ i.i.d. samples drawn from v^*, and $\theta^n \in \arg\max \sum_{i=1}^n \ell(\theta, x_i)$ a maximum likelihood estimator. Under mild hypothesis, $\theta_n \to \theta^*$ almost surely.

Finally, it is interesting to give a geometrical intuition behind maximum-likelihood estimation. The model $v(\cdot;\theta)$ can be seen as a parametrized "surface" - a *manifold* - in the space of distributions over $\mathcal{X}$. As seen above, finding θ^* amounts to finding the "projection" of ν^* on that manifold, with respected to the Kullback-Liebler divergence: $\theta^* = \arg\min_{\theta\in\Theta} \mathrm{KL}(v^*, v(\cdot;\theta))$. For a dataset $(x_1 \cdots, x_n)$, the maximum likelihood estimator $\hat{\theta}$ is obtained by projecting the empirical distribution $\hat{v} = \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$ on the model manifold. Figure 1.1 illustrates this.

1.1.2 Fisher information and Cramer-Rao bound

In this section, we are focusing on the properties of the expected log-likelihood function (expectation taken under the true distribution): $\mathbb{E}_{x\sim v^*}[\ell(\theta, x)]$. This function is interesting because it is the limit of the empirical log-likelihood.

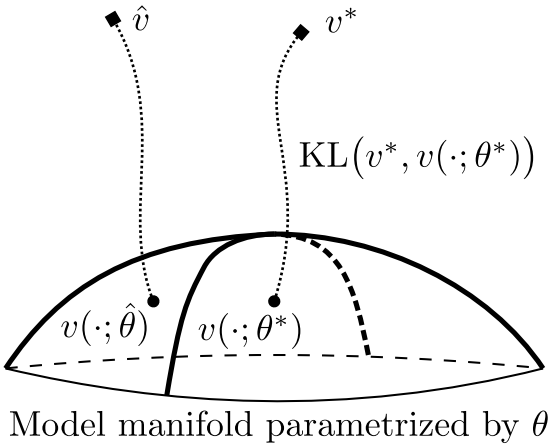


Figure 1.1 – A geometric representation of the topics discussed in this section. A model is a set of distributions $v(\cdot;\theta)$ parametrized by parameters $\theta \in \Theta$. It can be seen as a manifold in the distribution space. The true distribution v^* is a point in the distribution space, which lies on the manifold if the model holds. The model is identifiable if the mapping from Θ to the manifold is injective. The distribution of the maximum likelihood estimator is the projection of v^* on the manifold. The projection is made with respect to the Kullback-Leibler divergence. The lines are not straight to illustrate that this is not an Euclidean setting. Given a finite number of samples $(x_1, \cdots, x_n)$, we do not have access to v^* but rather to $\hat{v} = \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$, its empirical counterpart, and the empirical maximum likelihood estimator $\hat{\theta}$ is once more computed by projection.

We start by introducing the important notion of *Fisher score*. The Fisher score is $\psi(\theta, x) = -\nabla_\theta \ell(\theta, x) = -\frac{1}{v(x;\theta)}\nabla_\Theta v(x;\theta)$. This is the negative gradient of the log-likelihood with respect to the parameters: it measures the steepness of the log-likelihood function at a point in the sample set. For instance, if the model holds for parameter θ^*, the log-likelihood is *flat* at θ^*, on average:

Assume that the model holds at θ^*. Then:

$$\mathbb{E}_{x\sim v^*}[\psi(\theta^*, x)] = 0 \tag{1.9}$$

Indeed, in this case, $\mathbb{E}_{x\sim v^*}[\psi(\theta^*, x)] = -\int_{\mathcal{X}} \frac{1}{v(x;\theta^*)}\nabla_\Theta v(x;\theta^*)v(x;\theta^*)dx = -\nabla_\theta \int_{\mathcal{X}} v(x;\theta^*)dx = -\nabla_\theta 1 = 0$. This is natural: if the model holds, θ^* is a maximum of the expected log-likelihood $\mathbb{E}_{x\sim v^*}[\ell(\theta, x)]$. Eq. 1.9 shows that the gradient cancels at this point.

More generally, even if the model does not hold, maximum-likelihood estimation in the asymptotic regime finds a local maximum $\hat{\theta}$ of the expected log-likelihood, such that $\mathbb{E}_{x\sim v^*}[\psi(\hat{\theta}, x)] = 0$, and the best model in terms of Kullback-Leibler divergence.

We may look further, and study the *curvature* of this function: its Hessian.

The Hessian of the log-likelihood writes:

$$\nabla^2_\theta \ell(\theta, x) = \frac{1}{v(x;\theta)}\nabla^2_\theta v(x;\theta) - \psi(\theta, x) \otimes \psi(\theta, x) \ ,$$

where $\otimes$ is the outer product on the (dual) of Θ. For instance, if $\Theta = \mathbb{R}^p$, $\psi(\theta, x)\otimes\psi(\theta, x)$ is a $p \times p$ matrix of coefficients $\Big(\psi(\theta, x) \otimes \psi(\theta, x))\Big)_{ij} = \psi(\theta, x)_i \psi(\theta, x)_j$.

Therefore, the Hessian of the expected log-likelihood has two terms:

$$\nabla^2 \mathbb{E}_{x\sim v^*}[\ell(\theta, x)] = -\mathbb{E}_{x\sim v^*}[\psi(\theta, x) \otimes \psi(\theta, x)] + \mathbb{E}_{x\sim v^*}[\frac{1}{v(x;\theta)} \nabla^2_\theta v(x;\theta)]$$

The first term is the opposite of the Fisher Information matrix (FIM). The FIM is:

$$\mathcal{I}(\theta) = \mathbb{E}_{x\sim v^*}[\psi(\theta, x) \otimes \psi(\theta, x)] \tag{1.10}$$

In general, the FIM is a positive semi-definite matrix.

In the special case where the model holds, with parameter θ^*, we have $\mathbb{E}_{x\sim v^*}[\frac{1}{v(x;\theta^*)} \nabla^2_\theta v(x;\theta^*)] = \nabla^2 \mathbb{E}_{x\sim v^*}[1] = 0$, therefore:

$$\nabla^2 \mathbb{E}_{x\sim v^*}[\ell(\theta^*, x)] = -\mathcal{I}(\theta^*) \ . \tag{1.11}$$

When the model holds, the FIM gives the opposite of the curvature of the log-likelihood function at the maximum likelihood.

We are now ready to present one of the fundamental results of statistical inference: the Cramer-Rao lower bound.

Proposition 1.4 (Cramer-Rao lower bound)**.** *Assume that $\nu(\cdot, \theta)$ is a model parametrized by $\theta \in \mathbb{R}^p$. Let $T : \mathcal{X} \to \mathbb{R}^p$ a function (T is called an* estimator*). Denote $\mathbb{E}_{x\sim\nu(\cdot;\theta)}[T(x)] = \bar{T}(\theta)$ the exptectation of the estimator, and $J(\theta)$ its Jacobian (a $p \times p$ matrix). Assume that the FIM is never singular. Then:*

$$\mathrm{Cov}_\theta(T) \geq J(\theta)\mathcal{I}(\theta)^{-1} J(\theta)^\top \ , \tag{1.12}$$

where $\geq$ is the symetric matrices inequality, and $\mathrm{Cov}_\theta(T) = \mathbb{E}_{x\sim\nu(\cdot,\theta)}\Big[(T(x) - \bar{T}(\theta)) \otimes (T(x) - \bar{T}(\theta))\Big]$.

The proof is the following:

Proof Consider a vector concatenating $T(x)$ and the Fisher score $\psi(\theta, x)$: $V(\theta, x) = (T(x), \psi(\theta, x)) \in \mathbb{R}^{2p}$.

The covariance of V writes:

$$\mathrm{Cov}_\theta(V(\theta, \cdot)) = \begin{pmatrix} \mathrm{Cov}_\theta(T) & \mathbb{E}_{x\sim\nu(\cdot,\theta)}\Big[(T(x) - \bar{T}(\theta)) \otimes \psi(\theta, x)\Big] \\ \mathbb{E}_{x\sim\nu(\cdot,\theta)}\Big[\psi(\theta, x) \otimes (T(x) - \bar{T}(\theta))\Big] & \mathcal{I}(\theta) \end{pmatrix} \in \mathbb{R}^{2p\times 2p}$$

Let us focus on the upper right corner of the matrix $\mathbb{E}_{x\sim\nu(\cdot,\theta)}\Big[(T(x)-\bar{T}(\theta))\otimes\psi(\theta,x)\Big]$. First, it can be simplified to $\mathbb{E}_{x\sim\nu(\cdot,\theta)}[T(x)\otimes\psi(\theta,x)]$, since $\mathbb{E}_{x\sim\nu(\cdot,\theta)}[\psi(\theta,x)]=0$. Then, we have:

$$\begin{aligned}\mathbb{E}_{x\sim\nu(\cdot,\theta)}[T(x)\otimes\psi(\theta,x)] &= \int_{\mathcal{X}} T(x)\otimes\psi(\theta,x)v(x,\theta)dx &(1.13)\\ &= \int_{\mathcal{X}} T(x)\otimes\nabla_\theta v(x,\theta)dx &(1.14)\\ &= \nabla_\theta\int_{\mathcal{X}} T(x)v(x,\theta)dx &(1.15)\\ &= \nabla_\theta\bar{T}(\theta) &(1.16)\\ &= J(\theta)\ , &(1.17)\end{aligned}$$

so:

$$\mathrm{Cov}_\theta(V(\theta,\cdot)) = \begin{pmatrix}\mathrm{Cov}_\theta(T) & J(\theta)\\ J(\theta)^\top & \mathcal{I}(\theta)\end{pmatrix}\ .$$

What is important here is since it is a covariance matrix, it is a positive definite matrix.

In order to show that $\mathrm{Cov}_\theta(T)\geq J(\theta)\mathcal{I}(\theta)^{-1}J(\theta)^\top$, we finally show that for all $x\in\mathbb{R}^p$, $x\,\mathrm{Cov}_\theta(T)x\geq xJ(\theta)\mathcal{I}(\theta)^{-1}J(\theta)x$. To do so, for a vector $x\in\mathbb{R}^p$, consider the vector $z=(x,-I^{-1}(\theta)J(\theta)^\top x)\in\mathbb{R}^{2p}$.The positivity of $\mathrm{Cov}_\theta(V(\theta,\cdot))$ ensures $z\,\mathrm{Cov}_\theta(V(\theta,\cdot))z\geq 0$, which yields as expected after developing: $x\,\mathrm{Cov}_\theta(T)x - xJ(\theta)\mathcal{I}(\theta)^{-1}J(\theta)^\top x\geq 0$, concluding the proof. ■

In particular, consider an *unbiased* estimator T, such that $\bar{T}(\theta)=\theta$ for all $\theta\in\Theta$. Then, $J(\theta)=I_p$ and we find the simpler unbiased formula:

$$\mathrm{Cov}_\theta(T)\geq\mathcal{I}(\theta)^{-1}\ .$$

This bound is one of the most important results in statistical inference. It says that one cannot get arbitrary good estimators, and quantifies the minimal achievable covariance. Efficient estimators achieve this lower-bound. An estimator $T:\mathcal{X}\to\Theta$ is *efficient* (or *Fisher-efficient*) if it reaches the Cramer-Rao bound: $\mathrm{Cov}_\theta[T(x)]=J(\theta)\mathcal{I}(\theta)^{-1}J(\theta)^\top$. Note that when the model holds for a value θ^*, an estimator may also be called efficient if it reaches the Cramer-Rao bound at θ^*, and not necessarily elsewhere.

Example 1.5 (Gaussian case)**.** *Assume a Gaussian model on the mean: $\theta=\mu\in\mathbb{R}$, and $v(\cdot,\theta)=\mathcal{N}(\mu,\sigma^2)$. The Fisher information for this model - a scalar in this case- is $\mathcal{I}(\theta)=\mathbb{E}_{x\sim v(;,\theta)}[\frac{(x-\mu)^2}{\sigma^4}]=\frac{1}{\sigma^2}$. The Cramer-Rao lower bound is therefore σ^2: any unbiased estimator of the mean will have a variance greater than σ^2. If its variance is equal to σ^2, then it is efficient.*

For instance, the trivial estimator $T(x)=x$ is unbiased and efficient.

Of course, the previous result extends to a dataset $x_1,\cdots,x_n$ under the i.i.d. hypothesis: Assume that $v(\cdot,\theta)$ is a model parametrized by $\theta\in\mathbb{R}^p$. Let $T:\mathcal{X}^n\to\mathbb{R}^p$ an estimator, taking as input a dataset $(x_1,\cdots,x_n)$. Assume that the FIM of v, $I(\theta)$, is never singular. Then:

$$\mathrm{Cov}_\theta(T)\geq\frac{1}{n}J(\theta)I(\theta)^{-1}J(\theta)^\top\ . \tag{1.18}$$

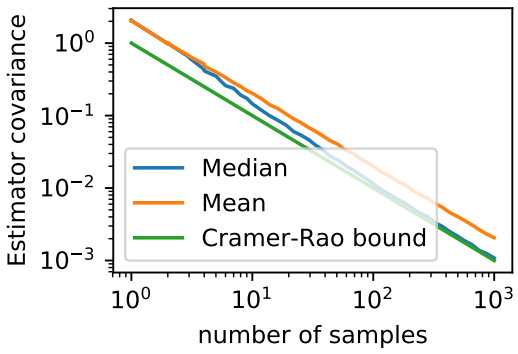


Figure 1.2 – Illustration of the Cramer-Rao bound for the simple example 1.6. The median (blue curve) and mean (orange curve) of datasets of $n = 1, \dots, 10^4$ points following the Laplace law are computed over 1000 repetitions. The variance of these estimators are estimated by averaging these experiments. They all go to 0 as n grows, but only the median estimator reaches asymptotically the Cramer-Rao bound (green curve.)

This inequality simply comes from the fact that $\mathcal{I}(\theta) = nI(\theta)$, where I is the FIM of v and $\mathcal{I}$ is the FIM of the product density $\nu(x_1, \dots, x_n; \theta) = \prod_{i=1}^n v(x_i; \theta)$: the FIM of an i.i.d. dataset of n samples is n times the FIM of a sample.

The final result of this section is the following: the maximum-likelihood estimator is Fisher efficient in the limit $n \to +\infty$: Let $\bar{\theta}_n$ the maximum likelihood estimate of a dataset of n samples $(x_1, \cdots, x_n)$ (this is a random variable since the dataset is random). Then, $J(\theta) \to I_p$ as n goes to infinity and:

$$\lim_{n \to +\infty} n \operatorname{Cov}_\theta(\bar{\theta}_n) = I(\theta)^{-1} \quad , \tag{1.19}$$

where convergence happens in probability.

Example 1.6 (An example of non-efficient maximum-likelihood estimator)**.** *Consider a Laplace model: $\theta \in \mathbb{R}$ with $v(x; \theta) = \frac{1}{2} \exp(-|x - \theta|)$. The maximum likelihood for this model and a set of points $(x_1, \dots x_n)$ is $\hat{\theta}_n \in \arg\min \sum_{i=1}^n |x - \theta|$, i.e. $\hat{\theta}_n$ is the median of the dataset. This estimator is unbiased by symmetry: $\mathbb{E}_{x_1, \cdots, x_n \sim v(x;\theta)}[\theta_n] = \theta$. The Fisher score is given by $\psi(\theta, x) = -\operatorname{sign}(x - \theta)$, and the FIM is $I(\theta) = \mathbb{E}[\psi^2] = 1$: the FIM is constant. Therefore, an efficient estimator must verify $\operatorname{Cov}[T(x)] = \frac{1}{n}$. However, it is simple to show that $\operatorname{Cov}[\hat{\theta}_1] = 2 > 1$ and $\operatorname{Cov}[\hat{\theta}_2] = 1 > \frac{1}{2}$, which means that this estimator is not efficient. To the best of my knowledge, no closed form formula exist for a general variance expression, but simulations suggest that it is always strictly greater than the Cramer-Rao bound.*

Another unbiased estimator for this model is the mean: $\bar{\theta}_n = \frac{x_1 + \cdots + x_n}{n}$. For this estimator, simple computations show $\operatorname{Cov}(\bar{\theta}_n) = \frac{2}{n}$. Therefore, unlike the maximum-likelihood estimator, the estimator is not asymptotically efficient. Figure 1.2 shows simulation illustrating the behavior of these estimators.

In the next section, we describe an important algorithm for maximum-likelihood estimation: the expectation-maximization algorithm.

1.1.3 The Expectation-Maximization algorithm

The expectation-maximization (EM) algorithm [Dempster et al., 1977] is a widely used tool for maximum-likelihood estimation of models with hidden variables. Assume a model on a product space $\mathcal{X} \times \mathcal{Y}$, $\nu(x, y; \theta)$. We call $\mathcal{X}$ the observed space, and $\mathcal{Y}$ the hidden space: we observe samples coming from $\mathcal{X}$, but not from $\mathcal{Y}$. The density of the observed data is given by the marginal distribution of x, $w(x; \theta) = \int_{y \in \mathcal{Y}} \nu(x, y; \theta) dy$, and the log-likelihood is:

$$\ell(\theta, x) = \log(w(x; \theta)) = \log \left(\int_{y \in \mathcal{Y}} \nu(x, y; \theta) dy \right) .$$

Maximizing this likelihood is often problematic, because the integral is usually not tractable. Let us illustrate this with a simple *Gaussian mixture model.*

Example 1.7 (Simple Gaussian mixture model)**.** *Assume that y is drawn following a Bernoulli variable of probability $\frac{1}{2}$. If $y = 1$, then a sample $x \sim \mathcal{N}(\mu, 1)$ is observed. If $y = 0$, then a sample $x \sim \mathcal{N}(\eta, 1)$ is observed.*

The model on (x, y) is a Gaussian mixture, with density:

$$\nu(x, y; \theta) = \frac{1}{\sqrt{2\pi}} \left(y \times \exp(-\frac{(x-\mu)^2}{2}) + (1-y) \times \exp(-\frac{(x-\eta)^2}{2}) \right) . \tag{1.20}$$

The complete log-likelihood therefore simply writes, up to additive and multiplicative constants, $\ell(x, y; \theta) = -y \times (x-\mu)^2 - (1-y) \times (x-\eta)^2$. Owing to the simplicity of this formula, given a complete dataset $((x_1, y_1), \cdots, (x_n, y_n))$, the maximum likelihood estimators are directly:

$$\mu^* = \frac{\sum_{i=1}^n y_i x_i}{\sum_{i=1}^n y_i}, \quad \eta^* = \frac{\sum_{i=1}^n (1-y_i) x_i}{\sum_{i=1}^n (1-y_i)} . \tag{1.21}$$

These complicated equations just state the obvious: the maximum likelihood estimator of μ is the average of all the observed samples x_i drawn following $\mathcal{N}(0, \mu)$, and the same goes for η.

This problem gets interesting when we no longer observe y, but only the incomplete dataset $X = (x_1, \cdots, x_n)$. Marginalizing eq. 1.20 with respect to y yields the incomplete likelihood:

$$w(x; \theta) = \frac{1}{2\sqrt{2\pi}} \left(\exp(-\frac{(x-\mu)^2}{2}) + \exp(-\frac{(x-\eta)^2}{2}) \right) .$$

The log-likelihood of the dataset has a more complicated form than in the complete case. Up to a constant,

$$L(\theta, X) = \sum_{i=1}^n \log \left(\exp(-\frac{(x_i-\mu)^2}{2}) + \exp(-\frac{(x_i-\eta)^2}{2}) \right) .$$

Maximizing this function in closed-form is no longer possible, we therefore resort to an iterative scheme: the EM algorithm.

Let θ a parameter. An EM step starting from θ^0 consists in the two following steps:

- Expectation step: Compute the function $Q_{\theta^0}(\theta, x) = \mathbb{E}[\log(\nu(x, y; \theta))|\theta^0, x]$.
- Maximization step: maximize Q_{θ^0} with respect to θ : $\theta^1 = \arg\max Q_{\theta^0}(\theta, x)$.

Note that the Q function is analytically given by $Q_{\theta^0}(\theta, x) = \int_{y\in\mathcal{Y}} \log \nu(x, y; \theta) p(y|\theta^0, x) dy$, where $p(y|\theta^0, x)$ is the marginal distribution of the unobserved data.

Example 1.8 (EM algorithm for the simple mixture model)**.** *We start for instance by taking $\theta^0 = (0, 1)$. We can then estimate the probabilities that the y_i's are 0 and 1. Following Bayes' rules, we have:*

$$\tau_i^0 = P(y_i = 1|\theta^0, x_i) = \frac{\exp(-\frac{(x_i-\mu^0)^2}{2})}{\exp(-\frac{(x_i-\mu^0)^2}{2}) + \exp(-\frac{(x_i-\eta^0)^2}{2})} \ .$$

This step is called the Expectation *step.*

Then, we replace the y_i's by these values in eq. 1.21, yielding a new vector θ^1. This step is called the Maximization *step. The EM algorithm consists in iterating these two steps.*

Let θ^{EM} a fixed point of the EM procedure. The following derivations show that θ^{EM} cancels the gradient of the log-likelihood L. μ^{EM} is such that:

$$\mu^{EM} = \frac{\sum_{i=1}^n \tau_i^{EM} x_i}{\sum_{i=1}^n \tau_i^{EM}} \ ,$$

where

$$\tau_i^{EM} = \frac{\exp(-\frac{(x_i-\mu^{EM})^2}{2})}{\exp(-\frac{(x_i-\mu^{EM})^2}{2}) + \exp(-\frac{(x_i-\eta^{EM})^2}{2})} \ .$$

The gradient of $L(\theta, X)$ with respect to μ can be written:

$$\frac{\partial}{\partial\mu} L = \sum_{i=1}^n (x_i - \mu) \frac{\exp(-\frac{(x_i-\mu)^2}{2})}{\exp(-\frac{(x_i-\mu)^2}{2}) + \exp(-\frac{(x_i-\eta)^2}{2})} \ ,$$

Therefore :

$$\frac{\partial}{\partial\mu} L = 0 \ \textit{for} \ \mu = \mu_{EM} \ .$$

In addition, we can show that the EM step increases the log-likelihood. Indeed, we have:

$$L(\theta, X) = \sum_{i=1}^n \log\left(\exp(-\frac{(x_i-\mu)^2}{2}) + \exp(-\frac{(x_i-\eta)^2}{2})\right) \tag{1.22}$$

$$= \sum_{i=1}^n \log\left(\tau_i \frac{\exp(-\frac{(x_i-\mu)^2}{2})}{\tau_i} + (1-\tau_i)\frac{\exp(-\frac{(x_i-\eta)^2}{2})}{1-\tau_i}\right) \tag{1.23}$$

$$\geq \underbrace{\sum_{i=1}^n \tau_i \log\left(\frac{\exp(-\frac{(x_i-\mu)^2}{2})}{\tau_i}\right) + (1-\tau_i)\log\left(\frac{\exp(-\frac{(x_i-\eta)^2}{2})}{1-\tau_i}\right)}_{\Lambda(\theta;\tau_i)} \ , \tag{1.24}$$

where the last inequality stems from Jensen's inequality. Note that this holds for any $\tau_i \in]0,1[$. *Define* $\Lambda(\theta;\tau_i)$ *as the last inequality term. It can be written:*

$$\Lambda(\theta;\tau_i) = \underbrace{-\frac{1}{2}\sum_{i=1}^{n}\tau_i(x_i-\mu)^2 + (1-\tau_i)(x_i-\eta)^2}_{Q_{\theta^0}(\theta,x)} \underbrace{-\sum_{i=1}^{n}\tau_i\log(\tau_i) + (1-\tau_i)\log(1-\tau_i)}_{H(\theta^0)} \ .$$

Since Λ *and* Q *are equal up to a constant in* θ, *their maximization in* θ *yield the same result,* θ^1. *Further, the inequality 1.24 is an equality for* $\theta = \theta^0$, *because of the definition of* τ^0. *Finally, we have:*

$$L(\theta^1, X) \geq \Lambda(\theta^1, \tau_i^0) \geq \Lambda(\theta^0, \tau_1^0) = L(\theta^0, X) \ ,$$

We can use the same technique to show that the EM algorithm is non-decreasing in general:

Proposition 1.9 (One EM iteration increases the likelihood)**.** *Let* $\theta^0 \in \Theta$, *and* θ^1 *the output of one EM iteration. Then:*

$$L(\theta^1, X) \geq L(\theta^0, X) \ .$$

Proof For all θ, θ' , Jensen's inequality yields:

$$\begin{aligned}
\ell(\theta, x) &= \log\left(\int_{y\in\mathcal{Y}} \nu(x,y;\theta)dy\right) && (1.25)\\
&= \log\left(\int_{y\in\mathcal{Y}} \frac{\nu(x,y;\theta)}{p(y|\theta',x)}p(y|\theta',x)dy\right) && (1.26)\\
&\geq \int_{y\in\mathcal{Y}} \log\left(\frac{\nu(x,y;\theta)}{p(y|\theta',x)}\right)p(y|\theta',x)dy && (1.27)\\
&\geq Q_{\theta'}(\theta,x) - H(\theta') \ , && (1.28)
\end{aligned}$$

and, noting $H(\theta') = \int_{y\in\mathcal{Y}} \log(p(y|\theta',x))p(y|\theta',x)dy$, we get:

$$\ell(\theta, x) \geq Q_{\theta'}(\theta,x) - H(\theta') \ . \quad (1.29)$$

with equality for $\theta = \theta'$. Therefore, we have $\ell(\theta^0, x) = Q_{\theta^0}(\theta^0, x) - H(\theta^0) \leq Q_{\theta^0}(\theta^1, x) - H(\theta^0) \leq l(\theta^1, x)$. The first inequality comes from the definition of θ^1, and the second comes from eq. 1.29. ■

This shows that the EM algorithm is "safe": one iteration guarantees an increase of the likelihood. Note that the proof and specifically eq. 1.29 allows to view the EM algorithm as a *minimization-majorization* algorithm [Neal and Hinton, 1998]: the E-step essentially consists in computing a function lower-bounding ℓ, which is equal to ℓ at the current iterate, and the M-step consists in maximizing this lower bound. Figure 1.3 illustrates this:

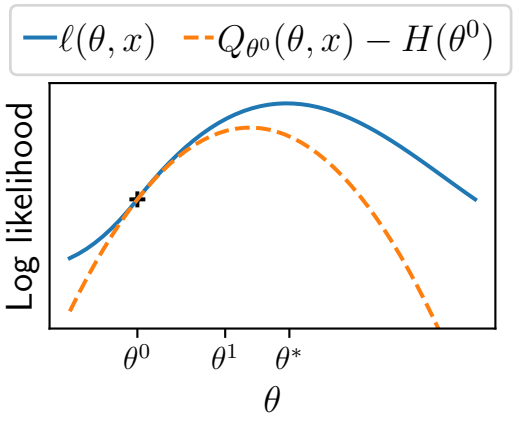


Figure 1.3 – An illustration of the EM algorithm as a minimization-majorization algorithm. We want to maximize the log-likelihood ℓ (blue curve) to find the optimal parameters θ^*. Instead, from a point θ^0, we compute a lower bound of ℓ, exact at θ^0 (orange curve). This lower bound is often simpler to majorize. The EM algorithm takes θ^1 as the arg-maximum of the orange curve.

A little bit more work shows that, under mild conditions, it converges to a stationary point of the log-likelihood: a point which cancels the gradient of the log-likelihood. However, assuming that the model holds with value θ^*, it is not true even on simple cases that the EM algorithm iterates tend to θ^*. Indeed, if the log-likelihood has another local maximum $\theta^\dagger$, the EM algorithm starting close enough from this point will converge to $\theta^\dagger$ instead of θ^*.

1.2 Optimization

The previous section describes the importance of the maximum-likelihood principle: the parameter is computed as

$$\theta^* \in \arg\min_{\theta \in \Theta} -L(\theta) \ .$$

This is an *optimization* problem: a function (the negative log-likelihood) should be minimized. Optimization is a large field of research, extremely important for machine learning. Deep neural networks, support vector machines, linear regression, K-means ... All these widely used machine learning algorithms are *fit* using optimization: their parameters are found as the minimum of a cost function. Note that this is more general than maximum-likelihood estimation: not all cost functions can be written as a negative log-likelihood.

We start by the basics of optimization: gradient descent and stochastic gradient descent.

1.2.1 First order and stochastic methods

Consider a differentiable function $f : \mathbb{R}^p \to \mathbb{R}$. We want to find $x \in \arg\min f$. The gradient of f is the direction of steepest ascent: $\nabla f(x) = (\frac{\partial f}{\partial x_1}, \cdots \frac{\partial f}{\partial x_p})$. In order to solve the optimization problem, gradient descent starts from a point $x^0 \in \mathbb{R}^p$ and iterates moves in the opposite of the gradient direction:

$$x^{t+1} = x^t - \rho_t \nabla f(x^t) \ ,$$

where $\rho_t > 0$ is called learning rate or step size.

This method is fairly intuitive: it follows the direction of steepest descent.

We can already see that under the assumption that this method converges, it finds local extrema (points canceling the gradient): Assume that the sequence of points x^t generated by gradient descent converges to a point $x^* \in \mathbb{R}^p$, and that $\rho_t > \varepsilon$ for some constant $\varepsilon > 0$. Assume that ∇f is continuous. Then:

$$\nabla f(x^*) = 0 \ .$$

This general result does not show that this method reaches a minimum. It is also hard to find conditions for this algorithm to converge. It is therefore interesting to find a framework in which we can give guarantees on the convergence of this method. Such a setting is convex optimization: the function f is assumed *convex*. A non-convex function may have a set of minima with a complicated structure (in particular many isolated points). On the contrary, the set of minima of a convex function has a simple structure: it is a convex set. We say that a function is L-smooth if its gradient is L-Lipschitz:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$

.

This provides the first convergence rate result:

Proposition 1.10 (Rate of convergence of gradient descent for smooth functions)**.** *Let f a convex and L smooth function. Let $x^* \in \arg\min f$ and $f^* = \min f$. Let x^t the sequence produced by gradient descent with a fixed step size $\rho_t = \frac{1}{L}$. Then:*

$$f(x^t) - f^* \leq \frac{\|x^0 - x^*\|^2}{2tL} \ .$$

This result shows that gradient descent converges in value (i.e. $f(x^t) \to f^*$) and gives the rate of convergence. This rate is generally thought to be poor: assuming for simplicity $L = \frac{1}{2}$ and $\|x^0 - x^*\| = 1$, it says that reaching a precision of 10^{-6} (i.e. $f(x^t) - f^* < 10^{-6}$) takes 10^6 iterations.

Under stronger hypothesis, the convergence rate is *linear*. For instance, assuming that f is μ strongly-convex, that is

$$\|\nabla f(x) - \nabla f(y)\| \geq \mu\|x - y\| \ ,$$

we have the following result:

Proposition 1.11. *Let f a convex, L-smooth and μ-strongly convex function. Let $x^* \in \arg\min f$ and $f^* = \min f$. Let x^t the sequence produced by gradient descent with a fixed step size $\rho_t = \frac{1}{L}$. Then:*

$$\|x^t - x^*\|^2 \leq (1 - \frac{\mu}{L})^t \|x^0 - x^*\|^2 \ .$$

This convergence rate is more satisfying, as it states that x^t converges to x^*, and that going from a precision of 10^{-3} to 10^{-6} only doubles the number of iterations.

We now turn to an important alternative to gradient descent when the cost function has a simple sum structure.

Stochastic gradient descent In many machine learning/statistics applications, the function f has a particular structure: it often decomposes as a sum:

$$f(x) = \sum_{i=1}^{n} f_i(x) \ .$$

We have already seen this arise in maximum-likelihood estimation for i.i.d. samples, where the negative log-likelihood is the sum of the individual log-likelihood functions:

$$-L(\theta) = -\sum_{i=1}^{n} \ell(\theta, x_i) \ .$$

Starting from a point $x^0 \in \mathbb{R}^p$, stochastic gradient descent (SGD) iterates [Robbins and Monro, 1951]:

- Pick an index $i \in [1, n]$
- $x^{t+1} = x^t - \rho_t \nabla f_i(x^t)$

There are many ways to pick an index between 1 and n. The two most widely spread are the cyclic order ($i = t \mod n + 1$), and the random order (i is taken uniformly at random). Usually, computing the individual functions f_i and their gradients is much simpler than computing the whole function f. Therefore, the stochastic gradient approach is appealing: one iteration is usually much cheaper than one iteration of gradient descent.

However, there is a stark difference between stochastic gradient descent and gradient descent: assume that we start from a solution $x^0 = x^* \in \arg\min f$. In this case, gradient descent does not move, since $\nabla f(x^*) = 0$: $x^1 = x^*$. On the contrary, if $\nabla f_i(x^*) \neq 0$ (which happens in many scenarii), then SGD will deviate from the solution.

Similarly, SGD is not a proper descent method: the direction $-\nabla f_i(x^t)$ may very well be an *ascent* direction (i.e. $\langle -\nabla f_i(x^t), \nabla f(x^t) \rangle > 0$) rather than a descent direction. Therefore, we may end up with $f(x^{t+1}) > f(x^t)$.

These simple problems hint that the convergence analysis of SGD is more difficult than that of gradient descent. We have for instance the following convergence result for randomized index picking:

Proposition 1.12 (Convergence of stochastic gradient descent)**.** *Assume that the gradients are bounded in expectation: $\mathbb{E}_i[\nabla f_i^2] \leq B$, and that f is μ-strongly convex. Then, the iterates of SGD with fixed step size $\rho_t = \rho$ verify:*

$$\mathbb{E}[\|x^t - x^*\|^2] \leq (1 - \rho\mu)^t \|x^0 - x^*\|^2 + \frac{\rho}{\mu} B \ .$$

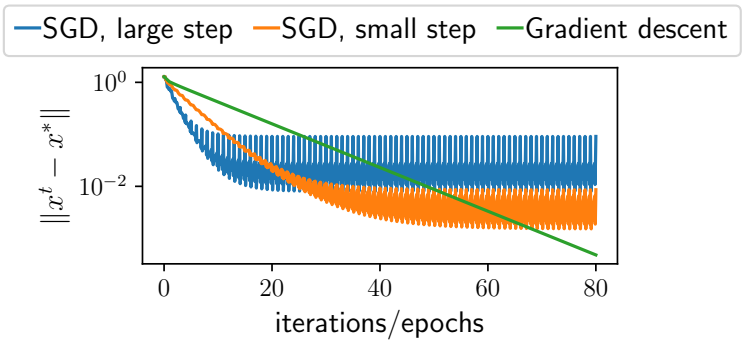


Figure 1.4 – Illustration of the behavior of SGD and gradient descent. An iteration of gradient descent corresponds to an epoch of SGD, i.e. a pass on the whole dataset. The function corresponds to ordinary least squares: $f(x) = \sum_{i=1}^{n}(\langle x, a_i\rangle - b_i)^2$, where $n = 100$ and $x \in \mathbb{R}^4$. Gradient descent is run with the canonical step size $1/L$. We observe its linear rate of convergence. SGD is run with a large step ($\rho = 0.03$) and a smaller step size ($\rho = 0.01$). We observe two phases for SGD: first, a steep convergence, and then a plateau with a lot of variance, where no more progress is made. We see, as justified by proposition 1.12, that a large step size leads to a steeper convergence in the first phase, but a higher plateau.

Although finer analyses exist, this result gives a nice intuition about gradient descent. The bound contains two terms. The first one is the deterministic (gradient descent) bound $(1-\rho\mu)^t\|x^0 - x^*\|^2$. It says that this term is small when $\rho \simeq \frac{1}{\mu}$. The other term is a gradient variance term, which is constant across iterations. It scales linearly with ρ, meaning that we want to take $\rho \to 0$.

This is the usual behavior of fixed step-size SGD: convergence is faster in the first iterations, but eventually reaches a plateau due to the gradient variance. A large step size leads to fast convergence in the beginning, but to a high plateau. A small step size gives slower convergence in the beginning, but a smaller plateau. Figure 1.4 illustrates this on a linear regression toy problem.

Therefore, a natural idea to obtain the best of both worlds (fast convergence in the beginning and low plateau) is to have a sequence of decreasing step-sizes. A classical rule for convex functions is $\rho_t = \frac{\alpha}{\sqrt{t}}$ for non-stongly convex functions, and $\rho_t = \frac{\alpha}{t}$ for strongly convex functions. These step-sizes leads to convergence rates [Nemirovsky and Yudin, 1983] of $O(\frac{1}{\sqrt{t}})$ in the first case and $O(\frac{1}{t})$ in the second. Further, these rates are *minmax optimal*, meaning that there exists a non-strongly convex function f such that for any step size sequence ρ_t, SGD applied on f with that sequence leads to iterates verifying $f(x^t) - f^* \geq \frac{\beta}{\sqrt{t}}$ with $\beta > 0$.

Therefore, it turns out that SGD is in practice a difficult method to use to minimize exactly a function. It is widely used for problems where an approximate minimization is satisfying, like for the overparametrized deep-learning models.

Indeed, Bottou and Bousquet [2008] argue that there are three sources of error in a machine learning model: Let us consider the maximum likelihood framework of the previous section, in the i.i.d. case. Samples $x_1, \dots, x_n \in \mathbb{R}^p$ are observed, drawn i.i.d. from a law v^* that we wish to estimate using a parametrized model $v(\cdot;\theta)$, $\theta \in \Theta$. In the following, for a density v, we denote $E_n[v]$ the negative empirical log-

likelihood: $E_n[v] = -\sum_{i=1}^n \log(v(x_i))$, and $E[v]$ the negative expected log-likelihood: $E[v] = -\int_{x\in\mathbb{R}^p} \log(v(x))v^*(x)dx$.

From the n points, we derive an estimator $\theta^n \in \Theta$ by solving the maximum-likelihood problem:

$$\theta^n \in \arg\min E_n(v(\cdot;\theta)) \ .$$

Consider the expected error that is made:

$$\text{Err} = E[v(\cdot;\theta^n)] - E[v^*] = \text{KL}(v(\cdot;\theta^n), v^*) \ .$$

Importantly, this error is intractable with a finite amount of samples since it involves expectations.

Denote θ^* the *expected* maximum-likelihood estimator: $\theta^* \in \arg\min E[v(\cdot;\theta)]$. The error decomposes as:

$$\text{Err} = \Big(E[v(\cdot;\theta^n)] - E[v(\cdot;\theta^*)]\Big) + \Big(E[v(\cdot;\theta^*)] - E[v^*]\Big) = \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{app}} \ ,$$

where $\mathcal{E}_{\text{est}}$ is the estimation error, i.e. the error made by minimizing the empirical negative log-likelihood rather than by minimizing its expected counterpart, and $\mathcal{E}_{\text{app}}$ is the approximation error, i.e. a measure of the distance between the model $v(\cdot;\theta)$ and the truth v^*.

In a large scale setting, solving the empirical maximum-likelihood problem exactly is sometimes too costly, and instead an *optimization error* is made. A suitable threshold $\varepsilon > 0$ is specified, and we find an approximate minimizer $\hat{\theta}^n \in \Theta$ such that $E_n[v(\cdot;\hat{\theta}^n)] < E_n[v(\cdot;\theta^n)] + \varepsilon$.

Now, the error $\text{Err} = E[v(\cdot;\hat{\theta}^n)] - E[v^*]$ decomposes in three parts:

$$\text{Err} = \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{opt}} \ ,$$

where $\mathcal{E}_{\text{opt}} = E[v(\cdot;\hat{\theta}^n)] - E[v(\cdot;\theta^n)]$. Therefore, it only makes sense to optimize well (bring $\mathcal{E}_{\text{opt}}$ to 0) if the two other quantities are small. In particular, there is an important trade-off between optimization and estimation error: with a finite time budget, should it be allocated to obtaining a finer minimizer (reducing $\mathcal{E}_{\text{opt}}$) or to processing more samples (reducing $\mathcal{E}_{\text{est}}$)? In a large scale settings, SGD is a much better algorithm than gradient descent to minimize the error Err.

1.2.2 Incremental EM and majorization-minimization: stochastic algorithms with descent guarantees

As mentioned previously, a drawback of SGD is that one iteration might very well increase the loss function instead of decreasing it. Therefore, it is of interest to develop stochastic algorithms (algorithms that process one sample at a time) with descent guarantees.

The expectation-maximization framework offers a nice framework to do so [Neal and Hinton, 1998]. We start by showing it on the simple Gaussian mixture model described in Example 1.7.

Example 1.13 (Incremental EM for Gaussian Mixture)**.** *Given a dataset* $(x_1, \cdots, x_n) \in \mathbb{R}^n$*, we want to fit a simple mixture of two Gaussians with adjustable means to it. The negative log-likelihood is a function of* $\theta = (\mu, \eta) \in \mathbb{R}^2$*, and writes, up to a constant:*

$$L(\theta) = -\sum_{i=1}^{n} \log\left(\exp(-\frac{(x_i-\mu)^2}{2}) + \exp(-\frac{(x_i-\eta)^2}{2})\right) \ .$$

The EM algorithm maps $\theta^0 \in \mathbb{R}^2$ *to* $\theta^1 \in \mathbb{R}^2$ *in the following way:*

- *E-step: Compute for all* $i = 1 \cdots n$

$$\tau_i^0 = \frac{\exp(-\frac{(x_i-\mu^0)^2}{2})}{\exp(-\frac{(x_i-\mu^0)^2}{2}) + \exp(-\frac{(x_i-\eta^0)^2}{2})} \ .$$

- *M-step: Set* $\theta^1 = (\mu^1, \eta^1)$ *where* $\mu^1 = \frac{\sum_{i=1}^n \tau_i^0 x_i}{\sum_{i=1}^n \tau_i^0}$ *and* $\eta^1 = \frac{\sum_{i=1}^n (1-\tau_i^0) x_i}{\sum_{i=1}^n (1-\tau_i^0)}$.

We have previously seen that it guarantees a decrease of the cost function L*. Importantly, the M-step is performed with a simple relationship* $\mu^1 = f(\tau_1^0, \cdots, \tau_n^0)$*.*

How can we turn this in a stochastic algorithm? Imagine that we are processing only the j*-th sample* x_j*. We can compute* τ_j^{new} *efficiently (i.e. only knowing the current variable* θ *and the iterate). Now, can we update* μ *? We want to compute* $f(\tau_1^0, \cdots, \tau_j^{new}, \cdots, \tau_n^0)$ *without using all the samples all over again. To do so, we can only store the previous value of* τ_j*,* τ_j^{old}*, and their sum* $S = \sum_{i=1}^n \tau_i$*. A simple computation shows that:*

$$f(\tau_1^0, \cdots, \tau_j^{new}, \cdots, \tau_n^0) = \frac{S \cdot f(\tau_1^0, \cdots, \tau_j^{old}, \cdots, \tau_n^0) + (\tau_j^{new} - \tau_j^{old}) x_j}{S + \tau_j^{new} - \tau_j^{old}} \ . \tag{1.30}$$

Hence, θ *can be updated by only storing* $n+1$ *variables, in constant time: each sample can be processed individually. Overall, the algorithm iterates:*

- *Pick an index* $j \in [1, n]$ *(at random or cyclic).*
- *E-step: Compute* $\tau_j = \frac{\exp(-\frac{(x_j-\mu)^2}{2})}{\exp(-\frac{(x_j-\mu)^2}{2}) + \exp(-\frac{(x_j-\eta)^2}{2})}$.
- *M-step: update* μ *using formula 1.30, and* η *using a similar formula.*

Now, can we actually prove something about this stochastic algorithm ? Recall that the EM inequality writes:

$$L(\theta) \leq \sum_{i=1}^{n} \tau_i \frac{(x_i-\mu)^2}{2} + (1-\tau_i)\frac{(x_i-\eta)^2}{2} + \tau_i \log(\tau_i) + (1-\tau_i)\log(1-\tau_i) \ .$$

Denote $\bar{L}(\theta, \tau_1, \cdots, \tau_n)$ *the right hand term. The E-step of the stochastic algorithm consists exactly in minimizing* $\bar{L}$ *with repect to* τ_j*. Then, the M-step consists in minimizing* $\bar{L}$ *with respect to* θ*. Therefore, we have the usual decrease guarantee for the incremental EM algorithm: each iteration decreases* $\bar{L}$*. Such a guarantee cannot be obtained with*

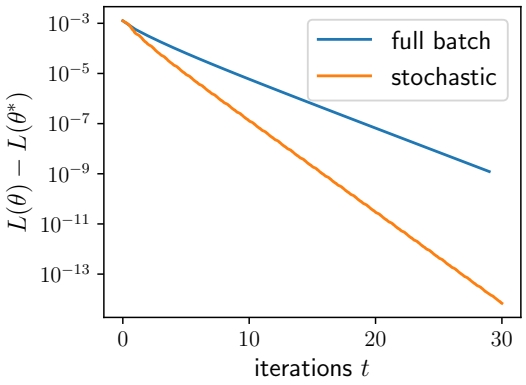


Figure 1.5 – Illustration of the convergence speed of the stochastic EM compared to the usual EM method for the simple Gaussian mixture problem of example 1.13. The stochastic EM is faster: it makes progress whenever a sample is processed, while the usual EM algorithm only updates the parameters once every sample is seen. Further, the stochastic EM method is safe: each iteration guarantees the decrease of a surrogate function.

regular SGD. Note that we however do not have the guarantee that the log-likelihood increases.

Figure 1.5 illustrates the convergence behavior of this method on a toy example, compared to the usual full-batch *EM.*

The Gaussian mixture example generalizes to EM algorithms for i.i.d. data. Indeed, the E-step can be viewed as the estimation of the density of the latent variables y. In [Neal and Hinton, 1998], the author rephrase the EM algorithm as:

- E-step: Estimate the marginal density $p_{\theta^0}(y) = p(x, y|\theta^0, x)$.
- M-step: Set θ^1 to maximize $Q_{\theta^0}(\theta) = \mathbb{E}_{y\sim p}[\log(\nu(x, y; \theta))]$.

For an i.i.d. set of points $(x_1, \cdots, x_n)$ with latent unobserved variables $(y_1, \cdots, y_n)$, it rewrites:

- E-step: Estimate the marginal densities for $i = 1 \cdots n$: $p^i_{\theta^0}(y_i) = p(x_i, y_i|\theta^0, x_i)$.
- M-step: Set θ^1 to maximize $Q_{\theta^0}(\theta) = \sum_{i=1}^n \mathbb{E}_{y_i\sim p^i}[\log(v(x_i, y_i; \theta))]$.

Therefore, the *incremental* (or stochastic) EM algorithm stores all the marginal densities $p^1, \cdots p^n$, and iterates:

- Pick an index $j \in [1, n]$.
- E-step: Estimate the marginal density corresponding to j: $p^j_{\theta^0}(y_i) = p(x_i, y_i|\theta^0, x_i)$, and update it in the stored memory.
- M-step: Set θ^1 to maximize $Q_{\theta^0}(\theta) = \sum_{i=1}^n \mathbb{E}_{y_i\sim p^i}[\log(v(x_i, y_i; \theta))]$.

As such, the M-step looks inefficient since it seems that one has to look at all the samples to update the parameters. However, in many cases (such as the one presented in example above), there is an efficient formula to update θ using only a few stored quantities.

Another related way of obtaining safe stochastic algorithms is to use a majorization-minimization approach [Mairal, 2013]. Assume that the task is to minimize the cost function

$$f(x) = \sum_{i=1}^{n} f^i(x) \ .$$

Assume that for all $x \in \mathbb{R}^p$, we can derive a surrogate function g_x^i such that:

- Majorization: $f^i(x') \leq g_x^i(x')$ whenever x' minimizes g_x^i.
- Matching: the error $h_x = g_x^i - f^i$ is L-smooth, $h_x(x) = 0$ and $\nabla h_x(x) = 0$.

The MISO (Minimization by Incremental Surrogate Optimization) algorithm starts from $x^0 \in \mathbb{R}^p$, and compute a surrogate $g_{x^0}^i$ for each individual function f^i. Then at iteration $t > 0$ it does the following:

- Pick an index $i_t \in [1, n]$.
- Majorization: Compute a surrogate $g_t^{x^{i_t}}$ of f^{i_t} at x^t, and set $g_t^{x^i} = g_{t-1}^{x^i}$ for $i \neq i_t$.
- Minimization: Set $x^t \in \arg\min \sum_{i=1}^n g_{x^t}^i(x)$.

Once again, the efficiency of the algorithm relies on the fact that one can perform efficiently the minimization step, without having to do a pass on the whole dataset. The spirit of this algorithm and of the incremental EM are similar, but these two algorithms differ in some aspects. MISO can be applied to arbitrary sums of functions which do not have to come from a latent variable model. However, some surrogates derived in the EM algorithm do not necessarily verify the smoothness condition required by MISO.

In this thesis, we employ these techniques to perform stochastic ICA with descent guarantees.

1.2.3 Quasi-Newton methods

We now turn to another pillar of optimization: quasi-Newton methods. Most of this introduction can be found in the corresponding chapter of [Nocedal and Wright, 1999].

In this part, we consider the minimization of a function f defined over $\mathbb{R}^p$ which is assumed twice differentiable. Importantly, we do not consider a convex setting: otherwise specified, all results in this part apply in a general non-convex setting. We recall that the Hessian of f at point $x \in \mathbb{R}^p$ is a $p \times p$ matrix of entries $(\nabla^2 f(x))_{ij} = \frac{\partial^2}{\partial x_i \partial x_j} f(x)$. In the following, for short, we denote $H(x) = \nabla^2 f(x)$, and $H^{-1}(x) = \left(\nabla^2 f(x)\right)^{-1}$ its inverse.

First, let us consider the mother of all quasi-Newton methods, Newton's method.

The Taylor expansion of f around a point $x \in \mathbb{R}^p$ writes, at the second order:

$$f(x+\varepsilon) = f(x) + \langle \nabla f(x), \varepsilon \rangle + \frac{1}{2}\langle \varepsilon, H(x), \varepsilon \rangle + o(\|\varepsilon\|^2) \ . \tag{1.31}$$

Assuming that $H(x)$ is positive definite, minimization of the right-hand side of the equation is obtained for $\varepsilon = -H^{-1}(x)\nabla f(x)$. This yields Newton's direction:

Definition 1.14 (Newton's direction)**.** *Newton's direction at a point $x \in \mathbb{R}^p$ is defined as $d(x) = -H^{-1}(x)\nabla f(x) \in \mathbb{R}^p$.*

Newton's method simply follows Newton's direction:

Definition 1.15 (Newton's method)**.** *Newton's method is an iterative algorithm starting from $x^0 \in \mathbb{R}^p$ and writes $x^{t+1} = x^t - H^{-1}(x^t)\nabla f(x^t)$ for $t \geq 0$.*

So far, there are two important result that we can state about this method. First, it converges in one step for a strongly convex quadratic function, which is expected since such a function matches its Taylor expansion at the second order.

Proposition 1.16. *Assume $A \in \mathbb{R}^{p\times p}$ is positive definite, $b \in \mathbb{R}^p$ and $c \in \mathbb{R}$ and that $f(x) = \frac{1}{2}\langle x, A, x\rangle + \langle b, x\rangle + c$. Then, for any $x^0 \in \mathbb{R}^p$, $x^1 \in \mathbb{R}^p$ found after one iteration of Newton's method is such that $x^1 = \arg\min f$.*

Proof Indeed, we have $\nabla f(x^0) = Ax^0 + b$ and $H(x^0) = A$. therefore, Newton's method finds in one step $x^1 = x^0 - A^{-1}(Ax^0 + b) = -A^{-1}b$, i.e the global minimizer of f. ■

Second, this method is invariant with respect to linear transforms of the input space.

Proposition 1.17 (Linear invariance of Newton's method)**.** *Consider $f : \mathbb{R}^p \to \mathbb{R}$, $A \in \mathbb{R}^{p\times p}$ an invertible matrix, and $g : \mathbb{R}^p \to \mathbb{R}$ such that $g(z) = f(Az)$ for any $z \in \mathbb{R}^p$. Consider $x \in \mathbb{R}^p$ and (x^t) the sequence of iterates produces by Newton's method on f startig from $x^0 = x$. Consider $z = A^{-1}x$, and (z^t) the sequence of iterates produced by Newton's method on g. Then, $z^t = A^{-1}x^t$ for all $t \leq 0$.*

Proof The proof is made by induction. The result holds for $t = 0$ by hypothesis. Assume $z^t = A^{-1}x^t$. We have the relationships: $\nabla g(z) = A^\top \nabla f(Az)$ and $\nabla^2 g(z) = A^\top \nabla^2 f(Az)A$. Therefore:

$$z^{t+1} = z^t - (\nabla^2 g(z^t))^{-1}\nabla g(z^t) = A^{-1}x^t - A^{-1}(\nabla^2 f(x^t))^{-1}\nabla f(x^t) = A^{-1}x^{t+1} \ .$$

This concludes the recursion. ■

This is an important property of Newton's method, stating that it is blind to the linear parametrization of the problem. For instance, gradient descent does not possess such property.

Finally, arguably the most important result about Newton's method is its local convergence, at a quadratic rate:

Proposition 1.18 (Convergence of Newton's method)**.** *Theorem 3.7. of [Nocedal and Wright, 1999]: Let $x^* \in \mathbb{R}^p$ a local minimum of f. We assume:*

- *$H(x^*)$ is positive definite.*
- *H is L-smooth in a neighborhood B of x^*: for $x, y \in B$, $\|H(x)-H(y)\| \leq L\|x-y\|$.*

Then, there is a neighborhood of x^*, D, such that we have:

- Convergence*: Newton's method starting from $x \in D$ produces iterates x^t that tend towards x^*.*
- Quadratic convergence*: There is a constant $K > 0$ such that, $\|x^{t+1} - x^*\| \leq K\|x^t - x^*\|^2$.*
- Quadratic convergence of the gradient*: There is a constant $K' > 0$ such that $\|\nabla f(x^{t+1})\| \leq K'\|\nabla f(x^t)\|^2$.*

This is the main result about Newton's method, and explains its importance: a quadratic rate of convergence is very desirable. Indeed, assuming for instance $K = 1$, if at iteration t the error is $\|x^t - x^*\| = 10^{-3}$, it will be of the order $\|x^{t+1} - x^*\| \sim 10^{-6}$ at next iteration.

Still, the previous result is only *local* and tells us nothing about the behavior of Newton's method far from the local minima.

In fact, in its current form, the method is terrible for most functions.

First, in a non-strictly-convex setting, the Hessian might not be positive definite, meaning that the inverse $H^{-1}(x^t)$ might not even be defined, or that we might have $\langle \nabla f(x^t), d(x^t) \rangle > 0$ and therefore Newton's direction might not even be a descent direction.

Second, the method may overshoot: even if the direction is good, maybe the algorithm goes too far. Indeed, consider in $\mathbb{R}$ the function $f(x) = |x|^{4/3}$. Simple derivations show that Newton's method starting from x^0 verify $x^t = -2^t x^0$: Newton's method diverges exponentially fast. Note that this function is not twice differentiable in 0, but it can be smoothed without changing its value outside of $[-x_0, x_0]$ which would lead to the same iterates.

Luckily, both of these pitfalls can be overcome: the first with *regularization*, the second with *line-search techniques*.

Regularization In a non-strictly-convex setting, the Newton's direction $d(x) = -H^{-1}(x)\nabla f(x)$ does not make sense if $H(x)$ is not invertible. It may also be an ascent direction if $H(x)$ is not positive definite (i.e. if x is a saddle point). Regularization is a process to make sure that these unfortunate events do not happen. Most regularizations methods work on the matrix $H(x)$ and transform it in order to find a positive definite matrix, which ensures that the corresponding direction is indeed a descent direction.

The simplest form of regularization consists in adding λI_p to $H(x)$. Indeed, the matrix $H(x) + \lambda I_p$ is positive definite as long as $\lambda > -\lambda_{min}(H(x))$. However, in order to perform such an operation, it is important to have an idea of $\lambda_{min}(H(x))$, which might be extremely costly since $H(x)$ is of size $p \times p$.

Methods involving more structure of H are also possible. If we have access to its eigenvalue decomposition $H(x) = UDU^\top$ where $D = \text{diag}(\lambda_1, \cdots, \lambda_p)$, one could consider the regularization $U\hat{D}U^\top$ where $\hat{D} = \text{diag}(\max(\lambda_1, \delta), \cdots, \max(\lambda_p, \delta))$ with $\delta > 0$ a

small number. This corresponds to clipping the eigenvalues of D. A drawback of this method is that δ should be chosen small in order to keep the regularized matrix close to H, but choosing δ too small leads to very large coefficients in the inverse of the regularized Hessian, and to a direction which might be of extremely large amplitude. An alternative is to simply flip the signs of the negative eigenvalues, but this is a rather heuristic method. Overall, there is no clear consensus about which method is best in practice.

Depending on the structure of the Hessian, other forms of regularization are possible. For instance if the Hessian is block diagonal, each block can be regularized individually. This observation will be used in the following chapters.

Line-search Assume that we are at a point in which Newton's direction is actually a descent direction: $\langle \nabla f(x^t), d(x^t) \rangle < 0$. There is no guarantee that the step $x^t + d(x^t)$ is actually decreasing the function. Line search methods look for a *step size* $\alpha^t > 0$ so that $x^{t+1} = x^t + \alpha^t d(x^t)$ is a suitable point. A first difficulty arises when we want to characterize what suitable means. Of course, we want to impose $f(x^{t+1}) < f(x^t)$, but some stronger conditions are necessary to enforce convergence. Indeed, such a condition is always verified even with an infinitesimal step.

For instance, we might even want to exactly minimize the function in the direction: finding $\alpha^t \in \arg\min_{\alpha > 0} f(x^t + \alpha d(x^t))$.

The following property characterizes these points:

Proposition 1.19. *Let $\alpha^t \in \arg\min_{\alpha>0} f(x^t+\alpha d(x^t))$, and $x^{t+1} = x^t+\alpha^t d(x^t)$. Then, $\langle \nabla f(x^{t+1}), d(x^t) \rangle = 0$.*

Proof Indeed, the derivative of $\alpha \to f(x^t+\alpha d(x^t))$ is given by $\langle \nabla f(x^t+\alpha d(x^t)), d(x^t) \rangle$, therefore at a minimum we indeed have $\langle \nabla f(x^t + \alpha d(x^t)), d(x^t) \rangle = 0$ ■

The most popular definition of "suitable" are the Wolfe conditions, which enforce both sufficient decrease of f and of $\langle \nabla f(x^{t+1}), d(x^t) \rangle$. For two constants $0 < c_1 < c_2 < 1$, they write:

- $f(x^t + \alpha d(x^t)) \leq f(x^t) + c_1 \alpha \langle \nabla f(x^t), d(x^t) \rangle$.
- $|\langle \nabla f(x^t + \alpha d(x^t)), d(x^t) \rangle| \leq -c_2 \langle \nabla f(x^t), d(x^t) \rangle$.

The first condition is a sufficient decrease condition, while the other imposes that α is close enough from a minimizer of f in the search direction. Lemma 3.1. of [Nocedal and Wright, 1999] states that there always exists an interval of value of α verifying these conditions when f is bounded below.

These conditions are important to verify in practice for safety of the algorithm, and in theory because they allow for convergence results. Theorem 3.2. of Nocedal and Wright [1999] writes:

Proposition 1.20 (Zoutendijk's theorem)**.** *Assume f is bounded below, differentiable and L-smooth. Let $x^t \in \mathbb{R}^p$ a sequence defined by recursion:*

$$x^{t+1} = x^t + \alpha^t d^t \ ,$$

where $d^t \in \mathbb{R}^p$ *is a descent direction at* x^t *(i.e.* $\langle \nabla f(x^t), d^t \rangle < 0$*) and* α^t *verifies Wolfe conditions. Denote* $\cos(\theta^t) = \frac{-\langle \nabla f(x^t), d^t \rangle}{\|\nabla f(x^t)\| \|d^t\|}$*. Then:*

$$\sum_{t=1}^{+\infty} \cos^2(\theta^t) \|\nabla f(x^t)\|^2 < +\infty \ .$$

Although more useful in its above form, the last inequality implies that $\cos(\theta^t)\|\nabla f(x^t)\|$ goes to 0.

In particular, assume that the search direction d^t never gets too close to orthogonality with $\nabla f(x^t)$: $\cos(\theta^t) > \gamma$ for all t. Then, Zoutendijk's theorem shows $\nabla f(x^t) \to 0$. This result is usually the strongest one can show in a general non-convex setting.

In practice, deriving an algorithm that finds points verifying Wolfe conditions is tedious. One can refer to section 3.4 of Nocedal and Wright [1999] for practical details. The main tool is interpolation: if a candidate step α is not suitable, a low order polynomial is fit using the function value and its derivative at α, and the next candidate step is the minimum of that polynomial.

Have we fixed Newton's method? Newton's method with line-search and regularization is now a well behaved algorithm, with good convergence guarantees provided by Zoutendijk's theorem, and the impressive quadratic convergence speed. However, this method is almost never used in practice, because it is in most cases too time consuming. Indeed, the Hessian is $p \times p$ matrix, for which each entry should be computed at each iteration. Then, in order to find Newton's direction and evaluate $H^{-1}(x^t)\nabla f(x^t)$, a $p \times p$ linear system should be solved, which is cumbersome.

In contrast, first order methods like gradient descent only need to compute the gradient, an $O(p)$ operation. Quasi-Newton method aim at imitating Newton's direction, while maintaining a linear cost of operations, and therefore only working with gradient evaluations. More specifically, they build an approximation of the Hessian matrix or its inverse, using information about geometry gathered at the previous iterations.

In the following, we focus on Newton's method using at each iteration an approximation of the Hessian, denoted H_t. It writes:

- Set the current direction to $d^t = -H_t^{-1}\nabla f(x^t)$.
- Find a suitable step size α^t and set $x^{t+1} = x^t + \alpha^t d^t$.

In the following, the Hessian inverse matrix is denoted $B_t = H_t^{-1}$. B_t or H_t are updated at each iteration using information gained on the geometry of the cost function. Quasi-Newton's methods are build on the principle that the change in ∇f provides information about $\nabla^2 f$, which is reflected by the first order Taylor expansion of ∇f:

$$\nabla f(x^{t+1}) = \nabla f(x^t) + \nabla^2 f(x^{t+1})(x^{t+1} - x^t) + o(\|x^{t+1} - x^t\|)$$

In the following, denote $y^t = \nabla f(x^{t+1}) - \nabla f(x^t)$ and $s^t = x^{t+1} - x^t$.

Definition 1.21 (Secant condition)**.** *The secant condition writes:* $H_{t+1}s^t = y^t$.

It is desirable that a quasi-Newton algorithms fulfills this natural geometric condition. Further, we want to impose positive definiteness on H_t, in order to guarantee that d^t is a descent direction. In order to keep computations light, most practical quasi-Newton methods start from an initial guess for the Hessian, H_0 (most of the time, $H_0 = \lambda I_p$ is a good choice), and then refine it a each step by doing $H_{t+1} = H_t + \Delta_t$, where Δ_t is a low rank matrix.

For instance, the simplest quasi-Newton method, called SR-1, does rank-1 updates one H_t, under the constraint of verifying the secant condition.

Definition 1.22. *The SR-1 method uses the recursion formula:*

$$H_{t+1} = H_t + \frac{(y^t - H_t s^t) \otimes (y^t - H_t s^t)}{\langle s^t, y^t - H_t s^t \rangle} \ .$$

This is the only rank one update on H_t that satisfies the secant condition.

The Sherman-Morisson formula gives a way to formulate this update in terms of inverse Hessian:

$$B_{t+1} = B_t + \frac{(s^t - B_t y^t) \otimes (s^t - B_t y^t)}{\langle y^t, s^t - B_t y^t \rangle} \ .$$

Sadly, this method is not practical for two reasons: there are no positivity guarantees on H_t, and the denominator in the update may vanish.

The BFGS method [Broyden, 1970, Fletcher, 1970, Goldfarb, 1970, Shanno, 1970] is one of the most widely used. It uses rank 2 updates rather than rank one updates, and this solves the problems SR-1 encounters.

Definition 1.23 (BFGS iteration)**.** *The BFGS method iterates:*

$$H_{t+1} = H_t + \frac{y^t \otimes y^t}{\langle y^t, s^t \rangle} - \frac{(H_t s^t) \otimes (H_t s^t)}{\langle s^t, H_t, s^t \rangle} \ .$$

It corresponds to finding the rank-2 update solution of $\arg\min_B \|B - B_t\|$ *s.t.* $B_t s^t = y^t$ *(for a specific matrix norm).*

In practice, it is more useful to have updates in terms of B_t. They write:

$$B_{t+1} = (I_p - \rho_t s^t \otimes y^t) B_t (I_p - \rho_t y^t \otimes s^t) + \rho_t s^t \otimes s^t, \ \text{with} \ \rho_t = \frac{1}{\langle y^t, s^t \rangle} \ . \tag{1.32}$$

First, a key property of BFGS is that H_t is always positive definite: therefore no regularization is needed. Second, assuming that f is quadratic strongly convex with Hessian A, we have that BFGS applied on f with a perfect line search and any positive definite initial guess for H_0 verifies $H_p = A$: this method recovers in finite time the true curvature of the function. In a general setting, under mild hypothesis, it also provides the desired quadratic convergence.

This method still suffers from a drawback: although less costly than Newton's method, it still requires storing a $p \times p$ matrix, which might be too much in a high dimensional setting. The L-BFGS (limited memory BFGS) [Liu and Nocedal, 1989] solves this

problem. L-BFGS only uses the past m iterations to build the Hessian approximation. It only stores the m previous values of y^t and s^t, which are the only quantities needed to build the Hessian approximation; therefore the memory load is linear in p. The BFGS equation (1.32) is only iterated m times. Denoting $V_t = I_p - \rho_t y^t \otimes s^t$, the BFGS update equation writes $B_{t+1} = V_t^\top B_t V_t + \rho_t s^t \otimes s^t$. Unrolling this equation for m steps yields:

$$\begin{aligned} B_t =& (V_{t-1}^\top \cdots V_{t-m}^\top) B_t^0 (V_{t-m} \cdots V_{t-1}) \\ &+ \rho_{t-m} (V_{t-1}^\top \cdots V_{t-m+1}^\top) s_{t-m} \otimes s_{t-m} (V_{t-m+1} \cdots V_{t-1}) \\ &+ \rho_{t-m+1} (V_{t-1}^\top \cdots V_{t-m+2}^\top) s_{t-m+1} \otimes s_{t-m+1} (V_{t-m+2} \cdots V_{t-1}) \\ &\cdots \\ &+ \rho_{t-1} s_{t-1} \otimes s_{t-1} \ , \end{aligned}$$

where B_t^0 is an initial guess for the Hessian.

This is a mathematical formula never computed in practice: there is an efficient two loop formula to compute the search direction $d^t = -B_t \nabla f(x^t)$, working only with vectors of size p and never computing $p \times p$ matrices. The fact that L-BFGS has a limited memory is usually an advantage, since it means that the outdated previous information about the landscape is forgotten. On the other hand, the Hessian approximation in BFGS contains information about the first iterate. In most cases, L-BFGS converges faster than BFGS. It also allows to change the initial guess for the Hessian at each iteration: in some problems there are much better choices than a multiple of identity. This idea is actually at the core of the Picard solver for ICA, developed in the next chapter. All these facts explain why L-BFGS is usually the go-to quasi-Newton method.

1.3 A bit of Riemannian geometry

In this section, we give a brief and informal introduction to Riemannian manifolds. Most of the content of this section is found in [Absil et al., 2009].

1.3.1 General concepts

We consider as "ambient space" the usual Euclidean space $E = \mathbb{R}^p$ or $\mathbb{R}^{p \times p}$, equipped with the canonical scalar product denoted $\langle \cdot, \cdot \rangle$. A m dimensional (sub-)manifold $\mathcal{M}$ is a subset of E, which locally resembles an Euclidean space of dimension m: for every $x \in \mathcal{M}$, there exists an homeomorphism between an open neighborhood of x and an open set in an m-dimensional Euclidean space. This Euclidean space is called the *tangent space*, and noted T_x, and its scalar product is denoted $\langle \cdot, \cdot \rangle_x$. The associated norm is $\| \cdot \|_x$. Therefore, each point $x \in \mathcal{M}$ is associated to a tangent space and a scalar product.

A Riemannian manifold is a manifold for which the mapping $x \to \langle \cdot, \cdot \rangle_x$ is continuous. This allows to define the length of curves on the manifold: For $x \in \mathcal{M}$, consider $\gamma : [0, 1] \to \mathcal{M}$ a differentiable curve on $\mathcal{M}$. Then for any $t \in [0, 1]$, the tangent vector $\gamma'(t)$ belongs to $T_{\gamma(t)}$. The length of γ is given by: $L(\gamma) = \int_0^1 \|\gamma'(t)\|_{\gamma(t)} dt$. This endows the manifold with a natural notion of distance. The distance between two points in the minimal length of curves linking them:

$$d(x, y) = \inf \{ L(\gamma) \ | \ \gamma(0) = x, \ \ \gamma(1) = y \}$$

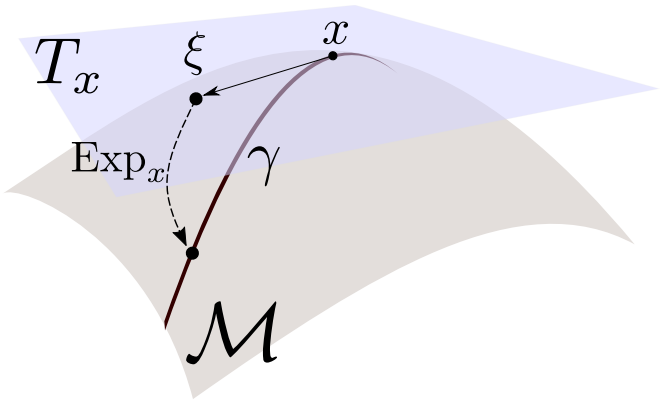


Figure 1.6 – Some useful manifold concepts: a manifold $\mathcal{M}$ is a "curvy" surface. At each point $x \in \mathcal{M}$, the tangent space T_x is an Euclidean space with foot x, where vectors $\xi \in T_x$ live. A geodesic γ is a minimum-length curve on the manifold. It defines a distance on the manifold: the distance between two points is the length of the geodesic linking them. The exponential act like a projection from the tangent space to the manifold, such that $d(x, \mathrm{Exp}_x(\xi)) = \|\xi\|_x + o(\|\xi\|_x)$.

Geodesics are curves realizing the distance: these are minimal length curves.

Note that these geodesics are often hard to compute or impossible to obtain in closed form. The same goes for the geodesic distance: it is the natural notion of distance on $\mathcal{M}$, yet it is sometimes intractable.

The exponential map at x, Exp_x, is a function from T_x to $\mathcal{M}$ (possibly only defined for $\xi \in T_x$ close enough from 0). $\mathrm{Exp}_x(\xi)$ is defined as $\gamma(1)$, where γ is the only geodesic satisfying $\gamma(0) = x$ and $\gamma'(0) = \xi$.

The exponential is smooth, and verifies the important relationship:

$$d(x, \mathrm{Exp}_x(\xi)) = \|\xi\|_x \ ,$$

for ξ of small enough norm. Therefore, it is a mapping from the tangent space to the manifold that preserves distance to the origin of T_x. Just like the geodesic distance, the exponential map is often hard to compute. We often resort to *retractions*, which are "first order" approximations of the exponential map. Let $x \in \mathcal{M}$, $\xi \in T_x$: a retraction at x is a function R_x such that $d(R_x(\xi), \mathrm{Exp}_x(\xi)) = O(\|\xi\|_x^2)$. Importantly, replacing the exponential by a retraction in an optimization algorithm does not really change its convergence results. For instance, Newton's method on manifolds using retraction still converges at a quadratic rate.

Figure 1.6 summarizes the notions seen thus far.

1.3.2 Optimization on manifolds

The previous notions allow to define the gradient of a function: let $f : \mathcal{M} \to \mathbb{R}$. We say that f is differentiable at x, if there exists a tangent vector $\mathrm{grad} f(x) \in T_x$ such that:

$$f(\mathrm{Exp}_x(\xi)) = f(x) + \langle \mathrm{grad} f(x), \xi \rangle_x + o(\|\xi\|_x) \ .$$

Note that the exponential map can be replaced by any retraction without changing the gradient value: if the previous relationship holds, for a retraction R at x, we also have $f(R_x(\xi)) = f(x)+\langle \text{grad}f(x), \xi\rangle_x+o(\|\xi\|_x)$. This allows to define first order optimization algorithms on manifolds. For instance, the gradient descent with step sizes $\rho_t > 0$ starts from $x^0 \in \mathcal{M}$ and iterates:

$$x^{t+1} = R_{x^t}(-\rho_t \text{grad}f(x^t)) \ .$$

Note that if $\mathcal{M}$ is the standard Euclidean space $\mathbb{R}^p$, we can take $R_x(\xi) = x + \xi$, and have $\text{grad}(f) = \nabla f$; the previous equation for gradient descent is thus a generalization of the standard Euclidean gradient descent.

If we want to develop higher order methods, we must define a notion of Hessian: a linear operator from T_x to T_x capturing the curvature of f at x. A simple way of defining it is to consider the function $\phi_x = f \circ \text{Exp}_x : T_x \to \mathbb{R}$, and define $\text{Hess}f(x) = \nabla^2\phi_x(0)$. ∇^2 is defined as the usual Hessian operator in the Euclidean space T_x endowed with its scalar product. Note that in a similar fashion, we have $\text{grad}f(x) = \nabla\phi_x(0)$. With this definition, we have the usual Taylor expansion:

$$f(\text{Exp}_x(\xi)) = f(x) + \langle \text{grad}f(x), \xi\rangle_x + \frac{1}{2}\langle \text{Hess}f(x)[\xi], \xi\rangle_x + o(\|\xi^2\|)$$

Note that here, the choice of retraction matters: using a retraction R in place of Exp would lead to a different Hessian operator.

Quasi-Newton methods are also developed on manifolds, and show the same success as in the Euclidean case. Recall that these methods aim at learning the curvature of the function using the previous evaluations of the gradient. An iteration writes:

- Compute a Hessian approximation H_t, an invertible linear mapping from T_{x^t} to T_{x^t}.
- Find a suitable step-size α, set a direction in the tangent space $\xi^t = -\alpha H_t^{-1}\text{grad}f(x^t)$, and update $x^{t+1} = R_{x^t}(\xi^t)$.

However, a difficulty arises when we want to transpose these algorithms (BFGS for instance) to the manifold framework, since the Euclidean version of these algorithms involves subtractions of quantities that live in different spaces in the manifold setting. For instance, the secant condition in an Euclidean setting writes $\nabla f(x^{t+1}) - \nabla f(x^t) = H_{t+1}(x^{t+1} - x^t)$. In a manifold setting, computing the difference between two points x^{t+1} and x^t does not make sense. Similarly, the difference between the two gradients should be fixed, since they do not live in the same space. The first problem is solved easily. Indeed, a Riemannian quasi-Newton iteration writes $x^{t+1} = R_{x^t}(\xi^t)$, therefore the Riemannian equivalent of $x^{t+1} - x^t$ is ξ^t.

However, H_{t+1} acts on $T_{x^{t+1}}$, not on T_{x^t} where ξ^t lives. Therefore, we would like to find a way to transport points in T_{x^t} to $T_{x^{t+1}}$. This will also be useful to properly subtract two gradients. Given a retraction R, we consider the transport $\mathcal{T}_{\xi_x}(\eta_x) \in T_{R_x(\xi_x)}$, where ξ_x and η_x are in T_x. It transports η_x from T_x to $T_{R_x(\xi_x)}$. Examples of transport are given in the next subsection. The secant condition then rewrites :

$$\text{grad}f(x^{t+1}) - \mathcal{T}_{\xi^t}\text{grad}f(x^t) = H_{t+1}\mathcal{T}_{\xi^t}\xi^t \ ,$$

where everything happens in the tangent space $T_{x^{t+1}}$.

Using this framework, the quasi-Newton algorithms described in the previous section naturally extend to the manifold setting, and show the same converge properties. Many more details are presented in [Absil et al., 2009].

1.3.3 Geometry of the general linear and orthogonal groups

In this manuscript, we are especially interested by two manifolds: the general linear group GL_p of $p \times p$ invertible matrices, and the orthogonal group $\mathcal{O}_p$ of $p \times p$ matrices M such that $MM^\top = I_p$.

The general linear group Since GL_p is an open set of $\mathbb{R}^{p\times p}$, it is a sub-manifold of dimension p^2, and we have $T_M = \mathbb{R}^{p\times p}$ for each $M \in \mathrm{GL}_p$. The tangent spaces can be endowed with the usual Frobenius scalar product $\langle M, M' \rangle = \sum_{i,j=1}^p M_{ij} M'_{ij}$.

However, a more practical scalar product consists in taking the right-invariant scalar product:

$$\langle M, M' \rangle_W = \langle MW^{-1}, M'W^{-1} \rangle \ . \tag{1.33}$$

It verifies the important right-invariance property:

$$\forall U \in \mathrm{GL}_p, \ \ \langle M, M' \rangle_W = \langle MU, M'U \rangle_{WU} \ .$$

We can already see that it acts as a "barrier" to avoid singular matrices. Consider a differentiable curve $\gamma : [0,1] \to \mathbb{R}^{p\times p}$ such that $\gamma(t) \in \mathrm{GL}_p$ for $t \in [0,1[$ and $\gamma(1) \notin \mathrm{GL}_p$. Then, $L(\gamma) = +\infty$. In other words, singular matrices are at an infinite distance from any non-singular matrices using this scalar product.

Possible retractions are given by the trivial $R_W(M) = W+M$, or $R_W(M) = \exp(MW^{-1})W$. Assume that a differentiable function f is defined over $\mathbb{R}^{p\times p}$ with Euclidean gradient ∇f. Then, we have:

$$\begin{aligned} f(R_W(M)) &= f(W+M) \\ &= f(W) + \langle \nabla f(W), M \rangle + o(\|M\|) \\ &= f(W) + \langle \nabla f(W) W^\top W, M \rangle_W + o(\|M\|) \ , \end{aligned}$$

therefore the Riemannian gradient of f, viewed as a function over $\mathcal{M}$, is:

$$\mathrm{grad} f(W) = \nabla f(W) W^\top W \ .$$

Therefore, gradient descent for this scalar product starts from $W^0 \in \mathrm{GL}_p$, and iterates:

$$W^{t+1} = W^t - \rho_t \nabla f(W^t)(W^t)^\top W^t \tag{1.34}$$

$$= \left(I_p - \rho_t \nabla f(W^t)(W^t)^\top \right) W^t \ . \tag{1.35}$$

which is a *multiplicative* update. Interestingly, multiplicative updates are a natural setting to recover this scalar product. Assume that we wish to update W as $(I_p + \mathcal{E})W$, where $\mathcal{E}$ is a small $p \times p$ perturbation. We find:

$$f((I_p + \mathcal{E})W) = f(W) + \langle \nabla f(W), \mathcal{E}W \rangle + o(\|W\|) \tag{1.36}$$

$$= f(W) + \langle \nabla f(W) W^\top, \mathcal{E} \rangle + o(\|W\|) \tag{1.37}$$

The term $\nabla f(W)W^\top$, often called the *relative gradient*, is therefore the steepest ascent direction with respect to a multiplicative update. The resulting gradient descent update on W is $W \leftarrow (I_p - \rho \nabla f(W)W^\top)W$, which coincides with the Riemannian gradient descent.

We will see in the next section that this scalar product plays a key role in Independent Component Analysis.

The orthogonal group The orthogonal group $\mathcal{O}_p = \{W \in \mathbb{R}^{p\times p} |\ WW^\top = I_p\}$ is also going to prove useful in the following chapters. It is a well studied manifold, and a special instance of the Stiefel manifold.

Its dimension is $\frac{p(p-1)}{2}$. The tangent space at a point W is given by $T_W = \{M \in \mathbb{R}^{p\times p} |\ WM^\top + MW^\top = 0\}$, which can be rewritten as $T_W = \{M \in \mathbb{R}^{p\times p} \text{ s.t. } M = ZW \text{ with } Z \in \mathcal{A}_p\} = \mathcal{A}_p W$, where $\mathcal{A}_p$ is the set of skew-symmetric matrices of size $p \times p$ (matrices such that $Z^\top = -Z^\top$). Interestingly, the right-invariant scalar product is constant and equal to the Frobenius scalar product on this manifold:

$$\text{For } W \in \mathcal{O}_p, M, M' \in T_W,\ \ \langle M, M' \rangle_W = \langle M, M' \rangle \ .$$

The exponential map is given by $\text{Exp}_W(M) = \exp(MW^\top)W$, for $M \in T_M$. Computing the matrix exponential is often expensive: when it is a problem, one can instead use the Cayley retraction: $R_W(M) = (I_p + \frac{1}{2}MW^\top)(I_p - \frac{1}{2}MW^\top)^{-1}W$

In order to properly implement second order methods, we should employ a vector transport, to transport vectors $\xi \in T_{W^t}$ to $T_{W^{t+1}}$. Denote $M^t \in T_{W^t}$ the move made at iteration t, that is such that $W^{t+1} = \text{Exp}_{W^t}(M^t)$. The canonical transport is obtained by projection ([Absil et al., 2009], example 8.1.8):

$$\mathcal{T}_{M^t}(\xi) = W^{t+1}\text{skew}((W^{t+1})^\top \xi) \in T_{W^{t+1}} \ ,$$

where the skew of a matrix M is $\frac{1}{2}(M - M^\top)$.

1.4 Independent Component Analysis

In this section, we give an introduction to the topic of Independent Component analysis, with help from the three fields discussed in the previous section (information geometry, optimization and manifolds). It is inspired by the two reference books on the topic: [Hyvärinen and Oja, 2000, Comon and Jutten, 2010].

Assume that we receive samples $x \in \mathbb{R}^p$. A general problem in machine learning and statistics is to extract structure from x, by finding a latent representation. ICA does exactly that, by assuming that x is a linear combination of sources $s \in \mathbb{R}^p$:

$$x = As \ ,$$

where $A \in \mathrm{GL}_p$ is the mixing matrix, and $s \in \mathbb{R}^p$ are the independent sources. The goal of ICA is to recover A and s from observations of x. ICA is a special case of *matrix factorization problem*: given a dataset $X = \{x_1, \dots, x_n\} \in \mathbb{R}^{p \times n}$ of n samples, ICA consists in *factorizing* the matrix X as :

$$X = AS \ ,$$

where $S \in \mathbb{R}^{p \times n}$ is the source matrix. One of its key feature is that there is no approximation made: we do not have $X \simeq AS$, we have $X = AS$.

The perfect data-fit in ICA has an important consequence: given enough samples, it is enough to only estimate either the sources or the mixing matrix to determine both.

Proposition 1.24. *Assume $X \in \mathbb{R}^{p \times n}$ where $n \geq p$. Assume rank$(X) = p$, and that $X = AS$ with $A \in \mathrm{GL}_p$ and $S \in \mathbb{R}^{p \times n}$. Then:*

$$S = A^{-1}X \ \text{ and } \ A = XS^{\dagger} \ ,$$

where $S^{\dagger}$ is the Moore-Penrose pseudo inverse of S.

The proof follows from the model equation $X = AS$. Therefore, the ICA model is completely identified if we estimate either A or S. Since A contains p^2 parameters, usually much less than S, ICA algorithms focus on estimating A. Estimation is done by finding a matrix $W \in \mathrm{GL}_p$ such that the vector $y = Wx$ contains signals that are as independent as possible, hereby estimating the inverse of A.

We want to emphasize that this perfect data fit is rather uncommon in matrix factorization techniques. Indeed, matrix factorization allow the factorization and the data to differ, and usually trade *data fit* for another property. For instance, a rank k representation of X can be obtained by finding $U \in \mathbb{R}^{p \times k}$ and $V \in \mathbb{R}^{k \times n}$ such that $X \simeq UV$; usually by minimizing $\|X - UV\|_2$. Additional terms and constraints can be employed. Non-negative matrix factorization [Paatero and Tapper, 1994, Lee and Seung, 1999] forces the entries of U and V to be positive, which might be a good prior in some contexts. Sparse dictionary learning [Olshausen and Field, 1996, Aharon et al., 2006] looks for a sparse representation, enforced by a ℓ_1 constraint: U, V are found by minimizing $\|X - UV\|_2 + \lambda \sum_{i=1}^{k} \|U_i\|_1$.

This introduction to ICA is organized as follows. In section 1.4.1, we discuss the problem of identifiability of ICA: can we really infer A from realizations of x? Then, we derive the ICA non-Gaussian likelihood and discuss the key role of the density prior on the sources in Section 1.4.2. We give some insights on ways of measuring independence and link it

with maximum likelihood estimation in Section 1.4.3. In section 1.4.4, we discuss the important notion of equivariance, and show that the geometry of the problem enables estimators with performance independent from the mixing matrix A. In Section 1.4.5, we describe one of the most widely used ICA algorithm: Infomax. In Section 1.4.6, we investigate ICA algorithms working under the constraint of decorrelation, and derive the FastICA algorithm. In Section 1.4.7, we show that one can exploit other statistical properties of the sources than non-Gaussianity to perform ICA, like non-stationarity or time correlation. Section 1.4.8 is devoted to the extension of the standard ICA model to ICA models with noise. Finally, Section 1.4.9 illustrates the use of ICA in M/EEG data processing.

There are several aspects of ICA that we do not review here, because they are less useful for the purpose of this thesis. For instance, we do not discuss the separation of complex signals, cumulant based ICA [Cardoso and Souloumiac, 1993], the original approach of Jutten and Herault [1991] or kernel ICA [Bach and Jordan, 2002].

The first question we should ask is whether ICA is *identifiable*: is there any hope to recover A from realizations of x?

1.4.1 Indeterminacies and identifiability

Before wanting to estimate the parameters of the ICA model, we should wonder whether if it is an identifiable model, or more generally, what are the indeterminacies of the model: Assuming that $x = As = A's'$, where $A, A' \in \mathrm{GL}_p$ and s, s' are vectors of independent entries, we want to find links between A and A'. Using the invertibility of A, denoting $C = A'^{-1}A$, the indeterminacy question becomes:

> Let s a vector of independent components of size p and $C \in \mathrm{GL}_p$. Under which condition does the vector $s' = Cs$ has independent entries?

First, we can see that there is a *scale* indeterminacy: indeed, C can be a diagonal matrix, since if $s_1, \dots, s_p$ are independent variables, so are $s' = (\alpha_1 s_1, \dots, \alpha_p s_p)$ for $\alpha_1, \dots, \alpha_p \in \mathbb{R}$. Likewise, there is a permutation ambiguity: if σ is a permutation of $\{1, \dots, p\}$, $s_{\sigma(1)}, \dots, s_{\sigma(p)}$ is also a vector of independent entries. Therefore, C can be any *scale-permutation matrix*:

Definition 1.25 (Scale-permutation matrix)**.** *A scale permutation matrix of size p is a matrix $C \in \mathrm{GL}_p$, for which there exists a permutation σ of $\{1, \dots, p\}$ and non-zero scalars $\alpha_1, \dots, \alpha_p$ such that:*

$$M_{ij} = \alpha_i \delta_{i,\sigma(i)} \ .$$

The scale-permutation ambiguity is intrinsic to any ICA problem, regardless of the marginal densities of s. Are there other indeterminacies?

Independence is a sufficient condition for decorrelation: if s has independent entries, then $\mathbb{E}[s_i s_j] = 0$ for $i \neq j$. This gives a necessary condition on C:

Proposition 1.26. *Assume that s has independent entries and that $s' = Cs$ also has independent entries. Then, $Cov(s') = CC^\top$ is a diagonal matrix, i.e. $C = \Lambda U$ with Λ a diagonal matrix and $U \in \mathcal{O}_p$.*

In the special case where s is assumed to be Gaussian, this condition is also sufficient:

Proposition 1.27 (Indeterminacies for Gaussian white noise)**.** *Assume s contains independent Gaussian entries. Then Cs also has independent entries if and only if $CC^\top$ is diagonal.*

Indeed, for Gaussian sources, independence is exactly equivalent to the second order condition of decorrelation. Gaussian sources are actually the only sources with such property. In [Comon, 1994], the indeterminacy of ICA is proved to be only scale and permutation if there is at most one Gaussian in the mix.

Theorem 1.28 (Identifiability of ICA, thm. 11 of [Comon, 1994], based on [Darmois, 1953])**.** *Assume that s contains independent entries, of which at most one is Gaussian. Let $C \in \mathrm{GL}_p$, such that $s' = Cs$ also has independent entries. Then, C is a scale-permutation matrix.*

Therefore, we say that we have solved the ICA problem if we have found a matrix A' which is equal to CA where C is a scale-permutation matrix, and we say that the ICA problem is not identifiable if there are more than one Gaussian in the mix. Importantly, this theorem implicitly makes an i.i.d. assumption on the data, because source samples s are identified to their density. ICA under the i.i.d. hypothesis is often called *non-Gaussian* ICA because identifiability requires that sources are non-Gaussian, except maybe one. Other frameworks than the i.i.d. setting exist, and we will cover them as well as their identifiability in due time in section 1.4.7.

1.4.2 Maximum-likelihood ICA and the role of density

In this section, we derive the non-Gaussian likelihood for ICA. Half of this thesis is devoted to the minimization of the negative log-likelihood.

Consider sources $s \in \mathbb{R}^p$ drawn from a density d^*, and denote d_i^* the marginal density of d^*. The independence assumption on the sources writes: $d^*(s) = \prod_{i=1}^p d_i^*(s_i)$. In practical cases, d^* is unknown, and it can be thought of as a parameter of the model: we denote d the model density on the source, which also factorizes as $d(s) = \prod_{i=1}^p d_i(s_i)$. The likelihood of x writes, by change of variable:

$$d_x(x) = \frac{1}{|A|} d(A^{-1}x)$$

Therefore, the negative log-likelihood of ICA with model d for a sample $x \in \mathbb{R}^p$ writes:

$$\ell(W, d, x) = -\log|W| - \sum_{i=1}^p \log(d_i([Wx]_i)) \ ,$$

where $W = A^{-1}$ is the unmixing matrix. For a dataset $X \in \mathbb{R}^{p \times n}$, the average negative log-likelihood writes:

$$\mathcal{L}(W, d, X) = -\log|W| - \frac{1}{n} \sum_{i=1}^p \sum_{j=1}^n \log(d_i([WX]_{ij})) \tag{1.38}$$

$$= -\log|W| - \sum_{i=1}^p \hat{E}[\log(d_i([WX]_i))] \ , \tag{1.39}$$

where $\hat{E}$ is the averaging operation. For simplicity, we will omit the 'X' (resp. 'd') in the argument of $\mathcal{L}$ when the dataset (resp. density model) is not ambiguous.

It is important to understand how the likelihood depends on model density d and the true density d^*. First, let us conduct a toy experiment. We take $p = 2$ and $n = 100$, and generate sources coming from the following density: $d_i^*(s_i) \propto \exp(-|s_i|^3)$. This density is *sub-Gaussian*: its tail decays faster than an exponential, and most of its mass in concentrated around 0. We mix these signals with a random 2×2 matrix, A. We compare three models for d:

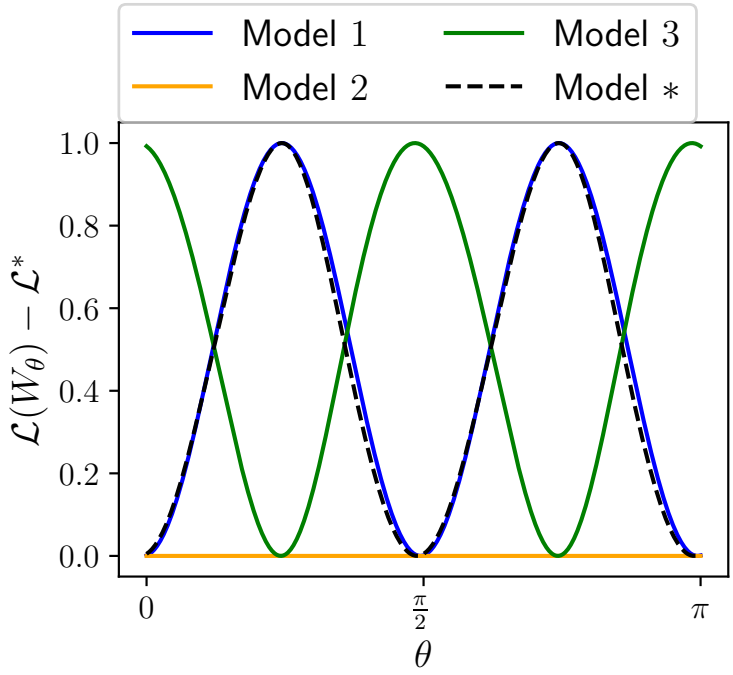
- Model 1: a different sub-Gaussian density, with fast decaying tails. $d_i \propto \exp(-|x|^4)$.
- Model 2: a Gaussian model. $d_i \propto \exp(-|x|^2)$.
- Model 3: a super-Gaussian model, with heavy tails. $d_i \propto \exp(-|x|)$.
- Model $*$: the true model $d = d^*$.

Then, we evaluate $\mathcal{L}(W_\theta A^{-1}, d)$, where W_θ is a rotation matrix: $W_\theta = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$. Choosing $\theta = 0$ leads to perfect separation and recovers exactly the original sources. Fig. 1.7 displays the results, as well as the profile of the four densities. Some comments are in order. First, the Gaussian model cannot discriminate. This is expected because in this case, $\mathcal{L}(W, d)$ is rotation invariant. Second, model 1 and model $*$ perform about as well. They do not recover exactly $W_\theta = I_p$ as minimum because there is a finite number of samples, but they still manage to correctly separate the mixture: in this case, model mismatch does not impair source recovery. Finally, model 3 completely fails: it recovers an unmixing matrix with $\theta = \pi/4$, corresponding to maximally mixing the sources (it takes $\hat{s_1} = \frac{\sqrt{2}}{2}(s_1 + s_2)$ and $\hat{s_2} = \frac{\sqrt{2}}{2}(s_1 - s_2)$). Another interesting fact is that all models have a stationary point around $\theta = 0$: $\frac{d}{d\theta}\Big|_{\theta=0} \mathcal{L}(W_\theta A^{-1}, d) = 0$. Some of these results are general, and formalized in the next parts. The discussion about super and sub-Gaussian density is hand-wavy, and no precise definition has been given. We will see in Section 1.4.5 a precise criterion based on d and d^* for A^{-1} to be an optimum of $\mathcal{L}$. Unfortunately, formalizing the notion of super- and sub-Gaussian density that fits well the ICA problem is hard. It would look like this:

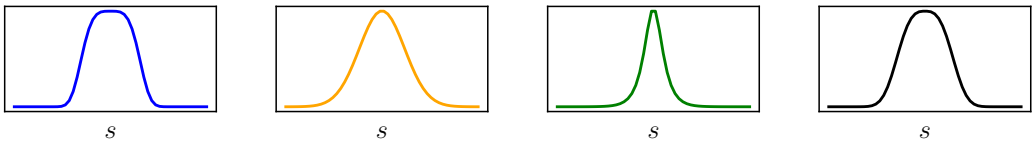
Definition 1.29 (Sub and Super Gaussian splitting)**.** *Let D the space of all distributions over $\mathbb{R}$. A splitting of D is a set of super-Gaussian densities, $S^+ \subset D$, and a set of sub-Gaussian densities $S^- \subset D$, such that:*

- *$D = S^+ \cup S^- \cup G$, where $G \subset D$ is the set of Gaussian densities and the union is disjoint.*
- *For sources $s \in \mathbb{R}^p$ with marginals $d_i^* \in S^+$, I_p is a local minimum of $\mathcal{L}(W, s, d)$ if and only if for all i, $d_i \in S^+$.*
- *For sources $s \in \mathbb{R}^p$ with marginals $d_i^* \in S^-$, I_p is a local minimum of $\mathcal{L}(W, s, d)$ if and only if for all i, $d_i \in S^-$.*

Unfortunately, such a splitting is hard to derive; and we conjecture that it does not exist.



(a) Comparison of the negative log-likelihoods for different source models d. The negative log-likelihoods are normalized to fit in the interval $[0, 1]$.



(b) Model 1 (c) Model 2 (d) Model 3 (e) Model $*$

Figure 1.7 – A toy example to investigate the effect of the density model on ICA estimation. Model 1 and model $*$ find minima for $\theta \simeq 0, \pi/2, \pi$, corresponding to correct separation. Model 2 is not discriminating. Choosing model 3 leads to $\theta \simeq \pi/4$, which corresponds to maximally mixing the components.

Another important note is that the choice of density fixes the scale indeterminacy. Indeed, it is simple to give an analysis of the scale of the mixture. Consider a rescaling move consisting in multiplying the i-th row of W by a factor λ, which we denote $T_{i,\lambda}(W)$. Define $f(\lambda) = \mathcal{L}(T_{i,\lambda}(W), d)$. Up to a constant, it holds:

$$f(\lambda) = -\log(\lambda) - \hat{E}[\log(d_i(\lambda y))] \ ,$$

where $y = [WX]_i$ is the i-th estimated source. In order to differentiate this expression, we introduce the score functions of d.

Definition 1.30 (Score functions)**.** *For a density model d, the i-th score function is* $\psi_i(y) = -\frac{d}{dy}\log(d_i(y))$.

In general, we assume ψ_i to be increasing and odd. We find that the minimum of f is reached at a point for which $\hat{E}[\lambda y \psi_i(\lambda y))] = 1$. The derivative of the left hand term is $\hat{E}[\psi_i(\lambda y))y + \lambda \psi_i'(\lambda y))y^2]$ which is positive if ψ_i is an increasing odd function: f only has one minimum, which is λ fixing the scale of the sources. Therefore, only a permutation ambiguity remains.

Some ICA algorithms attempt to estimate the unmixing matrix and the source density at the same time. Most of the time, a parametric model for d is used. The most rudimentary choice is a binary choice : a switch between a super and a sub-Gaussian density. For instance, [Jung et al., 1997] uses:

$$\psi(s) = s \pm \tanh(s).$$

The simplicity of this model makes it simple to perform density estimation. It is also possible to employ a wide range of density estimation technique, like fitting a Gaussian mixture model using the EM algorithm, but this leads to an important overhead to the ICA estimation.

We now describe how the maximum likelihood estimation relates to measures of independence.

1.4.3 Measures of independence

Given a dataset $X \in \mathbb{R}^{p \times n}$, the goal of ICA is to determine a matrix $W \in \mathrm{GL}_p$ such that the rows of the matrix $Y = WX$ are as independent as possible. Therefore, assuming that an independence criterion $\mathcal{I}$ is available, ICA boils down to solving:

$$W \in \underset{W \in \mathrm{GL}_p}{\arg\min} \{\mathcal{I}(Y) \mid Y = WX\} \ .$$

Maximum-likelihood offers a tractable way to quantify independence. Importantly, the negative log-likelihood fits in our framework of minimizing a criterion of independence of $Y = WX$, because it can be expressed only in terms of Y. Indeed, up to a constant:

$$\mathcal{L}(W, d) = -\frac{1}{2} \log |\frac{YY^\top}{n}| - \sum_{i=1}^{p} \hat{E}[\log(d_i(Y_i))] \ .$$

Next, we try to explain why it does measure independence of the rows of Y, and relate it to other measures of independence. As usual in maximum likelihood estimation, it is also understood (up to a constant) as the Kullback-Leibler divergence between the true distribution and the model. In the limit $n \to +\infty$:

$$\mathcal{L}(W, d) = \mathrm{KL}(d_x, d_{As}) + \mathrm{cst} \ ,$$

Where d_x is the true density of the samples, and d_{As} is the density of the vector As where $s \sim d$. Using invariance of the KL divergence, denoting $y = Wx$ and d_y the distribution of y, we find [Cardoso, 1998b]:

$$\mathcal{L}(W, q) = \mathrm{KL}(d_y, d) + \mathrm{cst} \ .$$

Therefore, solving the maximum-likelihood for ICA amounts to minimizing the discrepancy between the estimated sources y and the target independent sources of density d. We can go further in the decomposition. Denote d_y' the density of a vector of independent entries with the same marginals as y : $d_y'(y) = \prod_{i=1}^{p} d_{y_i}(y_i)$. Simple computations

show:

$$\mathrm{KL}(d_y, d) = \mathrm{KL}(d_y, d'_y) + \mathrm{KL}(d'_y, d)$$

The first term is called the *mutual information* of y.

Definition 1.31 (Mutual information)**.** *Let d_y a density over $\mathbb{R}^p$, and d'_y the density of independent components with same marginals as d_y. The mutual information of d is:*

$$\mathrm{MI}(d_y) = \mathrm{KL}(d_y, d'_y) \ .$$

By definition, we have $\mathrm{MI}(d_y) \geq 0$ and $\mathrm{MI}(d_y) = 0$ if and only if d_y corresponds to the density of independent components. Further, since d'_y and d fatorize, we have up to a constant:

$$\mathcal{L}(W, d) = \mathrm{MI}(d_y) + \sum_{i=1}^{p} \mathrm{KL}(d_{y_i}, d_i) \ . \tag{1.40}$$

In summary, the total mismatch of the model $\mathrm{KL}(d_y, d)$ is the sum of the deviation from independence $\mathrm{MI}(d_y)$ and the mismatch between the marginal densities of the y and the assumed distribution d. Therefore, minimizing the negative log-likelihood $\mathcal{L}$ with respect to W corresponds to finding a balance between independence and model matching. Going back to the example in fig. 1.7, at $\theta = 0$, $\sum_{i=1}^{p} \mathrm{KL}(d_{y_i}, d_i)$ is approximately minimized for models 1 and $*$, while it is maximal for model 4.

1.4.4 Equivariance and multiplicative updates

In this section, we investigate the important notion of equivariance, which leads to estimators with uniform performance: their statistical performance does not depend on the mixing matrix, only on the sources.

First, we introduce equivariant estimators:

Definition 1.32 (ICA estimator)**.** *An ICA estimator is an estimator $\mathcal{A}$ mapping a dataset $X \in \mathbb{R}^{p \times n}$ to an estimated mixing matrix $\hat{A} \in \mathrm{GL}_p$:*

$$\mathcal{A}(X) = \hat{A} \ .$$

The sources found by the estimator $\mathcal{A}$ from the dataset X are obtained by $\hat{S} = \Big(\mathcal{A}(X)\Big)^{-1} X$. An ICA estimator is called equivariant when it is equivariant with respect to the left multiplication by invertible matrices [Cardoso and Laheld, 1996]. We should also take the ICA indeterminacies discussed in the previous subsection 1.4.1 into account:

Definition 1.33 (Equivariant estimator for ICA)**.** *An ICA estimator $\mathcal{A}$ is equivariant if for all $X \in \mathbb{R}^{p \times n}$, $M \in \mathrm{GL}_p$, there exists a scale-permutation matrix C such that:*

$$\mathcal{A}(MX) = M\mathcal{A}(X)C \ .$$

Equivariant estimators have the extremely important property of *uniform performance*:

Proposition 1.34 (Uniform performance)**.** *Assume that $\mathcal{A}$ is an equivariant estimator, and $X = AS$, with $A \in \mathrm{GL}_p$. Then, there is a scale permutation matrix C such that:*

$$\mathcal{A}(X) = A\mathcal{A}(S)C \ .$$

In particular, the estimated sources from X are:

$$\hat{S} = \Big(\mathcal{A}(X)\Big)^{-1} X = C^{-1}\Big(\mathcal{A}(S)\Big)^{-1} S \ ,$$

which only depends on A through a scale and permutation: with respect to the ICA invariances, it does not depend on A.

The estimated sources of an equivariant estimator do not depend on the mixing matrix: the statistical performance of an equivariant estimator does not depend on the mixing matrix. Importantly, the maximum-likelihood estimator is equivariant when the maximum of the likelihood is reached (up to indeterminacies) for a single matrix.

Proposition 1.35 (Equivariance of the maximum likelihood estimator)**.** *The maximum-likelihood estimator with the source density model d is given by:*

$$\mathcal{A}(X) \in \operatorname*{arg\,min}_{A \in \mathrm{GL}_p} \ \mathcal{L}(A^{-1}, d, X) \ .$$

Assume that for any $A, A' \in \arg\min_A \mathcal{L}(A^{-1}, d, X)$, there is a scale-permutation matrix C such that $A = A'C$ (i.e. all the global minima of the log-likelihood are equal up to a scale and permutation). Then, for any $M \in \mathrm{GL}_p$, the global minimum of $\mathcal{L}(A^{-1}, d, MX)$ is also reached for a single matrix up to scale and permutation, and $\mathcal{A}(MX) = M\mathcal{A}(X)C$ for C a scale-permutation matrix.

Proof Consider $M \in \mathrm{GL}_p$. We have the simple relationship:

$$\mathcal{L}(A^{-1}, d, MX) = \log|A| - \sum_{i=1}^{p} \hat{E}[\log(d_i([A^{-1}MX]_i))] \tag{1.41}$$

$$= \mathcal{L}(A^{-1}M, d, X) + \log|M| \tag{1.42}$$

$$\tag{1.43}$$

Therefore, the minima of $\mathcal{L}(A^{-1}, d, MX)$ are characterized by those of $\mathcal{L}(A^{-1}, d, X)$:

$$A \in \arg\min \mathcal{L}(A^{-1}, d, X) \iff MA \in \arg\min \mathcal{L}(A^{-1}, d, MX) \ .$$

The advertised result follows. ■

Note that the maximum-likelihood invariance is specific to datasets verifying this property of unique minimum up to scale and permutation.

We now relate equivariant estimators to multiplicative algorithms.

Definition 1.36 (Multiplicative algorithm)**.** *A multiplicative algorithm to unmix the signal matrix X starts from an initial unmixing guess $W_0 \in \mathrm{GL}_p$ and iterates for $t = 0 \cdots T-1$:*

- *Compute the current estimated signals $Y_t = W_t X$, and a relative update $\mathcal{U}(Y_t) \in \mathbb{R}^{p \times p}$.*
- *Update the estimated unmixing matrix with learning rate ρ_t: $W_{t+1} = (I_p - \rho_t \mathcal{U}(Y_t))W_t$.*

The algorithm returns $\hat{A} = W_T^{-1}$.

Now, assume that $X = AS$, and consider $U_t = W_t A$. We have $Y_t = U_t S$, and the update equation writes:

$$U_{t+1} = (I_p - \rho_t \mathcal{U}(U_t S))U_t \ .$$

Therefore the evolution and the performance of the system is only determined by the initial matrix $U_0 = W_0 A$, and a study of the behavior of U_t completely characterizes the behavior of the algorithm: this is an *equivariant* algorithm. A striking property of these algorithms is that they are blind to the conditioning of the matrix A itself, only on the conditioning of U_0.

1.4.5 Non-Gaussian ICA: maximum-likelihood algorithms

Consider a dataset X and target density d for the sources. The relative gradient and Hessian of the negative log-likelihood (Eq. (1.38)) are given by writing the expansion of $\mathcal{L}((I_p + \mathcal{E})W)$ for $\mathcal{E} \in \mathbb{R}^{p \times p}$ a matrix of small norm (see also Section 1.3.3):

$$\mathcal{L}((I_p + \mathcal{E})W) = \mathcal{L}(W) + \langle G(W), \mathcal{E} \rangle + \frac{1}{2}\langle \mathcal{E}, \mathcal{H}(W)\mathcal{E} \rangle + o(\|\mathcal{E}\|^2) \ ,$$

where $G(W)$ is the relative gradient at W (a $p \times p$ matrix) and $\mathcal{H}(W)$ is the relative Hessian at W (a linear operator from $\mathbb{R}^{p \times p}$ to $\mathbb{R}^{p \times p}$, represented as a fourth-order tensor of shape $p \times p \times p \times p$). In the following, we denote ψ_i the score function associated with the density d (see Def. 1.30). Denoting $Y = WX$, we have:

$$G(W)_{ab} = \hat{E}[\psi_a(Y_a)Y_b] - \delta_{ab} \ ,$$

where δ_{ab} is the Kronecker symbol. This can be written in a dense form:

$$G(W) = \frac{1}{n}\psi(Y)Y^\top - I_p \ .$$

The maximum-likelihood is reached for signals Y such that $\hat{E}[\psi_a(Y_a)Y_b] = \delta_{ab}$. This equation resembles to a decorrelation condition of the form $\hat{E}[Y_a Y_b] = \delta_{ab}$, with the addition of the non-linearity ψ_a, which renders the equations (a, b) and (b, a) non-symmetric. The Infomax algorithm [Bell and Sejnowski, 1995] is a popular ICA algorithm.

Definition 1.37 (Infomax)**.** *The Infomax algorithm starts from an initial guess $W_0 \in \mathbb{R}^p$ and iterates:*

$$W_{t+1} = W_t - \rho_t \left(\frac{1}{n}\psi(Y_t)X^\top - W_t^{-\top} \right) \ ,$$

where the gradient is either computed on the whole dataset, or using only a mini-batch of samples.

This is an Euclidean gradient descent algorithm on $\mathcal{L}$, and therefore it is a not a multiplicative algorithm. Importantly, this algorithm was not derived in the original paper to perform maximum-likelihood ICA, but Cardoso [1997] linked this method to maximum-likelihood ICA. The idea of using the relative gradient for Infomax is found in [Amari et al., 1996], and writes:

$$W_{t+1} = (I_p - \rho_t G(W_t))W_t \ .$$

Because the gradient is only a function of Y, this formulation is a multiplicative algorithm as defined in 1.36. The relative version proposed by Amari et al. [1996] is used by most practitioners becaude of its better behavior than the original algorithm. However, this algorithm is still referred to as Infomax in most ICA libraries, even though it is not the formulation of Bell and Sejnowski.

We finish by giving an important property of maximum-likelihood ICA:

Proposition 1.38 (Independent sources are a stationary point of the log-likelihood)**.** *Assume that we observe i.i.d. signals $X = AS$, where the rows of S are independent. Then, for $i \neq j$, $G(A^{-1})_{ij} \to 0$ as $n \to +\infty$, regardless of the model for the density of the sources. Further, if the sources have a scale such that $\mathbb{E}[\psi_i(s_i)s_i] = 1$, $G(A^{-1}) \to 0$.*

Proof The results follows naturally from the equation $G(A^{-1})_{ij} = \hat{E}[\psi(s_i)s_j] - \delta_{ij}$. ■

This result makes no assumption on the source model d: even when the density of the sources is far from the model, the gradient of the log-likelihood asymptotically cancels. This was observed in Fig. 1.7: $\theta = 0$ was a stationary point for each model. If we want to study the stability of such a point, we should go beyond first order and compute the Hessian. It is obtained as:

$$H(W)_{abcd} = \delta_{ad}\delta_{bc} + \delta_{ac}\hat{E}[\psi'_a(Y_a)Y_bY_d] \ .$$

First, we note that it is sparse: it only has $O(p^3)$ coefficients even though it is a $p \times p \times p \times p$ operator. Second, letting $n \to +\infty$ and assuming that the rows of Y are independent (meaning that we have solved the ICA problem), the Hessian simplifies:

$$\tilde{H}(W) = \delta_{ad}\delta_{bc} + \delta_{ac}\delta_{bd}\mathbb{E}[\psi'_a(y_a)]\mathbb{E}[y_b^2] \ . \tag{1.44}$$

This Hessian *approximation* is a block diagonal operator with blocks of size 2×2: for a matrix $M \in \mathbb{R}^{p\times p}$, denoting $M' = \tilde{H}(W)M$, we find:

$$\begin{bmatrix} M'_{ab} \\ M'_{ba} \end{bmatrix} = \begin{bmatrix} \mathbb{E}[\psi'_a(y_a)]\mathbb{E}[y_b^2] & 1 \\ 1 & \mathbb{E}[\psi'_b(y_b)]\mathbb{E}[y_a^2] \end{bmatrix} \begin{bmatrix} M_{ab} \\ M_{ba} \end{bmatrix} \ .$$

In particular, we identify an important stability condition: $\tilde{H}$ is positive if and only if all the 2×2 blocks are positive.

Proposition 1.39 (Stability condition of the ICA solution [Amari et al., 1997])**.** *A^{-1} is a local minimum of $\mathcal{L}$ if and only if $G(A^{-1}) = 0$ and :*

$$\textit{For all } i, j, \ \ \mathbb{E}[\psi'_i(s_i)]\mathbb{E}[s_i^2]\mathbb{E}[\psi'_j(s_j)]\mathbb{E}[s_j^2] > 1 \ ,$$

where the expectations are taken with respect to the true densities d_i^*, d_j^* and ψ'_i is the score of the model d_i.

Under this condition, an algorithm solving maximum likelihood ICA starting close enough from A^{-1} will recover A. If this condition is not met, A^{-1} is a saddle point and will never be reached by a practical algorithm. To the best of our knowledge, there is sadly no global result on maximum-likelihood estimation in ICA, like conditions upon which all local minima of the log-likelihood are separating matrices.

The Picard algorithm, which we discuss in the next chapter, uses this Hessian approximation and a multiplicative framework to obtain a fast algorithm for maximum-likelihood ICA.

1.4.6 Orthogonal algorithms

Decorrelation is a necessary condition for independence. Therefore, many ICA algorithms enforce an orthogonal constraint. Given a dataset $X \in \mathbb{R}^{p\times n}$, the covariance of X is $C_X = \frac{1}{n}XX^\top$, and we say that X is decorrelated when $C_X = I_p$. A first step for ICA is often to decorrelate X by a linear transform, that is finding a matrix W such that WX is decorrelated. We have:

Proposition 1.40 (Decorrelating transforms)**.** *Let $X \in \mathbb{R}^{p\times n}$. For a matrix $W \in \mathrm{GL}_p$, we have:*

$$C_{WX} = I_p \text{ if and only if } W = UC_x^{-1/2} \text{ where } U \in \mathcal{O}_p \ .$$

Proof Indeed, $C_{WX} = I_p$ if and only if $WC_XW^\top = I_p$, which is rewritten as: $\left(WC_X^{1/2}\right)\left(WC_X^{1/2}\right)^\top = I_p$, i.e.$U = WC_X^{1/2} \in \mathcal{O}_p$. ■

Therefore, the set of decorrelating transforms is, up to a matrix multiplication, equal to the orthogonal group. Orthogonal algorithms look for an unmixing matrix of the form $W = UC_x^{-1/2}$ with $U \in \mathcal{O}_p$. However, due to the finite number of samples, this inherently limits the efficiency of such algorithms. Indeed, the covariance matrix of the sources is not really I_p: $C_S \neq I_p$. Therefore, A is not of the form $UC_x^{-1/2}$. This discrepancy can be quantified [Cardoso, 1994].

Still, the drop in efficiency is usually slim compared to model mismatch, and the orthogonal constraint can be leveraged to obtain fast algorithms. In the following, we assume that the data matrix has been decorrelated with a decorrelating transform: $C_X = I_p$. Therefore, orthogonal algorithms look for orthogonal unmixing matrices. FastICA [Hyvärinen, 1999] works under decorrelation constraint, and is a fast fixed point algorithm for ICA. The derivation of FastICA starts for one unit, where a vector $w \in \mathbb{R}^p$ such that $\|w\| = 1$ is of power one. For a non-quadratic function Γ, FastICA finds the extremal points of:

$$\hat{E}[\Gamma(wX)] \text{ s.t. } \|w\| = 1 \ .$$

The Lagrangian of this problem writes $\hat{E}[\Gamma(wX)] - \frac{1}{2}\beta\|w\|^2$, where $\beta \in \mathbb{R}$. Canceling its gradient with respect to w yields:

$$\hat{E}[g(wX)X] - \beta w = 0 \ ,$$

where $g = \Gamma'$.

FastICA solves this equation with Newton's method. The Jacobian of the previous term is:

$$\frac{1}{n}\sum_{j=1}^{n} g'([wX]_j)X_j \otimes X_j - \beta I_p \ ,$$

which is approximated as $(\hat{E}[g'(wX)] - \beta)I_p$, using $X_j \otimes X_j \simeq I_p$. Therefore, the approximate Newton's method with step-size 1 writes:

$$w \leftarrow w - \frac{1}{\hat{E}[g'(wX)] - \beta}(\hat{E}[g(wX)X] - \beta w) \ . \tag{1.45}$$

A normalizing step is performed afterwards:

$$w \leftarrow \frac{w}{\|w\|} \ .$$

Since a normalizing step is applied, multiplying the right-hand side of eq. 1.45 by a scalar leaves the iterations unchanged (or flips the signs of w). Doing so with $\beta - \hat{E}[g'(wX)]$ yields the practical form of FastICA:

$$w \leftarrow \hat{E}[g(wX)X] - \hat{E}[g'(wX)]w \ , \tag{1.46}$$

$$w \leftarrow \frac{w}{\|w\|} \ . \tag{1.47}$$

An extension to the estimation of the whole mixture and of a demixing matrix $W \in \mathcal{O}_p$ is proposed by performing the updates in parallel:

$$W \leftarrow \hat{E}[g(WX)X] - \mathrm{diag}(\hat{E}[g'(WX)])W, \tag{1.48}$$

$$W \leftarrow (WW^\top)^{-\frac{1}{2}}W \ , \tag{1.49}$$

where the last operation corresponds to a projection on $\mathcal{O}_p$.

So far, the algorithm is not a multiplicative algorithm, but it can simply be turned into one by doing:

$$W \leftarrow \left(\sum_{j=1}^{n} \psi(Y_j)Y_j - \mathrm{diag}(\sum_{j=1}^{n} \psi'(Y_j)) \right) W, \tag{1.50}$$

$$W \leftarrow (WW^\top)^{-\frac{1}{2}}W \ , \tag{1.51}$$

where implicitly, $Y = WX$. In [Hyvärinen, 1999], this approach is linked to maximum-likelihood ICA, through the choice of density. Assume that we have a binary density model:

$$-\log(d(y)) = y^2 + \epsilon\Gamma(y) + \mathrm{cst} \ ,$$

where Γ is such that $\Gamma(y) = o(y^2)$ as $y \to \pm\infty$, and ϵ is $+1$ or -1. The "log-likelihood" for this model, under unit norm constraint, writes:

$$\mathcal{L}(w, d) = \underbrace{\hat{E}[(wX)^2]}_{\text{constant}} + \epsilon\hat{E}[\Gamma(wX)] \ .$$

The local minima of $\mathcal{L}$ when $\epsilon = \pm 1$ are exactly the extremal points of $\hat{E}[\Gamma(wX)]$. This analysis is more difficult to conduct in the general case where the whole matrix W is estimated. In section 2.2.4 of this thesis, we show under mild conditions that the fixed points of FastICA indeed correspond to local minima of the log-likelihood with a binary density model, and that the iterations of FastICA match those of a quasi-Newton algorithm on the orthogonal manifold, and that a version of the Picard algorithm on the orthogonal manifold is faster than FastICA on real datasets.

1.4.7 Different routes to ICA

So far, we have studied the ICA likelihood under the non-Gaussian i.i.d. hypothesis. As we have seen previously, a mixture of Gaussian white noise (i.e. Gaussian i.i.d. samples) is not identifiable. Non-Gaussianity is a way to depart from Gaussian white noise, but two other simple deviations are also worth considering :

- Non-stationary samples [Pham and Cardoso, 2001]: samples are no longer supposed to be identically distributed.
- Time-correlated samples [Pham and Garat, 1997]: samples are no longer supposed to be independent from one another.

Just like non-Gaussian ICA, these two models lead to simple maximum-likelihood inference.

Non-stationary ICA A simple non-stationary model assumes that the density model is Gaussian, with a variance varying over time: the density of the i-th source and sample j is $d_{ij} = \mathcal{N}(0, \sigma_{ij}^2)$. The negative log-likelihood of this model writes as the sum of negative log-likelihoods over samples. Up to a constant:

$$\mathcal{L}(W, \sigma) = -\log|W| + \frac{1}{2n}\sum_{i=1}^{p}\sum_{j=1}^{n}\left[\frac{[WX]_{ij}^2}{\sigma_{ij}^2} + \log(\sigma_{ij}^2)\right] \quad . \tag{1.52}$$

So far, this model is highly unidentifiable: there are as many variance parameters σ^2 as there are data points. A customary hypothesis is that the variance profiles are piecewise constant over L intervals: there are $L+1$ integers $1 = n_1 < \cdots < n_{L+1} = n+1$ such that $\sigma_{ij}^2 = \lambda_{il}$ when $j \in [n_l, n_{l+1}[$. Denote $C_l = \frac{1}{\gamma_l}\sum_{j=n_l}^{n_{l+1}-1} X_j \otimes X_j$ the covariance of X in the l-th interval, where $\gamma_l = n_{l+1} - n_l$ is the interval length. We find:

$$are\mathcal{L}(W, \lambda) = -\log|W| + \frac{1}{2n}\sum_{i=1}^{p}\sum_{l=1}^{L}\gamma_l\left[\frac{[WC_lW^\top]_{ii}}{\lambda_{il}} + \log(\lambda_{il})\right] \quad .$$

Minimization with respect to λ_{il} is straightforward and yields $\lambda_{il} = [WC_lW^\top]_{ii}$. Up to a constant, the cost function therefore rewrites:

$$\mathcal{L}(W) = -\log|W| + \frac{1}{2n}\sum_{i=1}^{p}\sum_{l=1}^{L}\gamma_l\log([WC_lW^\top]_{ii}) \quad .$$

Using the identity $\log|WC_lW^\top| = 2\log|W|are + \log|C_l|$, the cost function is finally obtained up to a constant as:

$$\mathcal{L}(W) = \frac{1}{2n}\sum_{l=1}^{L}\gamma_l\log\left(\frac{|\mathrm{diag}(WC_lW^\top)|}{|WC_lW^\top|}\right) \quad . \tag{1.53}$$

Interestingly, this is a *joint-diagonalization* criterion, which verifies [Pham, 2001b]:

- $\mathcal{L}(W) \geq 0$.

- $\mathcal{L}(W) = 0$ if and only if $WC_lW^\top$ is a diagonal matrix for all l.

Therefore, solving ICA for this model amounts to finding an unmixing matrix W such that the matrices $WC_lW^\top$ are as diagonal as possible, where the measure of non-diagonality is given by Eq. (1.53). There is a simple explanation. In the limit where the interval lengths go to infinity: $\gamma_l \to +\infty$, we have:

$$C_l = \mathbb{E}[X_j \otimes X_j] = A \underbrace{\mathbb{E}[S_j \otimes S_j]}_{\text{diagonal}} A^\top \ ,$$

therefore the matrices C_l are jointly-diagonalized by A^{-1}.

This model is identifiable when no pair of sources have proportional variance profiles [Matsuoka et al., 1995]: this model leverages power diversity.

Time-correlated ICA This model should be understood as the Fourier dual of the non-stationary model. Denote $\tilde{s}_a(l) = \frac{1}{\sqrt{n}} \sum_{j=1}^n s_{aj} \exp(-\frac{2\pi i l j}{n})$ the l-th Fourier coefficient of the a-th source s_a. These coefficients are decorrelated as $n \to +\infty$, with variance P_{al} given by the power spectrum. The log-likelihood is then given by:

$$\mathcal{L}(W) = -\log|W| + \frac{1}{2n} \sum_{a=1}^{p} \sum_{l=1}^{n} \left[\frac{|\tilde{Y}_{al}|^2}{P_{al}} + \log(P_{al}) \right] \ ,$$

where $\tilde{Y}$ is the Fourier transform with respect to the rows of $Y = WX$. The resemblance with Eq. (1.52) is striking: the instantaneous power Y_{ij}^2 is replaced by the spectral instantaneous power $|\tilde{Y}_{al}|^2$, and the Gaussian variance σ_{ij}^2 is replaced by the power spectrum P_{al}. The piecewise constant hypothesis leads to piecewise constant spectra, and the estimation of the mixing is made by joint-diagonalizing the spectral covariance matrices $\tilde{C}_l$ corresponding to the covariance of the signals filtered in each frequency band of interest.

This model is identifiable when no pair of sources have proportional spectra Pham and Garat [1997]: it is a model leveraging spectral diversity.

In this thesis, we use an extension of this model to extract meaningful brain sources in Chapter 4.

Tensor-based approaches Another line of research consists in using a tensorial framework to extract sources [Sidiropoulos et al., 2000]. This approach is best explained by considering a continuous problem in "n": the sources are continuous functions $s_j(t) : \mathbb{R} \to \mathbb{R}$. Consider a kernel function $\phi(\tau, t, f) : \mathbb{R}^3 \to \mathbb{R}$, where for instance t represents time and f frequency. Assume that the sources factorize with respect to the kernel transform:

$$\int_{\mathbb{R}} s_j(\tau)\phi(\tau, t, f)d\tau = B_j(t) \times C_j(f) \ , \tag{1.54}$$

where $B_j(t)$ and $C_j(f)$ are the time and frequency components of the source s_j. Consider an ICA model $x_i = \sum_{j=1}^p A_{ij}s_j$. Let $T_i(t, f) = \int_{\mathbb{R}} x_i(\tau)\phi(\tau, t, f)d\tau$. Eq. (1.54) gives:

$$T_i(t, f) = \sum_{j=1}^{p} A_{ij} \times B_j(t) \times C_j(f) \ .$$

Considering discretized time intervals $t_1, \cdots t_q$ and frequency intervals $f_1, \cdots, f_r$, the third order tensor $\mathcal{T} \in \mathbb{R}^{p \times q \times r}$ of entries $\mathcal{T}_{ikl} = T_i(t_k, f_l)$ factorizes in a low rank decomposition:

$$\mathcal{T}_{ikl} = \sum_{j=1}^{p} A_{ij} \times B_{kj} \times C_{lj} \ ,$$

where $B \in \mathbb{R}^{q \times p}$ and $C \in \mathbb{R}^{r \times p}$. This decomposition is a polyadic decomposition of $\mathcal{T}$ of order p. Importantly, this decomposition is unique (up to trivial scale and ordering ambiguities) as long as a condition on the k-rank of the factors A, B, C is verified [Kruskal, 1977], or on the coherence of those matrices [Lim and Comon, 2013]. Therefore, the mixing matrix A is recovered by computing the polyadic decomposition of $\mathcal{T}$: ICA is performed by tensor decomposition. At first glance, it may not be clear where the independence assumption between the sources is used. It is hidden in the conditions for the unicity of the polyadic decomposition, through properties of the matrices B and C.

This framework relies crucially on the source factorization hypothesis of Eq. (1.54); and the kernel ϕ should be chosen accordingly. For instance, short-time Fourier or wavelet kernel constitutes a prototypical choice for ϕ, leading to the so-called space-time-frequency component analysis used in a variety of literature and giving interesting decomposition on brain signals [Miwakeichi et al., 2004, De Vos et al., 2007, Mørup et al., 2006, Weis et al., 2009].

Other approaches for tensor-based ICA, exploiting different transforms of the sources have also been developed, like spatial-time-wave-vector [Becker et al., 2012, 2016] or polarization [Raimondi and Comon, 2015].

1.4.8 ICA models with noise

We now turn to an extensions of the classical ICA model $X = AS$, by adding a noise term:

$$X = AS + N \ ,$$

where the noise N is assumed to be independent of S. First of all, this allows to estimate a different number of sources than sensors (i.e. having a non-square mixing matrix), without degenerate model. Therefore, in this section, we denote q the number of sources, so that $A \in \mathbb{R}^{p \times q}$, and $S \in \mathbb{R}^{q \times n}$. A first difficulty with such model is that estimation of A does not give access to the sources, and vice versa. This leads to some identifiability issues [Davies, 2004]: under usual assumptions, A is identifiable up to scale and permutation, but not the sources (and in particular their density). The tractability of this model differ greatly depending on which of the three routes to ICA one takes.

Non-Gaussian ICA For non-Gaussian ICA, a natural assumption is to assume that N contains i.i.d. Gaussian variables of variance σ^2. Let d denote the model on the density of the sources. The likelihood of A for a sample x writes, by marginalization

over the sources [Lewicki and Sejnowski, 2000]:

$$\ell(A, x) = \int_{s \in \mathbb{R}^q} p(x|A, s) d(s) ds \tag{1.55}$$

$$= \int_{s \in \mathbb{R}^q} \frac{1}{\sqrt{2\pi\sigma^{2^p}}} \exp\left(-\frac{1}{2\sigma^2}\|x - As\|^2\right) \prod_{i=1}^{q} d_i(s_i) ds \ . \tag{1.56}$$

This integral is not tractable because of the entanglement between x and s. Therefore, one can use the EM algorithm to estimate the parameters of the model, using the sources as a latent variable. The M-step writes simply:

$$A = R_{xs} R_{ss}^{-1} \ ,$$

where $R_{xs} = \hat{E}\left[x \otimes \mathbb{E}[s|A, x]\right]$ and $R_s s = \hat{E}\left[\mathbb{E}[s \otimes s|A, x]\right]$ are the sufficient statistics. The estimation of these correlation matrices is tedious, and greatly depends on the density model d: a solution modeling d as a mixture of Gaussian is discussed in [Moulines et al., 1997], it is computationally intensive.

An alternative is to use objective functions similar to dictionary learning [Olshausen and Field, 1996]:

$$A, S \in \arg\min \frac{1}{2}\|X - AS\|^2 + \lambda \sum_{i=1}^{q} \sum_{j=1}^{n} G(S_{ij}) \text{ s.t. } \mathcal{C}(A) = A \tag{1.57}$$

where G is a non-quadratic function and $\mathcal{C}(A)$ is a function imposing constraints on A, like $\mathcal{C}(A) = A^\top A - I_q$ to impose orthogonality or $\mathcal{C}(A) = \sum_{i=1}^{q} |\|A_i\| - 1|$ to impose unit-norm constraints. In the square $p = q$ case, this model relates to maximum likelihood ICA in the limit of no regularization.

Indeed, when $\lambda \to 0$, problem (1.57) becomes the constrained problem:

$$A, S \in \arg\min \sum_{i=1}^{q} \sum_{j=1}^{n} G(S_{ij}) \text{ s.t. } \mathcal{C}(A) = 0 \text{ and } X = AS \ , \tag{1.58}$$

which is non-Gaussian maximum-likelihood ICA with model on the sources $d_i = \exp(-G)$, under the condition that $\log|A|$ is constant under the constraint $\mathcal{C}(A) = 0$.

Non-stationary and time correlated ICA Thanks to their underlying Gaussian assumptions, these models incorporate the additional noise term with more ease. However, this approach has seldom been used in practice. We show a practical algorithm for this method in Chapter 4.

1.4.9 Applications of ICA: M/EEG signal processing

ICA is widely used in observational sciences, as it is a powerful unsupervised data exploration technique. It is used by several communities: astronomy [Nuzillard and Bijaoui, 2000, Maino et al., 2002, Cadavid et al., 2008], chemistry [Vrabie et al., 2007, Rutledge and Bouveresse, 2013], biology [Lee and Batzoglou, 2003, Scholz et al., 2004], or mechanical engineering [Poncelet et al., 2007]. The goal of this thesis is to develop better ICA algorithms for real life applications, with brain signal processing (and neuroscience) as the first application in mind.

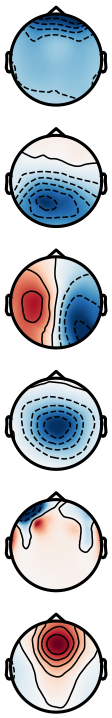


Figure 1.8 – Some ICA topographies

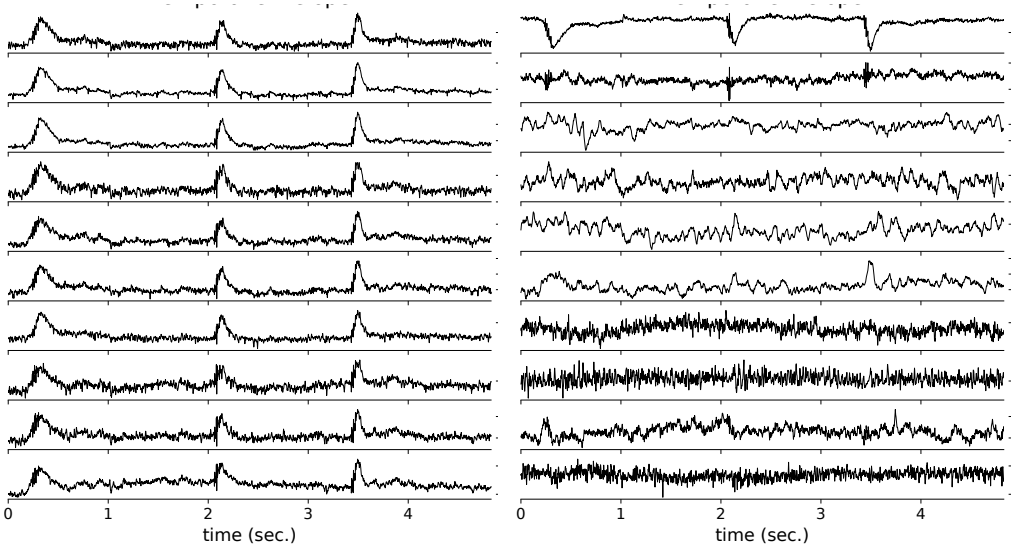
ICA is widely used to process the three principal non-invasive brain data acquisition techniques: electroencephalography (EEG) [Niedermeyer and Lopes da Silva, 2005], magnetoencephalography (MEG) [Hämäläinen et al., 1993], and functional magnetic resonance imaging (fMRI) [Huettel et al., 2004]. We especially focus on M/EEG data. Magnetoencephalography and electroencephalography respectively acquire magnetic and electric field at the level of the scalp. A typical research EEG device has $p \sim 50$ sensors, and a MEG device has ~ 100 sensors. A major problem in neuroscience is that of source estimation: the goal is to infer the sources that lead to the measurements. This is the M/EEG *inverse problem*. The *forward problem* models how sources lead to the measurement. Sources are usually modeled as dipoles located at a specific location in the brain. Thanks to linearity of Maxwell's equations, the electric / magnetic field received at the level of a sensor is a linear combination of the emission of those source dipoles. Denoting $X \in \mathbb{R}^{p \times n}$ the observed signals, it holds:

$$X = DZ,$$

where $D \in \mathbb{R}^{p \times q}$ is the *forward operator* and $Z \in \mathbb{R}^{q \times n}$ is the source activity. Some noise coming from other sources than the brain can also be taken into account. Usually, the brain is divided into a grid of thousands of small cubes, at the center of which dipoles are placed. This model assumes 1000's of potential sources, while in practice only a few are active: therefore, it is customary to employ regularized models to solve the inverse problem.

ICA offers an alternative. The linearity of the forward model means that ICA is a candidate of choice to extract brain sources [Makeig et al., 1997]. It amounts to assuming that the measurement X comes from only a few different sources: at most as many as there are sensors. Applying ICA on EEG/MEG data reveals many different types of signals. Figure 1.9 displays a few seconds of 10 sensors of an *EEG* recording, and 10 sources found by ICA on those data.

As it appears, ICA is a powerful tool to capture artifacts: eye blinks, heartbeat, line noise, environmental noise... It is widely used by practitioners to clean the brain signal from those contaminations. It is extremely simple to do: given an ICA decomposition $X = AS$, if the source i is to be suppressed, one can simply set $s_i = 0$ yielding clean sources S_{clean} and reconstruct the cleaned signal $X_{\text{clean}} = AS_{\text{clean}}$. Further, an ICA source is obtained as a linear combination of sensors. This allows to interpolate on the sensor topography, and display the estimated electric/magnetic field. Such maps are displayed in figure 1.8. It can also be used to fit a dipole, and see how good the fit is: a source with good fit is called a dipolar source.Delorme et al. [2012] argues that an ICA decomposition should lead to such dipolar sources, and that it can in fact be used as a measure for the quality of a decomposition.



(a) Raw EEG (b) ICA sources

Figure 1.9 – Applying ICA to EEG data allows to separate the signal in independent components. The first independent component corresponds to the eye blinks, which contaminates a lot the original signals. Components 2-5 show prototypical brain oscillations like α or β rhythms. The high frequency components (7-8-10) correspond to muscular artifacts.

1.5 Contributions

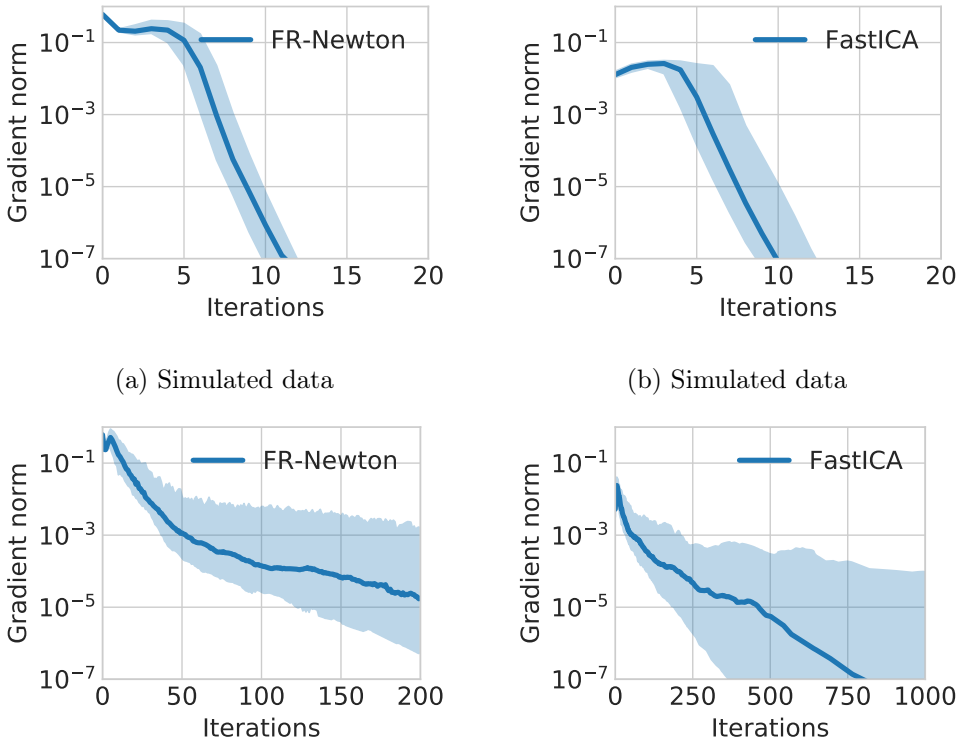
This thesis contains three main contributions. Now that we have introduced all the useful notions, we would like to motivate them with practical examples.

Chapter 2: Faster ICA by preconditioning with Hessian approximation This work is devoted to the problem of rapidly estimating the maximum-likelihood estimator for ICA, by minimizing $\mathcal{L}(W)$ in Eq. (1.38). A good starting point is the Hessian approximation in $\tilde{H}(W)$ in Eq. (1.44): it is very sparse, easy to invert and regularize, and quite cheap to compute.

It is therefore natural to propose a quasi-Newton method using $\tilde{H}$ instead of $\mathcal{H}$. Starting from $W_0 \in \mathrm{GL}_p$, this algorithm iterates:

$$W_{t+1} = \left(I_p - \rho_t \left(\tilde{H}(W_t) \right)^{-1} G(W_t) \right) W_t \ .$$

This exact algorithm is actually proposed by Zibulevsky [2003]. It works extremely well on synthetic data generated by the ICA generative model $X = AS$. However, it is much slower than one would expect applied on real datasets. The same observation can be made for the popular algorithm FastICA [Hyvärinen, 1999]. Figure 1.10 shows the behavior of these algorithms on synthetic and real data: on real data, the convergence is overall slow. What is happening? Recall that the Hessian approximation $\tilde{H}$ is obtained when the signals estimated by the algorithm are independent. For real data, the ICA model does not hold perfectly, and for most dataset X, there is no matrix W that make the rows of WX independent. The Hessian approximation is therefore limited.



(a) Simulated data

(b) Simulated data

(c) EEG data

(d) EEG data

Figure 1.10 – Behavior of Zibulevsky's algorithm (FR-Newton) and FastICA on synthetic and real data. To generate synthetic data, a source matrix of size 59×10000 where each entry follows a Laplace law ($S_{ij} \sim \frac{1}{2}\exp(-|x|)$) is generated. We mix it using a mixing matrix $A \in \mathbb{R}^{59\times59}$ with random Gaussian entries. The real data comes from an EEG dataset of the same size, 59×10000. The y-axis displays the gradient norm of the cost function $\|G(W)\|$. Shaded areas correspond to several runs of the algorithm.

In our first contribution, we propose to use $\tilde{H}$ as a first Hessian approximation in the L-BFGS algorithm, which is then going to refined to find a better approximation, closer to the true curvature of the loss function. This is the Preconditioned ICA for Real Data (Picard) algorithm. It is also extended to the orthogonal constraint (Section 2.2) in order to solve the same problem as FastICA. Experiments show that it is faster than other ICA algorithms on real data.

The code for Picard is available online at https://github.com/pierreablin/picard.

Chapter 3: Stochastic algorithms for ICA with descent guarantees Imagine that the data matrix at hand X contains an extremely large number of samples. This setting makes stochastic algorithms appealing: after seeing the whole dataset, a stochastic algorithm will have performed n steps, while a full-batch algorithm like Picard or FastICA will only perform one step. For ICA, the only stochastic technique employed is stochastic gradient descent, like the Infomax algorithm 1.37. As explained in the introduction, stochastic gradient descent is sometimes hard to properly set up, because of its learning rate. Figure 1.11 shows the convergence of Infomax on a toy

problem, with three learning rates.

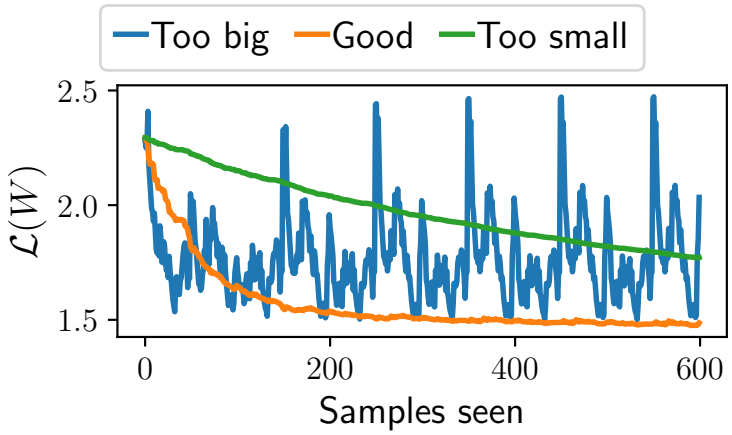


Figure 1.11 – Behavior of Infomax (relative stochastic gradient descent) on a toy 3×100 problem. Different curves correspond to different learning rates. This is an extremely important parameter which is hard to choose. This figure is sketchy, but similar problems occur when applying Infomax to M/EEG processing [Montoya-Martínez et al., 2017].

In our second contribution, we develop a majorization-minimization framework for minimizing $\mathcal{L}$, using the so called η-trick in order to majorize $-\log(p)$ by a quadratic. It leads to incremental algorithms with descent guarantees for ICA, that scale gracefully with the number of samples, are are much safer to use than SGD.

The code is available online at https://github.com/pierreablin/mmica.

Chapter 4: Spectral matching ICA for brain rhythms separation In our last contribution, we revisit SMICA, an ICA algorithm proposed in an astronomical setting [Delabrouille et al., 2003] (see Figure 1.12). It uses a noisy ICA model:

$$X = AS + N \ ,$$

where the sources are supposed to be time-correlated. This algorithm leverages spectral diversity of the sources. How does it perform on M/EEG data? Our experiments show that it leads to interesting decompositions, and that the noise model leads to many interesting features that noiseless ICA algorithms do no possess.

The code is available online at https://github.com/pierreablin/smica.

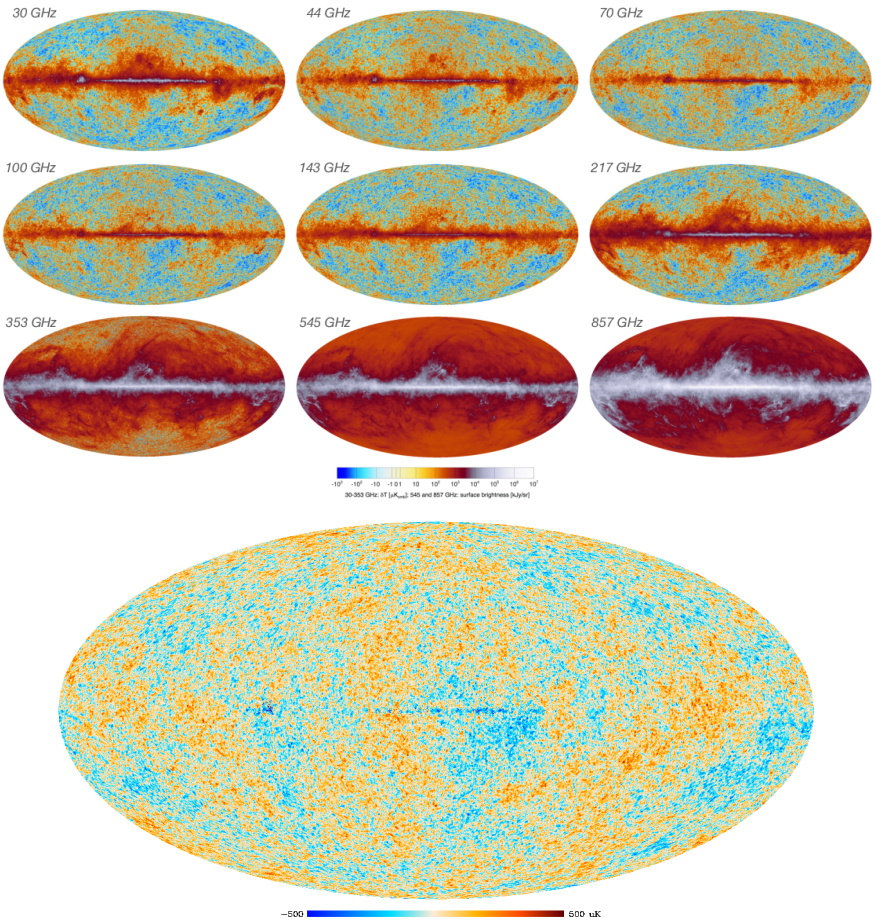


Figure 1.12 – The European mission Planck had for purpose to produce a clean map of the cosmological microwave background (CMB), which was emitted about 14 billion years ago. To do so, the Planck satellite took pictures of the universe in 9 frequency bands (top). Each picture is a linear combination of the background (the signal of interest), with foreground contaminations which are mutually independent. The problem of recovering the CMB is therefore an ICA problem. SMICA, an ICA algorithm based on spectral diversity and encompassing a noise model, was used to obtain the linear combination of the 9 top images to get the final snapshot of the CMB (bottom).

1.6 Publications

The different works presented in this document resulted in various publications and communications. Publications presented in this thesis are:

- Pierre Ablin, Jean-François Cardoso, and Alexandre Gramfort. Faster independent component analysis by preconditioning with Hessian approximations. *IEEE Transactions on Signal Processing*, 66(15):4040–4049, 2018a

- Pierre Ablin, Jean-François Cardoso, and Alexandre Gramfort. Faster ICA under orthogonal constraint. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018c

- Pierre Ablin, Jean-François Cardoso, and Alexandre Gramfort. Accelerating likelihood optimization for ICA on real signals. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 151–160. Springer, 2018b

- Pierre Ablin, Alexandre Gramfort, Jean-François Cardoso, and Francis Bach. Stochastic algorithms with descent guarantees for ICA. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1564–1573, 2019c

Two other articles borrowing ideas from [Ablin et al., 2018a] to accelerate optimization of different problems than ICA are not presented here:

- Pierre Ablin, Dylan Fagot, Herwig Wendt, Alexandre Gramfort, and Cédric Févotte. A quasi-Newton algorithm on the orthogonal manifold for NMF with transform learning. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 700–704. IEEE, 2019b

- Pierre Ablin, Jean-François Cardoso, and Alexandre Gramfort. Beyond Pham's algorithm for joint diagonalization. In *ESANN*, 2019a

Two other articles have been accepted at the NeurIPS 2019 conference:

- Pierre Ablin, Thomas Moreau, Mathurin Massias, and Alexandre Gramfort. Learning step sizes for unfolded sparse coding. *arXiv preprint arXiv:1905.11071*, 2019d

- David Sabbagh, Pierre Ablin, Gaël Varoquaux, Alexandre Gramfort, and Denis A Engeman. Manifold-regression to predict from meg/eeg brain signals without source modeling. *arXiv preprint arXiv:1906.02687*, 2019

Part I

Faster Independent Component Analysis

2

Faster ICA by preconditioning with Hessian approximations

Contents

2.1 The Picard algorithm

This section presents the work published in:

Pierre Ablin, Jean-François Cardoso, and Alexandre Gramfort. Faster independent component analysis by preconditioning with Hessian approximations. *IEEE Transactions on Signal Processing*, 66(15):4040–4049, 2018a

Only minor changes have been made: shorter introduction and notation harmonization.

2.1.1 Introduction

One of the most popular ICA algorithm is Infomax [Bell and Sejnowski, 1995], or rather its relative gradient implementation Amari et al. [1996]. It is widely used in neuroscience and is distributed in most neural processing toolboxes (e.g. EEGlab [Delorme and Makeig, 2004] or MNE [Gramfort et al., 2014]). The Infomax criterion can be shown to correspond to a likelihood function [Cardoso, 1997] based on a super-Gaussian component model. Consequently, a limitation of Infomax is that it can only separate super-Gaussian signals. Numerous significant neuroscience studies use Infomax in their data processing pipelines such as [Jung et al., 2000, McKeown et al., 1997, Allen et al.,

2011, Assaf et al., 2010, Allen et al., 2014, Wessel and Aron, 2015] to reference only a few. The purpose of this part is to provide a faster way of optimizing this popular criterion.

Infomax maximizes the likelihood using a stochastic gradient algorithm which may require some hand-tuning and often fails to converge, or only converges slowly [Montoya-Martínez et al., 2017]. Since speed is important in data exploration, various methods have been proposed for a faster maximization of the Infomax likelihood by using curvature information, that is by exploiting not only the gradient of the likelihood as in Infomax but also its second derivatives. We briefly review some of the methods found in the literature.

The most natural way of using curvature is to use the complete set of second derivatives (the Hessian) to set up the Newton method but it faces several difficulties: the Hessian is a large object, costly to evaluate and to invert for large data sets. It also has to be regularized since the ICA likelihood is not convex. The cost issue is addressed in [Tillet et al., 2017] by using a truncated Newton algorithm: an approximate Newton direction is found by an early stopping (truncation) of its computation via a conjugate gradient method. Further, each step in this computation is quickly computed by a 'Hessian-free' formula. Another approach to exploit curvature is to use approximations of the Hessian, obtained by assuming that the current signals are independent (see *e.g.* [Pham and Garat, 1997, Amari et al., 1997] or Section 2.1.2). For instance, a simple quasi-Newton method is proposed in [Zibulevsky, 2003] and in AMICA [Palmer et al., 2008], and a trust-region algorithm in [Choi and Choi, 2007].

We have re-implemented and compared these methods (see Section 2.1.7) and found that the Hessian approximations do yield a low cost per iteration but that they are not accurate enough on real data (which cannot be expected to follow the ICA model at high accuracy, *e.g.* in presence of some correlated sources). The approach investigated in this part overcomes this problem by using an optimization algorithm which 'learns' curvature from the past iterations of the solver (L-BFGS [Byrd et al., 1995]), and accelerates it by preconditioning with Hessian approximations.

This part is organized as follows. In Section 2.1.2, we recall the expression of the gradient and Hessian of the log-likelihood. We show how simple Hessian approximations can be obtained and regularized. That allows the L-BFGS method to be preconditioned at low cost yielding the Preconditioned ICA for Real Data (Picard) algorithm described in Section 2.1.5. In Section 2.1.6, we detail related algorithms mentioned in the introduction. Finally, Section 2.1.7 illustrates the superior behavior of the Picard algorithm by extensive numerical experiments on synthetic signals, on multiple electroencephalography (EEG) datasets, on functional Magnetic Resonance Imaging (fMRI) data and on natural images.

2.1.2 Likelihood and derivatives

Non Gaussian likelihood for ICA

The negative log-likelihood of ICA is given by:

$$\mathcal{L}(W) = -\log|W| - \hat{E}\left[\sum_{i=1}^{p} \log(d_i(y_i(t))\right] \quad , \tag{2.1}$$

where $\hat{E}$ denotes the empirical mean (sample average) and where, implicitly, $Y = WX$. Our aim is to minimize $\mathcal{L}(W)$ with respect to W which amounts to solving the ICA problem in the maximum likelihood sense.

We note from the start that this optimization problem is not convex for a simple reason: if W^* minimizes the objective function, any permutation of the columns of W^* gives another equivalent minimizer.

In this part, we consider the fast and accurate minimization of $\mathcal{L}(W)$ for a given source model, that is, working with fixed predetermined densities d_i, equal for all channels. It corresponds to the standard Infomax model commonly used in practice. In particular, our experiments use $-\log(d_i(\cdot)) = 2\log(\cosh(\cdot/2)) + \text{cst}\ \ \forall i$, which is the density model in standard Infomax, but any density that captures super-Gaussian signals could be used instead.

In the following, the *ICA mixture model* is said to hold if the signals actually are a mixture of independent components. We stress that *on real data*, the ICA mixture model is not expected to hold exactly. This may happen for several reasons depending on the origin of the data, among which convolutions, partially correlated sources, a different number of sources than sensors, non-stationarity or non-i.i.d sources. Yet, the linear ICA model is widely used in observational sciences.

Relative variations of the objective function

The variation of $\mathcal{L}(W)$ with respect to a *relative variation* of W is described, up to second order, by the Taylor expansion of $\mathcal{L}((I+\mathcal{E})W)$ in terms of a 'small' $p \times p$ matrix $\mathcal{E}$:

$$\mathcal{L}((I+\mathcal{E})W) = \mathcal{L}(W) + \langle G, \mathcal{E}\rangle + \frac{1}{2}\langle \mathcal{E}, H\mathcal{E}\rangle + \mathcal{O}(||\mathcal{E}||^3). \tag{2.2}$$

The first order term is controlled by the $p \times p$ matrix G, called the relative gradient [Cardoso and Laheld, 1996] and the second-order term depends on the $p \times p \times p \times p$ tensor H, called the relative Hessian [Zibulevsky, 2003]. Both these quantities can be obtained from the second order expansions of $\log|\cdot|$ and $\log d_i(\cdot)$:

$$\log|I_p + \mathcal{E}| = \operatorname{Tr}(\mathcal{E}) - \frac{1}{2}\operatorname{Tr}(\mathcal{E}^2) + \mathcal{O}(||\mathcal{E}||^3),$$
$$\log(d_i(y+e)) = \log(d_i(y)) - \psi_i(y)e - \frac{1}{2}\psi_i'(y)e^2 + \mathcal{O}(e^3),$$

for $\mathcal{E}$ a small matrix and e a small real number, where $\psi_i = -\frac{d_i'}{d_i}$ is called the *score function* (equal to $\tanh(\cdot/2)$ for the standard Infomax density). Collecting and re-arranging terms yields at first-order the classic expression

$$G_{ij} = \hat{E}[\psi_i(y_i)y_j] - \delta_{ij} \quad \text{or} \quad G(Y) = \frac{1}{n}\psi(Y)Y^\top - Id \tag{2.3}$$

and, at second order, the relative Hessian:

$$H_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik}\,\hat{E}[\psi_i'(y_i)y_j y_l]\ . \tag{2.4}$$

Note that the relative Hessian is sparse. Indeed, it has only of the order of p^3 non-zero coefficients: $\delta_{il}\delta_{jk} \neq 0$ for $i = l$ and $j = k$ which corresponds to p^2 coefficients, and $\delta_{ik} \neq 0$ for $i = k$ which happens p^3 times. This means that for a practical application with 100 sources the Hessian easily fits in memory. However, its computation requires

the evaluation of the terms $\hat{E}[\psi_i'(y_i)y_j y_l]$, resulting in a $\Theta(p^3 \times n)$ complexity. This fact and the necessity of regularizing the Hessian (which is not necessarily positive definite) in Newton methods motivate the consideration of Hessian approximations which are faster to compute and easier to regularize.

2.1.3 Hessian Approximations

The Hessian approximations are discussed on the basis of the following moments:

$$\begin{cases} \hat{h}_{ijl} = \hat{E}[\psi_i'(y_i)y_j y_l] \text{ , for } 1 \leq i,j,l \leq p \\ \hat{h}_{ij} = \hat{E}[\psi_i'(y_i)y_j^2] \text{ , for } 1 \leq i,j \leq p \\ \hat{h}_i = \hat{E}[\psi_i'(y_i)] \text{ , for } 1 \leq i \leq p \\ \hat{\sigma}_i^2 = \hat{E}[y_i^2] \text{ , for } 1 \leq i \leq p \end{cases} . \tag{2.5}$$

Hence, the true relative Hessian is $H_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik}\hat{h}_{ijl}$. A first approximation of H consists in replacing $\hat{h}_{ijl}$ by $\delta_{jl}\hat{h}_{ij}$. We denote that approximation by $\tilde{H}^2$:

$$\tilde{H}^2_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik}\delta_{jl}\hat{h}_{ij} \ . \tag{2.6}$$

A second approximation, denoted $\tilde{H}^1$, goes one step further and replaces $\hat{h}_{ij}$ by $\hat{h}_i\hat{\sigma}_j^2$ for $i \neq j$:

$$\begin{cases} \tilde{H}^1_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik}\delta_{jl}\hat{h}_i\hat{\sigma}_j^2 \text{ , for } i \neq j \\ \tilde{H}^1_{iiii} = 1 + \hat{h}_{ii} \end{cases} . \tag{2.7}$$

Those two approximations are illustrated on Fig 2.1. A key feature is that both approximations are *block diagonal* with blocks of size 2×2 and 1's on the off diagonal of those blocks: denoting $\tilde{H}$ for either $\tilde{H}^1$ or $\tilde{H}^2$, for $i \neq j$, the only non-zero coefficients in $\tilde{H}_{ijkl}$ are for $(k,l) = (i,j)$ or $(k,l) = (j,i)$, and $\tilde{H}_{ijji} = 1$.

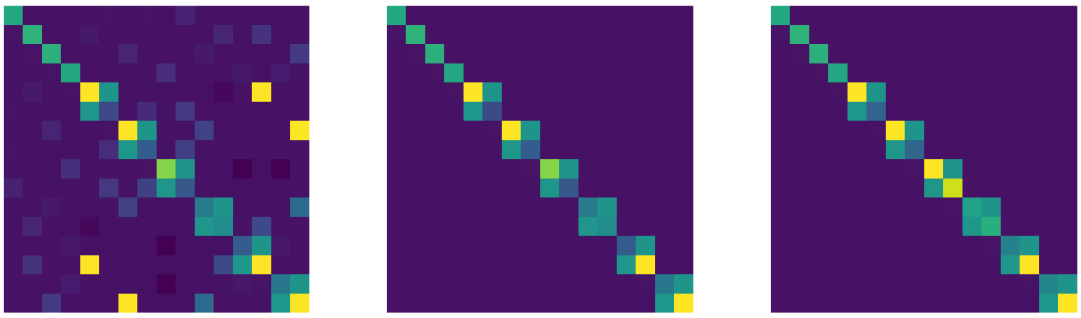


Figure 2.1 – The Hessian (left), its $\tilde{H}^2$ approximation (middle) and its $\tilde{H}^1$ approximation (right) for a mixture of $p = 4$ sources. The fourth order tensors are reshaped into matrices of size $p^2 \times p^2$ for visualization purpose. The first p rows correspond to the terms ε_{ii}, the following are arranged by pairs: $\varepsilon_{1,2}, \varepsilon_{2,1}, \varepsilon_{1,3}, \varepsilon_{3,1} \cdots \varepsilon_{p-1,p}, \varepsilon_{p,p-1}$. This arrangement shows the block-diagonal structure of the approximations, the dark purple color corresponding to zero coefficients. $\tilde{H}^2$ is H stripped from any off-block coefficient, and $\tilde{H}^1$ slightly differs from $\tilde{H}^2$ on its diagonal.

When the signals are independent, $\hat{h}_{ijl} = \delta_{jl}\hat{h}_{ij} = \delta_{jl}\hat{h}_i\hat{\sigma}_j^2$ asymptotically in n for $i \neq j$. This, together with $\hat{h}_{iii} = \hat{h}_{ii}$, means that *the two approximations asymptotically match the true Hessian if the signals Y are independent.* In particular, if an iterative algorithm

Algorithm 2.1 Regularization procedure

Input : Eigenvalue threshold $\lambda_{min} > 0$, approximate Hessian $\tilde{H}$ ($\tilde{H}^1$ or $\tilde{H}^2$

for *Each pair* (i, j) **do**

  Compute λ_{ij} using (2.9) **if** $\lambda_{ij} < \lambda_{min}$ **then**

    Add $(\lambda_{min} - \lambda_{ij})I_2$ to the block (i, j) of $\tilde{H}$

Output: Regularized $\tilde{H}$

converges to a solution on a problem where the ICA mixture model holds, the Hessian approximations get very close to the true Hessian of the objective function.

Away from convergence or if the ICA mixture model does not hold, one cannot expect those approximations to be very accurate. This is why we use them only as a *preconditioners* for our algorithm. They enjoy two properties which are critical in that respect, being fast to compute and easy to invert. Indeed, computing $\tilde{H}^1$ or $\tilde{H}^2$ is less costly than computing H. Evaluating $\tilde{H}^2$ requires the computation of $\hat{h}_{ij}$ for all (i, j), which is of order $\Theta(p^2 \times n)$. Obtaining $\tilde{H}^1$ is even faster, because it only requires the computation of the $\hat{h}_i$ and $\hat{\sigma}_i$, that is, $\Theta(p \times n)$.

Furthermore, the $\tilde{H}$ approximations are not only sparse: their block diagonal structure also allows $\tilde{H}^{-1}G$ to be computed quickly in closed-form. Indeed, defining $a_{ij} = \tilde{H}_{ijij}$, elementary linear algebra shows that

$$[\tilde{H}^{-1}G]_{ij} = \frac{a_{ji}G_{ij} - G_{ji}}{a_{ij}a_{ji} - 1} \qquad \text{for } i \neq j. \tag{2.8}$$

Hence, computing $\tilde{H}^{-1}G$ has complexity $\Theta(p^2)$.

2.1.4 Regularization of Hessian Approximations

Like the true Hessian, the Hessian approximations have no reason to be positive definite. This means that we have to set up a regularization procedure.

That can be done at little cost since the two Hessian approximations can be diagonalized in closed-form by diagonalizing each of the 2×2 blocks. The smallest eigenvalue for the block (i, j) is readily found to be:

$$\lambda_{ij} = \frac{1}{2}(a_{ij} + a_{ji} - \sqrt{(a_{ij} - a_{ji})^2 + 4}) \ , \tag{2.9}$$

with, again, $a_{ij} = \tilde{H}_{ijij}$, for either $\tilde{H} = \tilde{H}^1$ or $\tilde{H} = \tilde{H}^2$.

Based on this, we propose the simple regularization procedure detailed in Algorithm 2.1: the blocks with positive and large enough eigenvalues are left untouched, while the other blocks have their spectrum shifted so that their smallest eigenvalue is equal to a prescribed minimum value λ_{min}.

2.1.5 Preconditioned ICA for Real Data

Quasi-Newton methods attempt to estimate the local curvature of the objective function without explicitly computing its Hessian [Nocedal and Wright, 1999]. Indeed, popular methods such as DFP [Davidon, 1991, Fletcher, 1970, Fletcher and Powell, 1963] or BFGS [Broyden, 1970, Fletcher, 1970, Goldfarb, 1970, Shanno, 1970] build an approximation of the Hessian using solely function and gradient evaluations performed during optimization.

Algorithm 2.2 Preconditioned L-BFGS. Purple lines correspond to the distinctive steps of the algorithm.

Input : Mixed signals X, initial unmixing matrix W_0, memory size m, number of iterations K

for k=0,1,...,K **do**

  Set $Y = W_k X$ Compute the relative gradient G_k using (2.3) Compute the Hessian approximation $\tilde{H}_k$ using (2.6) or (2.7) Regularize $\tilde{H}_k$ using algorithm 2.1 Compute the search direction $p_k = -(\tilde{H}_k^m)^{-1} G_k$ using L-BFGS formula in algorithm 2.3 Compute the step length α_k using a line search Set $W_{k+1} = (I + \alpha_k p_k) W_k$

Output: Y, W_k

Algorithm 2.3 Two loops recursion for L-BFGS using a preconditioner

Input : Current gradient G_k, Hessian approximation $\tilde{H}_k$, previous s_i, y_i, ρ_i $\forall i \in \{k-m, \ldots, k-1\}$.

Set $q = -G_k$ **for** i=k-1,...,k-m **do**

  Compute $a_i = \rho_i \langle s_i, q \rangle$ Set $q = q - a_i y_i$

Set $r = \tilde{H}_k^{-1} q$ **for** i=k-m,...,k-1 **do**

  Compute $\beta = \rho_i \langle y_i, r \rangle$ Set $r = r + s_i(a_i - \beta)$

Output: $r = p_k$

The popular L-BFGS [Byrd et al., 1995] algorithm is used in many practical applications and obtains good results on a wide variety of problems. Rather than storing all the updates of the Hessian approximation leading to a dense matrix potentially too big to fit in memory like BFGS does, L-BFGS only stores the last m updates and then relies on a recursive inversion algorithm. The integer m is referred to as the memory parameter. The algorithm starts from an initial guess for the Hessian which is easily invertible, and builds on it by adding low rank updates from the m previous iterates and gradient evaluations. For lack of a better choice, standard L-BFGS uses a multiple of the identity matrix for the initial Hessian guess. However, if some Hessian approximation is available, it could be used instead. This is tantamount to preconditioning the problem with the said Hessian approximation [Jiang et al., 2004].

The Hessian approximations $\tilde{H}$ provide us with a very effective preconditioning, as shown below in Sec. 2.1.7, resulting in the 'Preconditioned ICA for Real Data' (Picard) algorithm. Picard exploits the Hessian approximations to initialize the recursive formula of L-BFGS. It is summarized in algorithms 2.2 and 2.3. We use the same notations as in [Nocedal and Wright, 1999]: $y_i = G_i - G_{i-1}$, $s_i = a_i p_i$ (this is the "relative" update of the unmixing matrix between two iterations) and $\rho_i = 1/\langle s_i, y_i \rangle$. As in standard L-BFGS algorithm, the search direction p_k is computed using recursive algorithm with two *for* loops, however the initial guess for the Hessian is here set to $\tilde{H}_k$.

Line search

The algorithm relies on a line search procedure which aims at finding a good step size α at each iteration. In theory, the line search procedure has to enforce Wolfe conditions [Wolfe, 1969, Nocedal and Wright, 1999] in order to guarantee convergence. The line search procedure proposed by Moré and Thuente [Moré and Thuente, 1994] is generally considered to be an efficient way to enforce such conditions. It is based upon cubic interpolation of the objective function in the direction of interest. Yet, for each

candidate step size, one must compute the values of the objective function and of the gradient, which can be costly.

A simpler line search strategy is backtracking. If, for $\alpha = 1$, the objective function is decreased, then that value is retained, otherwise the step size is divided by a factor of 2 and the process is repeated. This method only requires one evaluation of the likelihood at each step size, but it does not enforce Wolfe conditions.

In practice, backtracking is stopped when α becomes too small, which is an indication that the objective function has a pathological behavior in the search direction, since we rather expect values of the order of the "Newton value" $\alpha = 1$. In the case of too many backtracking steps, resulting in too small a step size, the algorithm would not move much, and might get stuck for a long time in that problematic zone. Therefore, after a fixed number of failed backtracking step, the L-BFGS descent direction is deemed inefficient and we fall back to descending along the relative gradient direction, and reset the memory (we found that this happens quite infrequently in the experiments). A similar restarting technique is presented in [Li and Fukushima, 2001].

If the line-search enforced Wolfe conditions, Picard would provably converge, as it is a quasi-Newton method with an Hessian approximation of spectrum bounded below by $\lambda_{\min}$ (c.f. Zoutendijk's conditions: [Nocedal and Wright, 1999], theorem 3.2.). However, we found that the backtracking line-search is faster in practice, and we never witnessed any case of non-convergence in the experiments performed in Section 2.1.7.

2.1.6 Related work

We compare our approach to the algorithms mentioned in Section 2.1.1. Some classical ICA algorithms such as FastICA [Hyvärinen, 1999], JADE [Cardoso and Souloumiac, 1993] or Kernel ICA [Bach and Jordan, 2002] are not included in the comparison because they do not optimize the same criterion.

Gradient descent

The gradient is readily available and directly gives an update rule for a gradient descent algorithm:

$$W \leftarrow \big(I - \alpha G\big)\, W \quad , \tag{2.10}$$

where $\alpha > 0$ is a step size found by line search or an annealing policy. In the experiments, we used an oracle line-search: at each step, we find a very good step size using a costly line-search, but do not take into account the time taken, as if the sequence of best step sizes were readily available. This algorithm is referred to as "Oracle gradient descent".

Infomax

We now give a brief explanation on how the Infomax [Bell and Sejnowski, 1995] algorithm actually runs. It is a stochastic version of rule (2.10): at each iteration of the algorithm, a relative gradient G' is computed from a 'mini-batch' of $n' \ll n$ randomly selected samples and a relative update $W \leftarrow (I - \alpha G')W$ is performed. The choice of relative gradient was proposed in [Amari et al., 1996].

The stochasticity of Infomax has benefits and drawbacks (see [Bottou et al., 2016] for a thorough review on the subject). On the good side, stochasticity accelerates the first few passes on the full data because the objective starts decreasing after only one mini

batch has been used, while for a full batch algorithm like the one presented above, it takes a full pass on the whole data to start making progress. Furthermore, if the number of samples is very large, computing the gradient using the whole dataset might be too costly, and then resorting to stochastic techniques is one way of coping with the issue.

Stochasticity, however, also comes with some disadvantages. The first one is that a plain stochastic gradient method with fixed batch size needs a very careful annealing policy for the learning rate to converge to a local minimum of the objective function. In practice, across iterations, the true gradient computed with the full set will not go to 0, but instead will reach a plateau.

This is directly linked to the choice of the step size. If it is too small the algorithm will not make much progress, and if it is too large, the algorithm will become unstable. In fact, the level of the plateau reached by the gradient is proportional to the step size [Bottou et al., 2016]. Line search techniques are also unpractical, because one has only access to noisy realizations of the gradient and of the objective if one works only on a mini-batch of samples. In practice, the standard Infomax implementation (e.g. in EEGlab [Delorme and Makeig, 2004]) relies on heuristics. It starts from a given step size α_0, and decreases it by a factor ρ if the angle between two successive search directions is greater than some constant θ. That makes 3 parameters that have to be set correctly, which may be problematic [Montoya-Martínez et al., 2017].

Truncated Newton's method

As explained above, direct Newton's method is quite costly. The so-called truncated Newton method [Nocedal and Wright, 1999] manages to obtain directions similar to Newton's method at a fraction of the cost. The idea is to compute $H^{-1}G$ using the conjugate gradient method [Nocedal and Wright, 1999] which does not require the construction of H but only a mean of computing a Hessian-vector product HM. In the ICA problem, there is an efficient way to do so, a *Hessian free product*. Indeed, using expression (2.4) of the true Hessian, one finds:

$$(HM)_{ij} = \sum_{k,l} H_{ijkl} M_{kl} = M_{ji} + \hat{E}[\psi_i'(y_i) y_j \sum_l M_{il} y_l]$$

or in a matrix form:

$$HM = M^\top + \frac{1}{n}[\psi'(Y) \cdot (MY)]Y^\top$$

where $\cdot$ is the element-wise matrix product. This computation comes at roughly the same cost as a gradient evaluation, its complexity being $\Theta(p^2 \times n)$.

The idea of truncated Newton's method is that instead of perfectly computing $H^{-1}G$ using conjugate gradient, one can stop the iterations before termination, at a given error level, and still obtain a useful approximation of Newton's direction.

This method is applied to ICA in [Tillet et al., 2017] where the authors also use a stochastic framework with variable batch size, speeding up the algorithm during the first steps. We did not implement such a strategy in order to have a fairer comparison.

One way of incorporating Hessian approximations in this method (not implemented in [Tillet et al., 2017]) is to use them once again as preconditioners for the linear conjugate gradient. We found that this idea roughly halves the number of conjugate gradient iterations for a given error in solving $H^{-1}G$.

A difficulty arising with this method is the Hessian regularization. Because it avoids the computation of the Hessian, finding its smallest eigenvalue is not straightforward, and heuristics have to be used, like in [Tillet et al., 2017]. However, we do not want these hand tuned parameters to bias the algorithm comparison. Hence, in our implementation of the algorithm, we compute H and its smallest eigenvalue λ_m but we do not include the associated cost in the timing of the algorithm. Then, we regularize H by adding $-2\lambda_m Id$ to it if $\lambda_m < 0$.

These steps are summarized in Algorithm 2.4 in which the step marked with a $(*)$ is not counted in the final timing.

Algorithm 2.4 Truncated Newton's method. Purple lines correspond to the distinctive steps of the algorithm.

Input : Mixed signals X, initial unmixing matrix W_0, number of iterations K.

Set $Y = W_0 X$ **for** k=0,1,...,K **do**

  Compute the relative gradient G_k using (2.3) Compute the Hessian approximation $\tilde{H}_k$ using (2.6) or (2.7) $(*)$ Compute a regularization level λ for H_k Compute the search direction $p_k = -(H_k + \lambda I)^{-1} G_k$ by preconditioned conjugate gradient with regularized $\tilde{H}_k$ Set $W_{k+1} = (I + \alpha_k p_k) W_k$ (α_k set by line search) Set $Y \leftarrow (I + \alpha_k p_k) Y$

Output: Y, W_k

Simple Quasi-Newton method

The simplest way to take advantage of the Hessian approximations is to use them as replacement of H in Newton algorithm. The descent direction is then given by $-\tilde{H}^{-1} G$. We will refer to this as the simple quasi-Newton method, which is detailed in Algorithm 2.5. Note that any Hessian approximation can be used as long as it is guaranteed to be positive definite. This optimization algorithm is used in [Zibulevsky, 2003] with $\tilde{H}^2$ (however, the regularization technique differs from our implementation), and in [Palmer et al., 2012] with $\tilde{H}^1$.

In the experiments, we refer to this algorithm as "Simple quasi-Newton H2" and "Simple quasi-Newton H1" where we respectively use $\tilde{H}^2$ and $\tilde{H}^1$ as approximations.

Algorithm 2.5 Simple quasi-Newton. Purple lines correspond to the distinctive steps of the algorithm.

Input : Mixed signals X, initial unmixing matrix W_0, number of iterations K.

Set $Y = W_0 X$ **for** k=0,1,...,K **do**

  Compute the relative gradient G_k using (2.3) Compute the Hessian approximation $\tilde{H}_k$ using (2.6) or (2.7) Regularize $\tilde{H}_k$ using algorithm 2.1 Compute the search direction $p_k = -(\tilde{H}_k)^{-1} G_k$ Set $W_{k+1} = (I + \alpha_k p_k) W_k$ (α_k set by line search) Set $Y \leftarrow (I + \alpha_k p_k) Y$

Output: Y, W_k

Trust-region method

Another way to proceed is to use a trust-region algorithm [Nocedal and Wright, 1999], rather than a line-search strategy. It is the idea proposed in [Choi and Choi, 2007], where $\tilde{H}^2$ is used to build a local quadratic approximation of the objective, and then minimization is done with a trust region update. In the experiments, we denote this

algorithm by "Trust region ICA". For the experiments, we used a direct translation of the author's code in Python, which produces the same iterations as the original code.

2.1.7 Experiments

We went to great lengths to ensure a fair comparison of the above algorithms. We reimplemented each algorithm using the Python programming language. Since the most costly operations are by far those scaling with the number of samples (*i.e.* evaluations of the likelihood, score and its derivative, gradient, Hessian approximations and Hessian free products), we made sure that all implementations call the same functions, thereby ensuring that differences in convergence speed are caused by algorithmic differences rather than by details of implementation.

Our implementations of Picard, the simple quasi-Newton method, the truncated Newton method and the trust region method are available online[1]. A Matlab/Octave and Python package of Picard is also released at https://github.com/pierreablin/picard.

Experimental setup

All the following experiments have been performed on the same computer using only one core of an Intel Core i7-6600U @ 2.6 GHz. Computations are performed in double precision. For optimized numerical code we relied on Numpy [Van der Walt et al., 2011] using Intel MKL as the linear algebra backend library, and the numexpr package[2] to optimize CPU cache. It was particularly efficient in computing $\log\cosh(y_i(t)/2)$ and $\tanh(y_i(t)/2)$ $\forall i, t$.

For each ICA experiment, we keep track of the gradient infinite norm (defined as $\max_{ij}|G_{ij}|$) across time and across iterations. The algorithms are stopped if a certain predefined number of iterations is exceeded or if the gradient norm reaches a small threshold (typically 10^{-8}).

Each experiment is repeated a certain number of times to increase the robustness of the conclusions. The algorithms all start from the same initial point, and we end up with several gradient norm curves. On the convergence plots, we only display the median of these curves to maintain readability: half experiments finished faster and the other half finished slower than the plotted curve.

Besides the algorithms mentioned above, we have run the standard L-BFGS algorithm, and Picard using $\tilde{H}^1$ and $\tilde{H}^2$.

The parameters of each algorithm has been set by hand to provide the fastest convergence. Regarding Infomax, for each dataset, we tried several values for the parameters in a logarithmic grid for α_0 : $\alpha_0 = 10^{-6} \cdots 10^{-3}$, and a linear grid for ρ: $\rho \in \{0.1, 0.5, 0.9\}$. We took for θ the default value given in EEGlab: $\theta = 60°$. For the algorithms using the Hessian approximations, the regularization level is set to the same value $\lambda_{\min} = 0.01$. Finally, the defaults parameters of the trust region algorithm and of L-BFGS have been used. A thorough review of Picard's parameters is provided in Section 2.1.7.

[1] https://github.com/pierreablin/faster-ica
[2] https://github.com/pydata/numexpr

Preprocessing

A standard preprocessing for ICA is applied in all our experiments, as follows. Any given input matrix X is first mean-corrected by subtracting to each row its mean value. Next, the data are whitened by multiplication by a matrix inverse square root of the empirical covariance matrix $C = \frac{1}{n}XX^\top$. After whitening, the covariance matrix of the transformed signals is the identity matrix. In other words, the signals are decorrelated and scaled to unit variance.

Simulation study

In this first study, we present results obtained on synthetic data. The general setup is the following: we choose the number of sources p, the number of samples n and a probability density for each source. For each of the p densities, we draw n independent samples. Then, a random mixing matrix whose entries are normally distributed with zero mean and unit variance is created. The synthetic signals are obtained by multiplying the source signals by the mixing matrix and preprocessed as described in Section (2.1.7).

We repeat the experiments 100 times, changing for each run the seed generating the random signals.

We consider 3 different setups:

- *Experiment A*: $n = 10000$ independent samples of $p = 50$ independent sources. All sources are drawn with the same density $d(x) = \frac{1}{2}\exp(-|x|)$.
- *Experiment B*: $n = 10000$ independent samples of $p = 15$ independent sources. The 5 first sources have density $d(x) = \exp(-|x|)$, the 5 next sources are Gaussian, and the 5 last sources have density $d(x) \propto \exp(-|x^3|)$.
- *Experiment C*: $n = 5000$ independent samples of $p = 40$ independent sources. The ith source has density $d_i = \alpha_i\mathcal{N}(0, 1) + (1 - \alpha_i)\mathcal{N}(0, \sigma^2)$ where $\sigma = 0.1$ and α_i is a sequence of linearly spaced values between $\alpha_1 = 0.5$ and $\alpha_n = 1$.

In experiment A, the ICA assumption holds perfectly, and each source has a super Gaussian density, for which the choice $\psi = \tanh(\cdot/2)$ is appropriate. In experiment B, the first 5 sources can be recovered by the algorithms for the same reason. However, the next 5 sources cannot because they are Gaussian, and the last 5 sources cannot be recovered either because they are sub-Gaussian. Finally, in experiment C, the mixture is identifiable but, because of the limited number of samples, the most Gaussian sources cannot be distinguished from an actual Gaussian signal. The results of the three experiments are shown in Figure 2.2.

Experiments on EEG data

Our algorithms were also run on 13 publicly available[3] EEG datasets [Delorme et al., 2012]. Each recording contains $p = 71$ signals, and has been down-sampled by 4 for a final length of $n \simeq 75000$ samples. EEG measures the changes of electric potential induced by brain activity. For such data, the ICA assumption does not perfectly hold. In addition, brain signals are contaminated by a lot of noise and artifacts. Still, it has been shown that ICA succeeds at extracting meaningful and biologically plausible

[3] https://sccn.ucsd.edu/wiki/BSSComparison

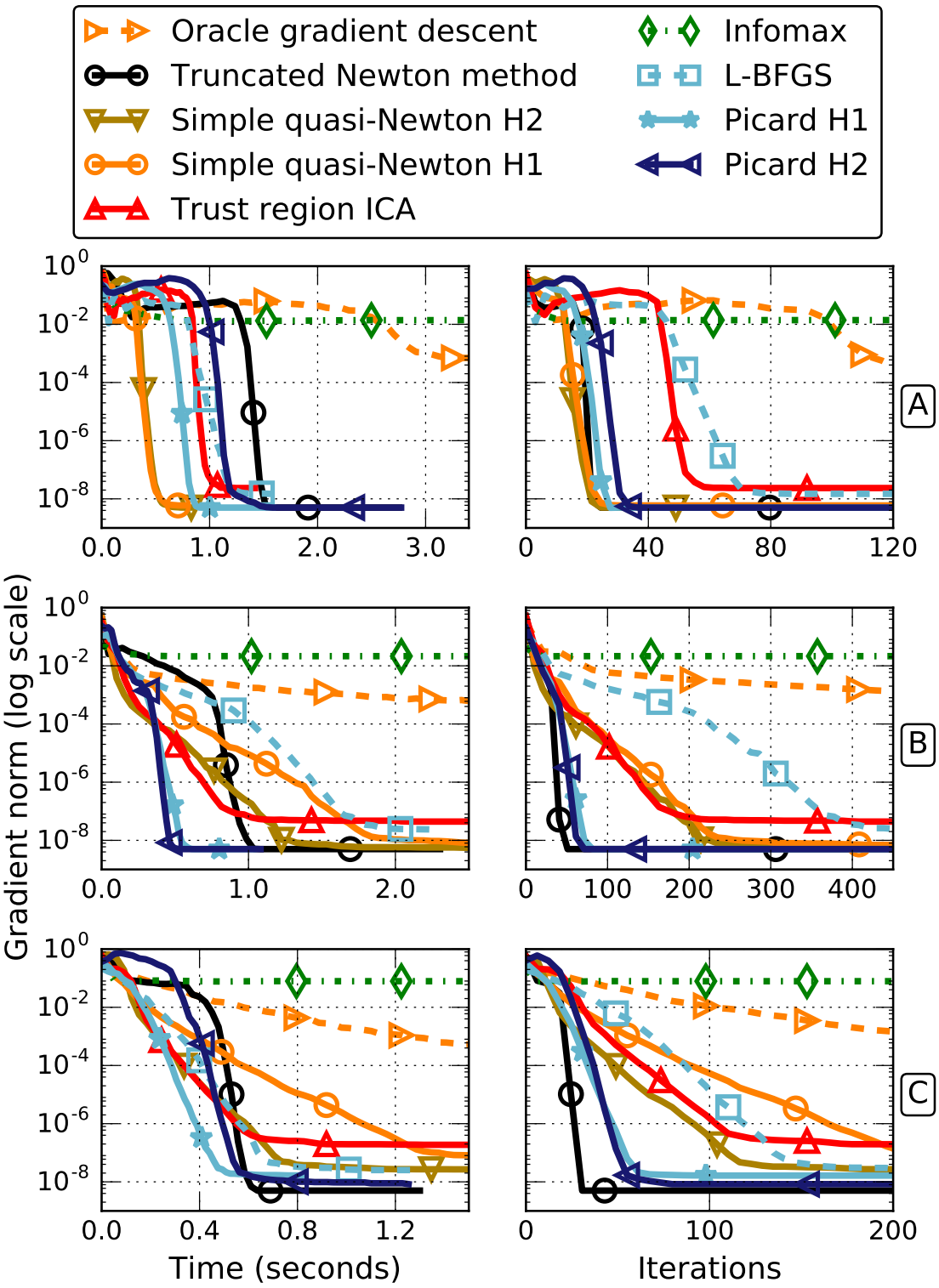


Figure 2.2 – Comparison of the optimization algorithms on three synthetic experiments. Top: experiment A, middle: experiment B, bottom: experiment C. Left: infinity norm of the relative gradient as a function of CPU time, right: same as a function of iterations (pass on the full set for Infomax). Solid lines correspond to algorithms informed of the approximate Hessians, dashed lines are their standard counterparts.

sources from these mixtures [Jung et al., 1997, Makeig et al., 1997, Delorme et al., 2012]. Results are displayed on the top row of Figure 2.3.

Experiments on fMRI data

For those experiments, we used preprocessed fMRI data from the ADHD-200 consortium [Consortium et al., 2012]. We perform group ICA using the CanICA framework [Varoquaux et al., 2010] in Nilearn [Abraham et al., 2014]. From the fMRI data of several patients, CanICA builds a signal matrix to which classical ICA is applied. We use that matrix to benchmark the algorithms. The problem is of size $p = 40$ and $n \simeq 60000$. See middle row of Figure 2.3.

Experiments on natural images

Given a grayscale image, we extract n square patches (contiguous squares of pixels) of size (s, s). Each patch is vectorized, yielding an $s^2 \times n$ data matrix. One may compute an ICA of this data set and see the columns of the mixing matrix W^{-1} as features, or dictionary atoms, learned from random patches.

The ICA algorithms are run on a set of 100 natural images of open country[4] [Oliva and Torralba, 2001], using $p = 30000$ patches of side 8×8 pixels, resulting in a 64×30000 data matrix. The patches are all centered and scaled so that their mean and variance equal 0 and 1 respectively before whitening as in Section 2.1.7. Results are shown at the bottom of Figure 2.3.

Comparison of the output objective function values

Finally, we compare the values of the final objective function $\mathcal{L}(W^*)$ where W^* is the output of different algorithms. On EEG data, the simple quasi-Newton with $\tilde{H}^1$, the trust region ICA, Picard and Infomax are run from 50 different random initializations. Figure 2.4 displays the histogram of the final objective function values for the different algorithms. Picard, simple quasi-Newton and trust-region ICA give similar histograms, which means that they end up on average in the same local minima. Finally, we can see that the final objective values returned by Infomax are higher than for the other algorithms, illustrating once again that it fails to perfectly converge. The same experiment is repeated on the 13 EEG dataset and leads to similar conclusions.

Discussion

On the first synthetic experiment, where the ICA mixture model holds, second order algorithms are all seen to perform well, converging in a handful of iterations. For this problem, the fastest algorithms are the simple quasi-Newton methods, which means that Picard does not improve significantly over the Hessian approximations $\tilde{H}^1$ or $\tilde{H}^2$. This is expected since the Hessian approximations are accurate when the ICA mixture model holds.

On the two other simulations, the ICA model is not identifiable because of the Gaussian signals. First order methods perform poorly. We can observe that for algorithms relying only on the Hessian approximations (simple quasi-Newton and trust-region ICA), the convergence speed is reduced. On the contrary, Picard and truncated Newton manage

[4] http://cvcl.mit.edu/database.htm

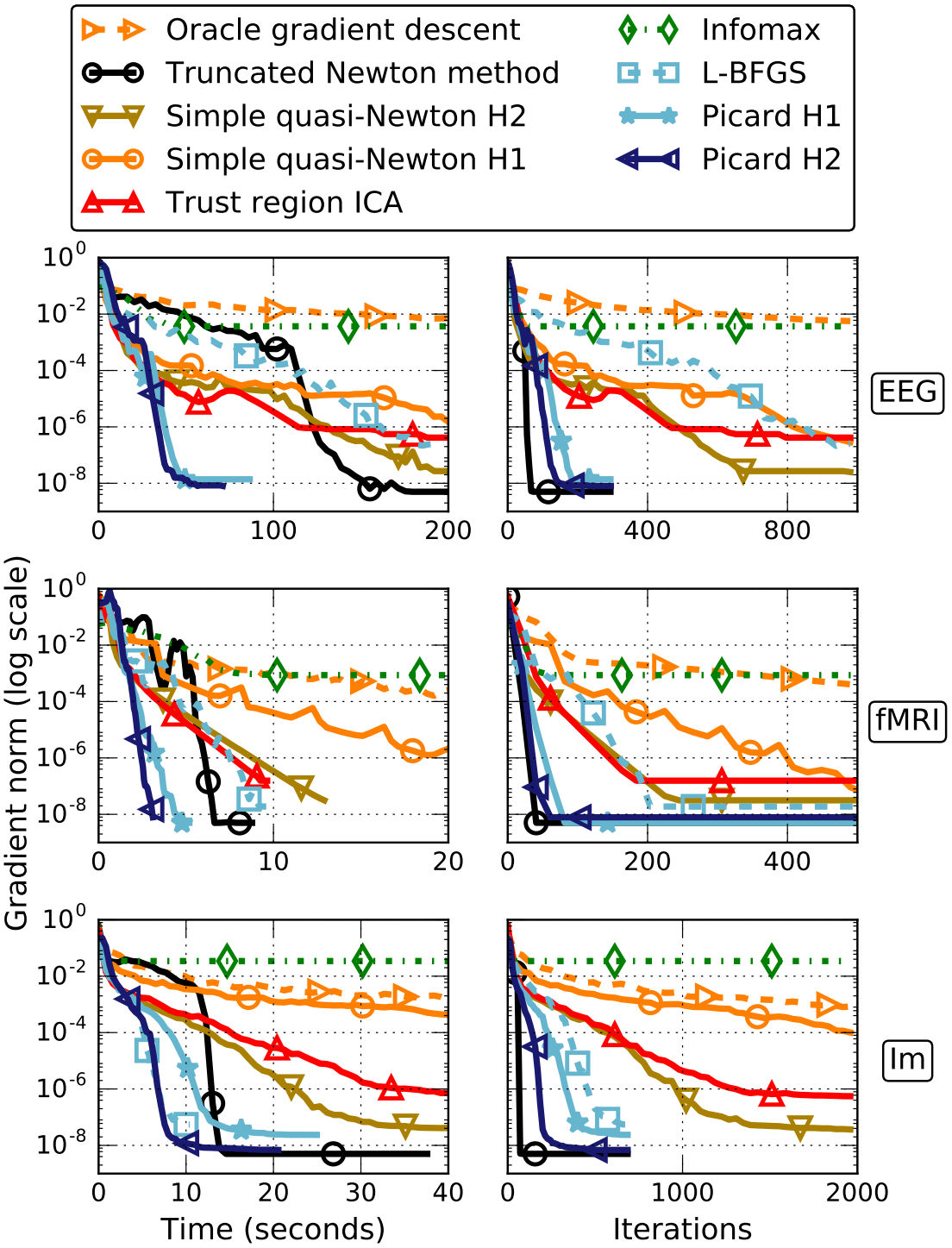


Figure 2.3 – Comparison of the optimization algorithms on real data. Top: EEG dataset. Middle: fMRI dataset. Bottom: Image patches dataset. Left: infinity norm of the relative gradient w.r.t. CPU time, right: same w.r.t. iterations (pass on the full data for Infomax). Solid lines correspond to algorithms informed of the approximate Hessian, dashed lines are their standard counterparts.

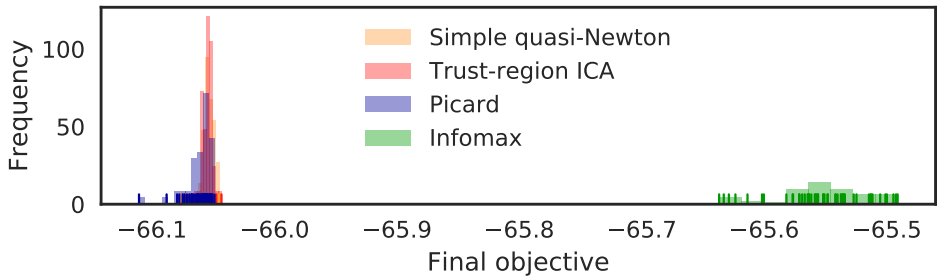


Figure 2.4 – Histogram of the final negative log-likelihood obtained after running different algorithms on an EEG dataset. Algorithms start from 50 random white initializations. Simple quasi-Newton, Trust region ICA and Picard yield similar histograms, while Infomax does not minimize the objective as accurately.

to keep a very quick convergence. On those synthetic problems, it is not clear whether or not the greater accuracy of $\tilde{H}^2$ over $\tilde{H}^1$ justifies the added computation cost.

On EEG and fMRI data, Picard still converges quickly, in a fraction of the time taken by the other algorithms. For this problem, using $\tilde{H}^2$ for preconditioning leads to faster convergence than $\tilde{H}^1$. The results are even more striking on images, where Picard, standard L-BFGS and truncated Newton converge in a few seconds while the other algorithms show a very slow linear convergence pattern.

On all experiments, truncated Newton's method converges in fewer iterations than Picard. This happens because it follows a direction very close to Newton's true direction, which is the direction second order algorithms try to mimic when the current iterate is close to the optimum. However, if we compare algorithms in terms of time, the picture is different: the reduced number of iterations does not make up for the added cost compared to Picard.

Complexity comparison of truncated Newton and preconditioned L-BFGS

Truncated Newton's method uses the full information about the curvature. In our experiments, we observe that while this method converges in fewer iterations than Picard, it is slower in terms of CPU time. The speed of truncated Newton depends on many parameters (stopping policy for the conjugate gradient, regularization of H and of $\tilde{H}$), so we propose a complexity comparison of this algorithm and Picard, to understand if the former might sometimes be faster than the latter.

Operations carried by the algorithms fall into two categories. First, there are operations that do not scale with the number of samples n, but only with the number of sources p. Regularizing the Hessian, computing $\tilde{H}^{-1}G$ and the L-BFGS inner loop are such operations. The remaining operations scale linearly with n. Computing the score, its derivative, or evaluating the likelihood are all $\Theta(p \times n)$ operations. The most costly operations are in $\Theta(p^2 \times n)$. They are: computing the gradient, computing $\tilde{H}^2$, and for the truncated Newton method, computing a Hessian-free product.

For the following study, let us reason in the context where n is large in front of p^2, as it is the case for most real data applications. In that context, we do not count operations not scaling with n. This is a reasonable assumption on real datasets: on the EEG problem, these operations make for less than 1% of the total timing for Picard.

To keep the analysis simple, let us also assume that the operations in $\Theta(p \times n)$ are negligible in front of those in $\Theta(p^2 \times n)$. When computing a gradient, the coefficients of $\tilde{H}^2$ or a Hessian free product, the costly operation in $\Theta(p^2 \times n)$ is numerically the same: it is the computation of a matrix product of the form $Y_1 Y_2^\top$ where Y_1 and Y_2 have the same shape as Y. With that in mind, we assume that each of these operations take the same time, t_G.

In order to produce a descent direction, Picard only needs the current gradient and Hessian approximations; the remaining operations do not scale with n. This means that each descent direction takes about $2 \times t_G$ to be found. This complexity is exactly the same as the simple quasi-Newton method. On the other hand, truncated Newton requires the two same operations, as well as one Hessian-free product for each inner iteration of the conjugate gradient. If we denote by N_{cg} the number of inner loops for the conjugate gradient, we find that truncated Newton's method takes $(2 + N_{cg}) \times t_G$ to find the descent direction.

Now, in our experiments, we can see that truncated Newton converges in about half as many iterations as Picard. Hence, for truncated Newton to be competitive, each of its iterations should take no longer than twice a Picard iteration. That would require $N_{cg} \leq 2$ but in practice, we observed that many more conjugate gradient iterations are needed (usually more than 10) to provide a satisfying Newton direction. On the other hand, if the conjugate gradient algorithm is allowed to perform only $N_{cg} = 2$ inner loops at each iteration, it results in a direction which is far from Newton's direction, drastically increasing the number of iterations required for convergence.

This analysis leads us to think that the truncated Newton's method as described in Section 2.1.6 cannot be faster than Picard.

Study of the control parameters of Picard

Picard has four control parameters: a binary choice between two Hessian approximations, the memory size m for L-BFGS, the number of tries allowed for the backtracking line-search n_{ls} and a regularization constant λ_{min}.

Our experiments indicate that $\tilde{H}^2$ is overall a better preconditioner for the algorithm, although the difference with $\tilde{H}^1$ can be small.

Through experiments, we found that the memory size had barely no effect in the range $3 \leq m \leq 15$. For a smaller value of m, the algorithm does not have enough information to build a good Hessian approximation. If m is too large, the Hessian built by the algorithm is biased by the landscape explored too far in the past.

The number of tries for the line-search has a tangible effect on convergence speed. Similarly, the optimal regularization constant depends on the difficulty of the problem. However, on the variety of different signals processed in our experiments (synthetic, EEG, fMRI and image), we used the same parameters $m = 7$, $n_{ls} = 10$ and $\lambda_{min} = 10^{-2}$. As reported, those values yielded uniformly good performance and we never witness any case of non-convergence.

2.1.8 Conclusion

While ICA is massively used across scientific domains, computation time for inference can be a bottleneck in many applications. The purpose of this work to design a fast and accurate algorithm for maximum-likelihood ICA.

For this optimization problem, there are computationally cheap approximations of the Hessian. This leads to simple quasi-Newton algorithms that have a cost per iteration only twice as high as a gradient descent, while offering far better descent directions. Yet, such approximations can be far from the true Hessian on real datasets. As a consequence, practical convergence is not as fast as one can expect from a second order method. Another approach is to use a truncated Newton algorithm, which yields directions closer to Newton's algorithm, but at a much higher cost per iteration.

In this work, we introduce the Preconditioned ICA for Real Data (Picard) algorithm, which combines both ideas. We use the Hessian approximations as preconditioners for the L-BFGS method. The algorithm refines the Hessian approximations to better take into account the true curvature. The cost per iteration of Picard is similar to the simple quasi-Newton methods, while providing far better descent directions. This was demonstrated, through careful implementation of various literature methods and extensive experiments over synthetic, EEG, fMRI and image data, where we showed clear gains in running time compared to the state-of-the-art.

2.2 Extension to the orthogonal constraint

This section presents the work published in:

Pierre Ablin, Jean-François Cardoso, and Alexandre Gramfort. Faster ICA under orthogonal constraint. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018c

Only minor changes have been made: shorter introduction and notation harmonization.

2.2.1 Introduction

Given an $p \times n$ data matrix X made of p signals of length n, an ICA algorithm finds an $p \times p$ 'unmixing matrix' W such that the rows of $Y = WX$ are 'as independent as possible'. An important class of ICA methods further constrains the rows of Y to be uncorrelated. Assuming zero-mean and unit variance signals, that is:

$$\frac{1}{n} Y Y^\top = I_p \ . \tag{2.11}$$

The 'whiteness constraint' (2.11) is satisfied if, for instance,

$$W = OW_0 \qquad W_0 = (\tfrac{1}{n} X X^\top)^{-1/2} \tag{2.12}$$

and the matrix O is constrained to be orthogonal: $OO^\top = I_p$. For this reason, ICA algorithms which enforce signal decorrelation by proceeding as suggested by (2.12) — whitening followed by rotation— can be called 'orthogonal methods'.

The orthogonal approach is followed by many ICA algorithms. Among them, FastICA [Hyvärinen, 1999] stands out by its simplicity, good scaling and a built-in ability to deal with both sub-Gaussian and super-Gaussian components. It also enjoys an impressive convergence speed when applied to data which are actual mixture of independent components [Oja and Yuan, 2006, Shen et al., 2008]. However, real data can rarely be accurately modeled as mixtures of independent components. In that case, the convergence of FastICA may be impaired or even not happen at all. Such problems have already been noted in the literature [Zarzoso et al., 2006, Zarzoso and Comon, 2009]. In this section, we extend Picard to an orthogonal version, dubbed Picard-O, which solves the same problem as FastICA, but faster on real data.

In section 2.2.2, the non-Gaussian likelihood for ICA is studied in the orthogonal case yielding the Picard Orthogonal (Picard-O) algorithm in section 2.2.3. Section 2.2.4 connects our approach to FastICA: Picard-O converges toward the fixed points of FastICA, yet faster thanks to a better approximation of the Hessian matrix. This is illustrated through extensive experiments on four types of data in section 2.2.5.

2.2.2 Likelihood under whiteness constraint

Once again, the negative log-likelihood writes:

$$\mathcal{L}(W) = -\log|W| - \hat{E}\left[\sum_{i=1}^p \log(d_i(y_i(t))\right] \ , \tag{2.13}$$

where $\hat{E}$ denotes the empirical mean (sample average) and where, implicitly, $[y_1, \cdots, y_p]^\top = Y = WX$.

We consider maximum likelihood estimation under the whiteness constraint (2.11). By (2.12) and (2.13), this is equivalent to minimizing $\mathcal{L}(OW_0)$ with respect to the orthogonal matrix O. To do so, we propose an iterative algorithm. A given iterate $W_k = O_k W_0$ is updated by replacing O_k by a more likely orthonormal matrix O_{k+1} in its neighborhood. Following classical results of differential geometry over the orthogonal group [Bertram, 2008], we parameterize that neighborhood by expressing O_{k+1} as $O_{k+1} = e^{\mathcal{E}} O_k$ where $\mathcal{E}$ is a (small) $p \times p$ skew-symmetric matrix: $\mathcal{E}^\top = -\mathcal{E}$.

The second-order Taylor expansion of $\mathcal{L}(e^{\mathcal{E}} W)$ reads:

$$\mathcal{L}(e^{\mathcal{E}} W) = \mathcal{L}(W) + \langle G, \mathcal{E} \rangle + \frac{1}{2} \langle \mathcal{E}, H\mathcal{E} \rangle + \mathcal{O}(||\mathcal{E}||^3). \tag{2.14}$$

We obtain:

$$G_{ij} = \hat{E}[\psi_i(y_i) y_j] - \delta_{ij} \ , \tag{2.15}$$

$$H_{ijkl} = \delta_{il}\delta_{jk} \hat{E}[\psi_i(y_i) y_i] + \delta_{ik}\, \hat{E}[\psi_i'(y_i) y_j y_l] \ . \tag{2.16}$$

where $\psi_i = -\frac{d_i'}{d_i}$ is called the *score function*. Note that the Gradient is the same as before, but that there is an additional term in the Hessian due to the different parametrization.

This Hessian (2.16) is quite costly to compute, but a simple approximation is obtained using the form that it would take for large n and independent signals. In that limit, one has:

$$\hat{E}[\psi_i'(y_i) y_j y_l] \approx \delta_{jl}\, \hat{E}[\psi_i'(y_i)] \hat{E}[y_j^2] \quad \text{for} \quad i \neq j, \tag{2.17}$$

hence the approximate Hessian (recall $\hat{E}[y_j^2] = 1$):

$$\tilde{H}_{ijkl} = \delta_{il}\delta_{jk} \hat{E}[\psi_i(y_i) y_i] + \delta_{ik}\delta_{jl}\, \hat{E}[\psi_i'(y_i)] \ \text{ if } i \neq j. \tag{2.18}$$

Plugging this expression in the 2nd-order expansion (2.14) and expressing the result as a function of the $p(p-1)/2$ free parameters $\{\mathcal{E}_{ij},\ 1 \leq i < j \leq p\}$ of a skew-symmetric matrix $\mathcal{E}$ yields after simple calculations:

$$\langle G, \mathcal{E} \rangle + \frac{1}{2} \langle \mathcal{E}, \tilde{H}\mathcal{E} \rangle = \sum_{i<j} (G_{ij} - G_{ji})\, \mathcal{E}_{ij} + \frac{\hat{\kappa}_i + \hat{\kappa}_j}{2}\, \mathcal{E}_{ij}^2 \tag{2.19}$$

where we have defined the non-linear moments:

$$\hat{\kappa}_i = \hat{E}[\psi_i(y_i) y_i] - \hat{E}[\psi_i'(y_i)] \ . \tag{2.20}$$

If $\hat{\kappa}_i + \hat{\kappa}_j > 0$ (this assumption will be enforced in the next section), the form (2.19) is minimized for $\mathcal{E}_{ij} = -(G_{ij} - G_{ji})/(\hat{\kappa}_i + \hat{\kappa}_j)$: the resulting quasi-Newton step would be

$$W_{k+1} = e^D W_k \quad \text{for} \quad D_{ij} = -\frac{2}{\hat{\kappa}_i + \hat{\kappa}_j} \frac{G_{ij} - G_{ji}}{2}. \tag{2.21}$$

That observation forms the keystone of the new orthogonal algorithm described in the next section.

2.2.3 The Picard-O algorithm

As we shall see in Sec. 2.2.4, update (2.21) is essentially the behavior of FastICA near convergence. Hence, one can improve on FastICA by using a more accurate Hessian approximation. Using the exact form (2.16) would be quite costly for large data sets. Instead, following the same strategy as in Sec. 2.1, we base our algorithm on the L-BFGS method (which learns the actual curvature of the problem from the data themselves) using approximation (2.19) only as a *pre-conditioner*.

L-BFGS and its pre-conditioning: The L-BFGS method keeps track of the m previous values of the (skew-symmetric) relative moves $\mathcal{E}_k$ and gradient differences $\Delta_k = (G_k - G_k^\top)/2 - (G_{k-1} - G_{k-1}^\top)/2$ and of auxiliary quantities $\rho_k = \langle \mathcal{E}_k, \Delta_k \rangle$. It returns a descent direction by running through one backward loop and one forward loop. It can be pre-conditioned by inserting a Hessian approximation in between the two loops as summarized in algorithm 2.6.

Stability: If the ICA mixture model holds, the sources should constitute a local minimum of $\mathcal{L}$. According to (2.19), that happens if $\hat{\kappa}_i + \hat{\kappa}_j > 0$ for all $i < j$ (see also [Cardoso, 1998a]). We enforce that property by taking, at each iteration:

$$\psi_i(\cdot) = \text{sign}(k_i)\psi(\cdot) \tag{2.22}$$

where $k_i = \hat{E}[\psi(y_i)y_i] - \hat{E}[\psi'(y_i)]$ and ψ is a fixed non-linearity (a typical choice is $\psi(u) = \tanh(u)$). This is very similar to the technique in extended Infomax [Lee et al., 1999]. It enforces $\hat{\kappa}_i = |k_i| > 0$, and the positivity of $\tilde{H}$. In practice, if for any signal i the sign of k_i changes from one iteration to the next, L-BFGS's memory is flushed.

Regularization: The switching technique guarantees that the Hessian approximation is positive, but one may be wary of very small values of $\hat{\kappa}_i + \hat{\kappa}_j$. Hence, the pre-conditioner uses a floor: $\max((\hat{\kappa}_i + \hat{\kappa}_j)/2, \kappa_{\min})$ for some small positive value of $\kappa_{\min}$ (typically $\kappa_{\min} \simeq 10^{-2}$).

Line search: A backtracking line search helps convergence. Using the search direction D_k returned by L-BFGS, and starting from a step size $\alpha = 1$, if $\mathcal{L}(\exp(\alpha D_k)W_k) < \mathcal{L}(W_k)$, then set $\mathcal{E}_{k+1} = \alpha D_k$ and $W_{k+1} = \exp(\mathcal{E}_{k+1})W_k$, otherwise divide α by 2 and repeat.

Stopping: The stopping criterion is $||G - G^\top|| < \varepsilon$ where ε is a small tolerance constant.

Combining these ideas, we obtain the Preconditioned Independent Component Analysis for Real Data-Orthogonal (Picard-O) algorithm, summarized in table 2.7.

The Python code for Picard-O is available online at https://github.com/pierreablin/picard.

2.2.4 Link with FastICA

This section briefly examines the connections between Picard-O and symmetric FastICA [Hyvärinen, 1999]. In particular, we show that both methods essentially share the same solutions and that the behavior of FastICA is similar to a quasi-Newton method.

Recall that FastICA is based on an $p \times p$ matrix $C(Y)$ matrix defined entry-wise by:

$$C_{ij}(Y) = \hat{E}[\psi_i(y_i)y_j] - \delta_{ij}\hat{E}[\psi_i'(y_i)] \ . \tag{2.23}$$

Algorithm 2.6 Two-loop recursion L-BFGS formula

Input : Current gradient G_k, moments $\hat{\kappa}_i$, previous $\mathcal{E}_l$, Δ_l, ρ_l $\forall l \in \{k-m, \dots, k-1\}$.

Set $Q = -(G_k - G_k^\top)/2$ **for** *l=k-1,...,k-m* **do**

  Compute $a_l = \rho_l \langle \mathcal{E}_l, Q \rangle$ Set $Q = Q - a_l \Delta_i$

Compute D as $D_{ij} = Q_{ij} / \max\left(\frac{\hat{\kappa}_i + \hat{\kappa}_j}{2}, \kappa_{\min}\right)$ **for** *l=k-m,...,k-1* **do**

  Compute $\beta = \rho_l \langle \Delta_l, D \rangle$ Set $D = D + \mathcal{E}_l(a_l - \beta)$

Output: Descent direction D

Algorithm 2.7 The Picard-O algorithm

Input : Initial signals X, number of iterations K

Sphering: compute W_0 by (2.11) and set $Y = W_0 X$ **for** $k = 0 \cdots K$ **do**

  Compute the signs $\text{sign}(k_i)$ Flush the memory if the sign of any source has changed Compute the gradient G_k Compute search direction D_k using algorithm 2.6 Compute the step size α_k by line search Set $W_{k+1} = \exp(\alpha_k D_k) W_k$ and $Y = W_{k+1} X$ Update the memory;

Output: Unmixed signals Y, unmixing matrix W_k

The symmetric FastICA algorithm, starting from white signals Y, can be seen as iterating $Y \leftarrow C_w(Y)Y$ until convergence, where $C_w(Y)$ is the orthogonal matrix computed as

$$C_w(Y) = (CC^\top)^{-\frac{1}{2}} C \ . \tag{2.24}$$

In the case of a fixed score function, a sign-flipping phenomenon appears leading to the following definition of a fixed point: Y is a fixed point of FastICA if $C_w(Y)$ is a diagonal matrix of ± 1 [Wei, 2015]. This behavior can be fixed by changing the score functions as in (2.22). It is not hard to see that, if ψ is an odd function, such a modified version has the same trajectories as the fixed score version (up to to irrelevant sign flips), and that the fixed points of the original algorithm now all verify $C_w(Y) = I_p$.

Stationary points : We first relate the fixed points of FastICA (or rather the sign-adjusted version described above) to the stationary points of Picard-O.

Denote C_+ (resp. C_-) the symmetric (resp. skew-symmetric) part of $C(Y)$ and similarly for G. It follows from (2.15) and (2.23) that

$$C = C_+ + C_- = C_+ + G_-$$

since $C_- = G_- = (G - G^\top)/2$.

One can show that Y is a fixed point of FastICA if and only if $G(Y)$ is symmetric and $C_+(Y)$ is positive definite. Indeed, at a fixed point, $C_w(Y) = I_p$, so that by Eq. (2.24), one has $C(Y) = (CC^\top)^{1/2}$ which is a positive matrix (almost surely). Conversely, if $G(Y)$ is symmetric, then so is $C(Y)$. If $C(Y)$ is also positive, then its polar factor $C_w(Y)$ is the identity matrix, so that Y is a fixed point of FastICA.

The modification of FastICA ensures that the diagonal of $C(Y)$ is positive, but does not guarantee positive definiteness. However, we empirically observed that on each dataset used in the experiments, the matrix $C_+(Y)$ is positive definite when $G_-(Y)$ is small. Under that condition, we see that the stationary points of Picard-O, characterized by $G_-(Y) = 0$ are exactly the fixed points of FastICA.

Asymptotic behavior of FastICA : Let us now expose the behavior of FastICA close to a fixed point *i.e.* when $C_w(Y) = \exp(\mathcal{E})$ for some small skew-symmetric matrix $\mathcal{E}$.

At first order in $\mathcal{E}$, the polar factor $C_w = \exp(\mathcal{E})$ of C is obtained as solution of (proof omitted):

$$G_- = \frac{C_+\mathcal{E} + \mathcal{E}C_+}{2} \quad . \tag{2.25}$$

Denote by $\hat{H}$ the linear mapping $\hat{H} : \mathcal{E} \to -\frac{C_+\mathcal{E}+\mathcal{E}C_+}{2}$. When FastICA perform a small move, it is (at first order) of the form $W \leftarrow e^D W$ with $D = -\hat{H}^{-1}(G_-)$. It corresponds to a quasi-Newton step with $\hat{H}$ as approximate Hessian.

Furthermore, under the mixture model assumption, close from separation and with a large number of samples, C_+ becomes the diagonal matrix of coefficients $\delta_{ij}\hat{\kappa}_i$ and $\hat{H}$ simplifies, giving the same direction D given in (2.21).

In summary, we have shown that a slightly modified version of FastICA (with essentially the same iterates as the original algorithm) has the same fixed points as Picard-O. Close to such a fixed point, each of FastICA's iteration is similar to a quasi-Newton step with an approximate Hessian. This approximation matches the true Hessian if the mixture model holds, but this cannot be expected in practice on real data.

2.2.5 Experiments

This section illustrates the relative speeds of FastICA and Picard-O. Both algorithms are coded in the Python programming language. Their computation time being dominated by score evaluations, dot products and sample averages, a fair comparison is obtained by ensuring that both algorithms call the exact same procedures so that speeds differences mostly stems from algorithmics rather than implementation.

Figure 2.5 summarizes our results. It shows the evolution of the projected gradient norm $||G - G^\top||$ versus iterations (left column) and versus time (right column). The 4 rows correspond to 4 data types: synthetic mixtures, fMRI signals, EEG signals, and image patches. FastICA and Picard-O speeds are compared using $\psi(\cdot) = \tanh(\cdot)$. The signals are centered and whitened before running ICA.

Experiments are repeated several times for each setup. The solid line shows the median of the gradient curves and the shaded area shows the $10\ \% - 90\ \%$ percentile (meaning that half the runs completed faster than the solid line and that 80% have convergence curves in the shaded area).

Synthetic data We generate $p = 50$ i.i.d. sources of length $n = 10000$. The 25 first sources follow a uniform law between -1 and 1, the 25 last follow a Laplace law ($d \propto \exp(-|x|)$). The $p \times n$ source matrix S is multiplied by a random square mixing matrix A. This experiment is repeated 100 times, changing each time the seed generating the signals and the mixing matrix.

fMRI This is functional MRI data processed by group ICA [Varoquaux et al., 2010]. The datasets come from ADHD-200 consortium [Consortium et al., 2012]. The problem is of size $p = 60$, $n = 60000$, and the experiments are repeated over 20 datasets.

EEG ICA is applied on 13 publicly available[5] electroencephalography datasets [Delorme et al., 2012]. Each recording contains $p = 71$ signals, of length $n \simeq 75000$.

Image patches We use a database of 80 different images of open country [Oliva and Torralba, 2001]. From each image, $n = 10000$ patches of size 8×8 are extracted and vectorized to obtain $p = 64$ signals, before applying ICA.

Results. FastICA is slightly faster than Picard-O on the simulated problem, for which the ICA mixture model holds perfectly. However, on real data, the rate of convergence of FastICA is severely impaired because the underlying Hessian approximation is far from the truth, while our algorithm still converges quickly. Picard-O is also more consistent in its convergence pattern, showing less spread than FastICA.

2.2.6 Discussion

We have shown that, close from its fixed points, FastICA's iterations are similar to quasi-Newton steps for maximizing a likelihood. Furthermore, the underlying Hessian approximation matches the true Hessian of the problem if the signals are independent. However, on real datasets, the independence assumption never perfectly holds. Consequently, FastICA may converge very slowly on applied problems [Chevalier et al., 2004] or can get stuck in saddle points [Tichavsky et al., 2006].

To overcome this issue, we propose the Picard-O algorithm. It uses a preconditioned L-BFGS technique to solve the same minimization problem as FastICA. Extensive experiments on three types of real data demonstrate that Picard-O can be orders of magnitude faster than FastICA.

[5] https://sccn.ucsd.edu/wiki/BSSComparison

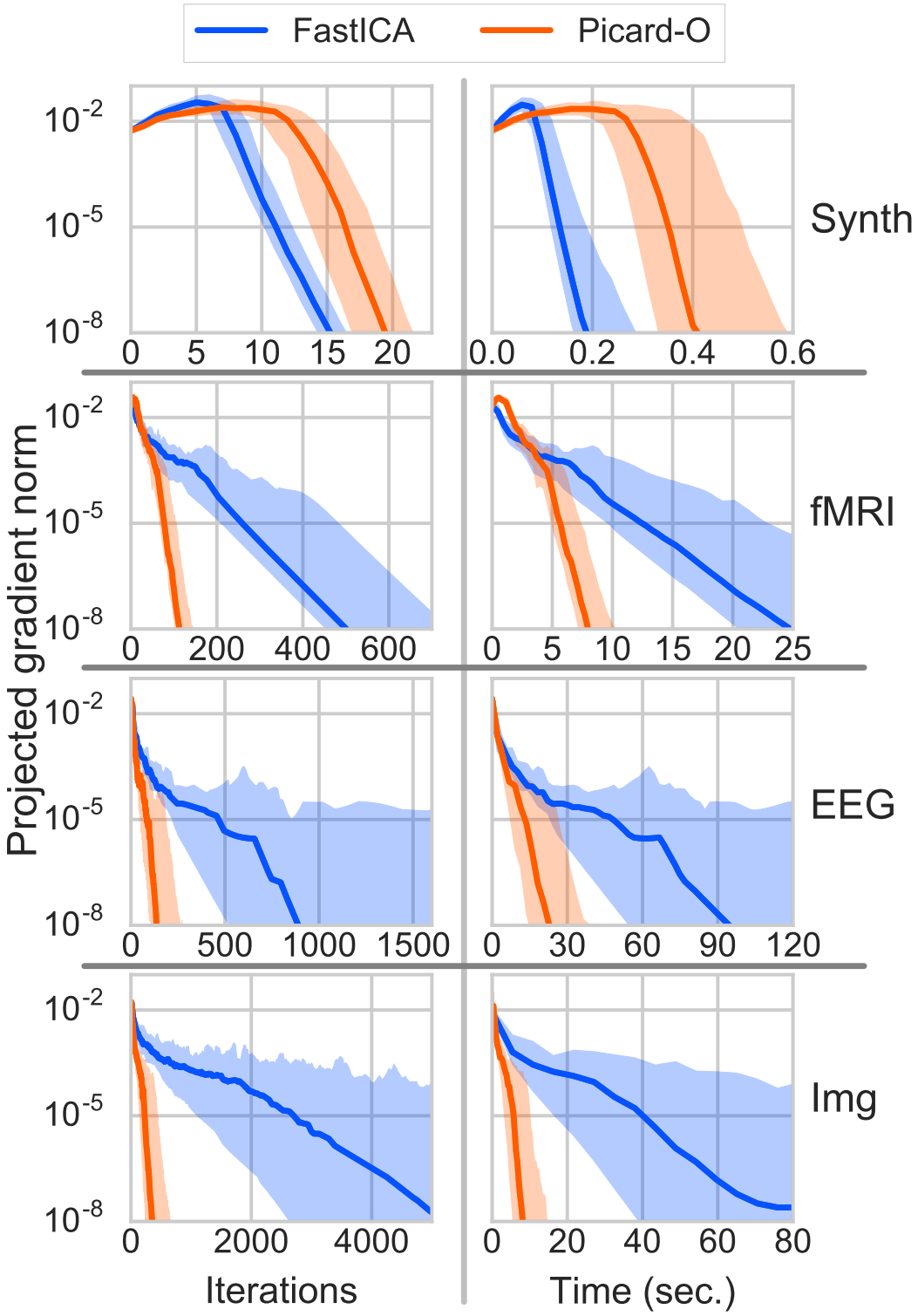


Figure 2.5 – Comparison between FastICA and Picard-O. Gradient norm vs iterations (left column) and vs time (right column). From top to bottom: simulated data, fMRI data, EEG data and image data. Solid line corresponds to the median of the runs, the shaded area covers the 10% – 90% percentiles.

3

Stochastic algorithms for ICA with descent guarantees

Contents

3.1 Stochastic algorithms for ICA

This section presents the work published in:

Pierre Ablin, Alexandre Gramfort, Jean-François Cardoso, and Francis Bach. Stochastic algorithms with descent guarantees for ICA. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1564–1573, 2019c

Only minor changes have been made: shorter introduction and notation harmonization.

3.1.1 Introduction

In this section, we make a clear distinction between the expected negative log-likelihood, and the empirical log-likelihood. The *expected risk* is:

$$\begin{aligned} \mathcal{L}(W) &:= \mathbb{E}_{\boldsymbol{x}}[\ell(\boldsymbol{x}, W)] \\ &= -\log|W| - \sum_{i=1}^{p} \mathbb{E}[\log(d([W\boldsymbol{x}]_i))] \ . \end{aligned} \tag{3.1}$$

Given a set of n i.i.d. samples of $\boldsymbol{x}$, $X = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n] \in \mathbb{R}^{p \times n}$, the *empirical risk* reads:

$$\begin{aligned} \mathcal{L}_n(W) &:= \frac{1}{n} \sum_{j=1}^{n} \ell(\boldsymbol{x}_j, W) \\ &= -\log|W| - \frac{1}{n} \sum_{i=1}^{p} \sum_{j=1}^{n} \log(d([WX]_{ij})) \ . \end{aligned} \tag{3.2}$$

We focus on the inference of W in two cases. The first case is the *finite-sum* setting: using only n samples, W is found by minimizing $\mathcal{L}_n$. The second case is the *online* setting, where a stream of samples arriving one by one is considered. In this case, n goes to infinity, and then $\mathcal{L}_n$ tends towards $\mathcal{L}$. It is important to note that it is theoretically established [Amari et al., 1997] and empirically observed that these criteria allow to unmix super-Gaussian sources even if their densities are different from d.

Although not formulated like this in the original article, [Cardoso, 1997] shown that Infomax solves the empirical risk minimization problem 3.2. It does so by using a stochastic gradient method. However, $\mathcal{L}_n$ not being convex, it is hard to find a good step-size policy which fits any kind of data [Bottou et al., 2016]. As a consequence, Infomax can take an extremely long time before it reaches convergence, or even fail to converge at all [Montoya-Martínez et al., 2017]. Still, the stochasticity of Infomax makes it efficient when the number of samples n is large, because the cost of one iteration does not depend on n.

On the other hand, several full-batch second-order algorithms have been derived for the exact minimization of $\mathcal{L}_n$. For instance, in [Zibulevsky, 2003], an approximation of the Hessian of $\mathcal{L}_n$ is used to obtain a simple quasi-Newton method. In [Choi and Choi, 2007], a trust region method is proposed using the same Hessian approximation. Full-batch methods are robust and sometimes show quadratic convergence speed, but an iteration can take a very long time when the number of samples n is large. They also crucially rely on a costly line-search strategy because of the non-convexity of the problem.

In this work, we make the following contributions:

- We introduce a set of surrogate functions for ℓ, allowing for a majorization-minimization (MM) approach. We show that this view is equivalent to an EM algorithm for ICA. Consequently, techniques like incremental EM [Neal and Hinton, 1998] and online EM [Cappé and Moulines, 2009] can be efficiently applied to this problem.

- Critically, the surrogate functions can be minimized in closed-form with respect to any single row of W. Thus, the incremental algorithm guarantees the decrease of the surrogate loss at each iteration, without having to resort to expensive line-search techniques. To the best of our knowledge, this feature is a novelty in the field of ICA algorithms.

- Owing to a cheap partial update, the cost of one iteration of the proposed algorithm is similar to the cost of a stochastic gradient descent step. Through experiments, the proposed methods are shown to perform better than the state-of-the-art, while enjoying the robust property of guaranteed decrease.

Notation. In the following, scalar values are noted in lower case (e.g. y), vectors in bold font (e.g. $\boldsymbol{x}$), and matrices in upper case (e.g. W). For a matrix M, $M_{i:}$ denotes its i-th row, and $M_{:j}$ denotes its j-th column.

3.1.2 Representations of super-Gaussian densities

Super-Gaussian densities can be represented in at least two forms: either variationally through a surrogate function, or probabilistically through a Gaussian scale mixture [Palmer et al., 2006]. These two representations lead to the same optimization algorithms but with a slightly different view point.

Surrogate functions

The density d is assumed symmetric and *super-Gaussian* in the sense that $-\log(d(\sqrt{x}))$ is an increasing concave function over $(0, +\infty)$. Following [Palmer et al., 2006], there exists a function f such that:

$$G(y) := -\log(d(y)) = \min_{u \geq 0} \frac{uy^2}{2} + f(u), \tag{3.3}$$

and the minimum is reached for a unique value denoted as $u^*(y)$. Simple computations show that $u^*(y) = \frac{G'(y)}{y}$. For $\boldsymbol{u} \in \mathbb{R}_+^{p \times 1}$, we introduce a new objective function $\tilde{\ell}(\boldsymbol{x}, W, \boldsymbol{u})$ that reads:

$$\tilde{\ell}(\boldsymbol{x}, W, \boldsymbol{u}) := -\log|W| + \sum_{i=1}^{p} [\frac{1}{2} u_i \, [W\boldsymbol{x}]_i^2 + f(u_i)], \tag{3.4}$$

and the associated empirical risk, for $U = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n] \in \mathbb{R}_+^{p \times n}$:

$$\tilde{\mathcal{L}}_n(W, U) := \frac{1}{n} \sum_{j=1}^{n} \tilde{\ell}(\boldsymbol{x}_j, W, \boldsymbol{u}_j) \tag{3.5}$$

$$= -\log|W| + \frac{1}{n} \sum_{i=1}^{p} \sum_{j=1}^{n} [\frac{1}{2} U_{ij} \, [WX]_{ij}^2 + f(U_{ij})]. \tag{3.6}$$

Following Eq. (3.3), we have:

Lemma 3.1 (Majorization). *Let $W \in \mathbb{R}^{p \times p}$. For any $U \in \mathbb{R}_+^{p \times n}$, $\mathcal{L}_n(W) \leq \tilde{\mathcal{L}}_n(W, U)$, with equality if and only if $U = u^*(WX)$.*

Lemma 3.2 (Same minimizers). *Let $W \in \mathbb{R}^{p \times p}$, and $U = u^*(WX)$. Then, W minimizes $\mathcal{L}_n$ if and only if (W, U) minimizes $\tilde{\mathcal{L}}_n$.*

Proof: Using the function G introduced in Eq. (3.3), the loss $\mathcal{L}_n$ writes:

$$\mathcal{L}_n(W) = -\log|W| + \frac{1}{n} \sum_{i=1}^{p} \sum_{j=1}^{n} G([WX]_{ij})$$

For a given matrix $U \in \mathbb{R}^{p \times n}$, using Eq. (3.3) we have for all i, j: $G([WX]_{ij}) \leq \frac{1}{2} U_{ij} [WX]_{ij}^2 + f([WX]_{ij})$, with equality if and only if $U_{ij} = u^*([WX]_{ij})$. Summing these equations yields as expected:

$$-\log|W| + \frac{1}{n} \sum_{i=1}^{p} \sum_{j=1}^{n} G([WX]_{ij}) \leq$$

$$-\log|W| + \frac{1}{n} \sum_{i=1}^{p} \sum_{j=1}^{n} [\frac{1}{2} U_{ij} [WX]_{ij}^2 + f([WX]_{ij})]$$

with equality if and only if for all i, j, $U_{ij} = u^*([WX]_{ij})$. $\square$

In line with the majorization-minimization (MM) framework [Mairal, 2015], these two lemmas naturally suggest to minimize $\mathcal{L}_n(W)$ by alternating the minimization of the auxiliary function $\tilde{\mathcal{L}}_n(W, U)$ with respect to W and U. This will also be shown to be equivalent to the EM algorithm for the Gaussian scale mixture interpretation in the next Section.

The rest of the paper focuses on the minimization of $\tilde{\mathcal{L}}_n$ rather than $\mathcal{L}_n$, which yields the same unmixing matrix by Lemma 3.2.

EM algorithm with Gaussian scale mixtures

Super-Gaussian densities can also be represented as scale mixtures of Gaussian densities [Palmer et al., 2006], that is, $d(y) = \int_0^{+\infty} g(y, \eta) q(\eta) d\eta$, where $g(y, \eta) = \frac{1}{\sqrt{2\pi\eta}} \exp(-\frac{y^2}{2\eta})$ is a centered Gaussian density of variance η, and $q(\eta)$ a distribution on the variance of the Gaussian distribution. It turns out that the EM algorithm using the above form for our ICA model is exactly equivalent to the alternating optimization of $\tilde{\mathcal{L}}_n$ (see a proof in the appendix). The variable U corresponds to the scale parameter in [Palmer et al., 2006] and the EM algorithm alternates between setting U to the posterior mean $u^*(Y)$ (E-step) and a descent move in W (M-step).

Relationship to the noisy case. Many articles (e.g. [Palmer et al., 2006, Girolami, 2001, Bermond and Cardoso, 1999]) have proposed EM-based techniques for the estimation of the latent parameters of the more general linear model:

$$\boldsymbol{x} = A\boldsymbol{s} + \boldsymbol{n} \ , \tag{3.7}$$

where A is the mixing matrix, and $\boldsymbol{n} \sim \mathcal{N}(0, \Sigma)$ is a Gaussian variable accounting for noise. In [Palmer et al., 2006], the matrix A is assumed to be known, as well as the noise covariance Σ. On the contrary, the present work deals with the case where A is unknown, and where there is no noise. The noisy case (with unknown A) is studied in e.g. [Bermond and Cardoso, 1999, Girolami, 2001]. An EM algorithm is derived for the estimation of $\boldsymbol{s}$, A and Σ. In the appendix, it is shown that this EM algorithm makes no progress in the limit of noise-free observations since the EM update rule for A becomes $A \leftarrow A$ when $\Sigma = 0$. Hence, the EM algorithms found in the literature for the noisy case suffer considerable slowdown in high signal-to-noise regime. In contrast, the approach derived in the following section is not affected by this problem.

Examples

Many choices for G can be found in the ICA literature. In the following, we omit irrelevant normalizing constants. The original Infomax paper [Bell and Sejnowski, 1995] implicitly uses $G(y) = \log(\cosh(y))$ since it corresponds to $G'(y) = \tanh(y)$ and $u^*(y) = \frac{\tanh(y)}{y}$. This density model is one of the most widely used. However, since an ICA algorithm has to evaluate those functions many times, using simpler functions offers significant speedups. One possibility is to use a Student distribution: $G(y) = \frac{1}{2}\log(1 + y^2)$, for which $u^*(y) = \frac{1}{1+y^2}$. In the following, we choose the Huber function: $G(y) = \frac{1}{2}y^2$ if $|y| < 1$ and $G(y) = |y| - \frac{1}{2}$ if not. This gives $u^*(y) = 1$ if $|y| < 1$, $u^*(y) = \frac{1}{|y|}$ otherwise.

3.1.3 Stochastic minimization of the loss function

Using a MM strategy, $\tilde{\mathcal{L}}_n(W,U)$ is minimized by alternating descent moves in U and in W. We propose an incremental technique which minimizes $\tilde{\mathcal{L}}_n$ with a finite number of samples, and an online technique where each sample is only used once. The pseudo code for these algorithms is given in Algorithms 25 and 31. The difference between incremental and online technique only reflects through the variable U which is estimated at the majorization step. Hence, we first discuss the minimization step.

Minimization step: Descent in W

Expanding $[WX]_{ij}^2$, the middle term in the new loss function (3.4) is quadratic in the rows of W:

$$\tilde{\mathcal{L}}_n = -\log|W| + \frac{1}{2}\sum_{i=1}^{p} W_{i:}A^i W_{i:}^\top + \frac{1}{n}\sum_{i=1}^{p}\sum_{j=1}^{n} f(U_{ij}), \tag{3.8}$$

where $W_{i:}$ denotes the i-th row of W, and the A^i's are $p \times p$ matrices given by:

$$A^i_{kl} := \frac{1}{n}\sum_{j=1}^{n} U_{ij}X_{kj}X_{lj} \ . \tag{3.9}$$

Therefore, when U is fixed, with respect to W, $\tilde{\mathcal{L}}_n$ is the sum of the log det function and a quadratic term. The minimization of such a function is difficult, mostly because the log det part introduces non-convexity. However, similarly to a coordinate descent move, it can be *exactly partially* minimized in closed-form:

Lemma 3.3 (Exact partial minimization)**.** *Let $i \in [1,p]$, and $\boldsymbol{m} \in \mathbb{R}^{1\times p}$ ($\boldsymbol{m}$ is a* row *vector). Consider the mapping $\Theta_i(\boldsymbol{m}) : \mathbb{R}^{1\times p} \to \mathbb{R}^{p\times p}$ such that the matrix $\Theta_i(\boldsymbol{m})$ is equal to I_p, except for its i-th row which is equal to $\boldsymbol{m}$.*

Let $W \in \mathbb{R}^{p\times p}$ and $U \in \mathbb{R}^{p\times n}$. Define $K := WA^iW^\top \in \mathbb{R}^{p\times p}$. Then,

$$\arg\min_{\boldsymbol{m}\in\mathbb{R}^{1\times p}} \tilde{\mathcal{L}}_n(\Theta_i(\boldsymbol{m})W, U) = \frac{1}{\sqrt{(K^{-1})_{ii}}}(K^{-1})_{i:} \ . \tag{3.10}$$

Proof: With respect to $\boldsymbol{m}$, $\tilde{\mathcal{L}}_n(\Theta_i(\boldsymbol{m})W, U)$ is of the form $\phi(\boldsymbol{m}) = -\log(|m_i|) + \boldsymbol{m}K\boldsymbol{m}^\top$. Restraining to the region $m_i > 0$, this function is strongly convex and smooth, and thus possesses a single minimum found by cancelling the gradient. Simple algebra shows :

$$\nabla\phi(\boldsymbol{m}) = -\frac{1}{m_i}\boldsymbol{e}^i + \boldsymbol{m}K \ ,$$

where $\boldsymbol{e}^i$ is the i-th canonical basis vector. Cancelling the gradient yields $\boldsymbol{m} = \frac{1}{m_i}(K^{-1})_{i:}$, and inspection of the i-th coordinate of this relationship gives $m_i = \frac{(K^{-1})_{ii}}{m_i}$, providing the expected result. □

In other words, we can exactly minimize the loss with a *multiplicative update* of one of its rows. Performing multiplicative updates on the iterate W enforces the *equivariance* of the proposed methods [Cardoso and Laheld, 1996].

Majorization step : Descent in U

For a fixed unmixing matrix W, Lemma 3.1 gives: $\arg\min_U \tilde{\mathcal{L}}_n(W, U) = u^*(WX)$. Such an operation works on the full batch of samples X. When only one sample $X_{:j} = \boldsymbol{x}_j \in \mathbb{R}^{p\times 1}$ is available, the operation $U_{:j} \leftarrow u^*(W\boldsymbol{x}_j)$ minimizes $\tilde{\mathcal{L}}_n(W, U)$ with respect to the j-th column of U. As seen previously (Section 3.1.3), we only need to compute the A^i's to perform a descent in W, hence one needs a way to accumulate those matrices.

Incremental algorithm. To do so in an incremental way [Neal and Hinton, 1998], a memory $U^{\text{mem}} \in \mathbb{R}^{p\times n}$ stores the values of U. When a sample $\boldsymbol{x}_j$ is seen by the algorithm, we compute $U^{\text{new}}_{:j} = u^*(W\boldsymbol{x}_j)$, and update the A^i's as:

$$A^i \leftarrow A^i + \frac{1}{T}(U^{\text{new}}_{ij} - U^{\text{mem}}_{ij})\boldsymbol{x}_j\boldsymbol{x}_j^\top \quad . \tag{3.11}$$

The memory is then updated by $U^{\text{mem}}_{:j} \leftarrow U^{\text{new}}_{:j}$ enforcing $A^i = \frac{1}{n}\sum_{j=1}^n U^{\text{mem}}_{ij}\boldsymbol{x}_j\boldsymbol{x}_j^\top$ at each iteration.

Online algorithm. When each sample is only seen once, there is no memory, and a natural update rule following [Cappé and Moulines, 2009] is:

$$A^i \leftarrow (1-\rho(n))A^i + \rho(n)U_{ij}\boldsymbol{x}_j\boldsymbol{x}_j^\top \quad , \tag{3.12}$$

where n is the number of samples seen, and $\rho(n) \in [0, 1]$ is a well chosen factor. Setting $\rho(n) = \frac{1}{n}$ yields the unbiased formula $A^i(n) = \frac{1}{n}\sum_{j=1}^n U_{ij}\boldsymbol{x}_j\boldsymbol{x}_j^\top$. A more aggressive policy $\rho(n) = \frac{1}{n^\alpha}$ for $\alpha \in [\frac{1}{2}, 1)$ empirically leads to faster estimation of the latent parameters. Note that since we are averaging sufficient statistics, there is no need to multiply $\rho(n)$ by a constant.

Complexity analysis

Memory: The proposed algorithm stores p matrices A^i, which requires a memory of size $\frac{p^2(p+1)}{2}$ (since they are symmetric). In the incremental case, it stores the real numbers U_{ij}, requiring a memory of size $p \times n$. In most practical cases of ICA, the number of sources p is very small compared to n ($n \gg p$), meaning that the dominating memory cost is $p \times n$. In the online case, the algorithm only loads one mini-batch of data at a time, leading to a reduced memory size of $p \times n_b$, where n_b is the mini-batch size.

Time: The majorization step requires to update each coefficient of the matrices A^i's, meaning that it has a time complexity of $p^3 \times n_b$. The minimization step requires to solve p linear systems to obtain the matrices $K^{-1}_{i:}$. Each one takes $O(p^3)$. An improvement based on preconditioned conjugate gradient method [Shewchuk et al., 1994] is proposed in the appendix to reduce the computational cost. The total cost of the minimization step is thus $O(p^4)$. In practice, $p \ll n_b$, so the overall cost of one iteration is dominated by the majorization step, and is $\frac{p^2(p+1)}{2} \times n_b$. A stochastic gradient descent algorithm with the same mini-batch size n_b, as described later in Section 3.1.4, has a lower time complexity of $p^2 \times n_b$. We now propose a way to reach the same time complexity with the MM approach.

Gap-based greedy update

In order to reduce the complexity by one order of magnitude in the majorization step, only a subset of fixed size $q < p$ of the matrices A^i is updated for each sample. Following Eq. (3.3), it is given by what we call *gap* : a positive quantity measuring the decrease

Algorithm 3.1 Incremental MM algorithm for ICA

Input : Samples $X \in \mathbb{R}^{p \times n}$

Param: Number of iterations $t_{\max}$, mini-batch size n_b, number of coordinates to update per sample q

Init : Initialize $W = I_p$, $U^{\text{mem}} = 0 \in \mathbb{R}^{n \times p}$ and $A^i = 0 \in \mathbb{R}^{p \times p}$, $\forall i \in [1, p]$

for $t = 1, \dots, t_{max}$ **do**
  Select a mini-batch b of size n_b at random
  for *each index* $j \in b$ **do** // Majorization
    Select $\boldsymbol{x} = X_{:j}$
    Compute $\boldsymbol{u}^{\text{new}} = u^*(W\boldsymbol{x})$
    Compute the gaps (3.13)
    Find the q sources $i_1, \dots, i_q$ corresponding to the largest gaps
    Update A^i for $i = i_1, \dots, i_q$ using Eq. (3.11)
    Update the memory: $U^{\text{mem}}_{:j} = \boldsymbol{u}^{\text{new}}$
  for $i = 1, \dots, p$ **do** // Minimization
    Update the i-th row of W using Eq. (3.10)
return W

in $\tilde{\mathcal{L}}_n$ provided by updating U_{ij}. In the following, define $\tilde{U}_{i'j'} := U^{\text{mem}}_{i'j'}$ if $(i', j') \neq (i, j)$, and $\tilde{U}_{ij} := U^{\text{new}}_{ij} = u^*([WX]_{ij})$. The gap is given by:

$$\text{gap}(W, U^{\text{mem}}_{ij}) := \tilde{\mathcal{L}}_n(W, U^{\text{mem}}) - \tilde{\mathcal{L}}_n(W, \tilde{U}) \tag{3.13}$$

$$= \frac{1}{2} U^{\text{mem}}_{ij} \, [WX]^2_{ij} + f(U^{\text{mem}}_{ij}) - G([WX]_{ij}) \ . \tag{3.14}$$

Since all the above quantities are computed during one iteration anyway, computing the gap for each signal $i \in [1, p]$ only adds a negligible computational overhead, which scales linearly with p. Then, in a greedy fashion, only the coefficients U_{ij} corresponding to the q largest gaps are updated, yielding the largest decrease in $\tilde{\mathcal{L}}_n$ possible with q updates. In the experiments (Figure 3.4), we observe that it is much faster than a random selection, and that it does not impair convergence too much compared to the full-selection ($q = p$). In the online setting, there is no memory, so we simply choose q indices among p at random.

Related work: The matrices A^i are *sufficient statistics* of the surrogate ICA model for a given value of U. The idea to perform a coordinate descent move (3.10) after each update of the sufficient statistics is inspired by online dictionary learning [Mairal et al., 2009], Gaussian graphical models [Honorio et al., 2012] and non-negative matrix factorization [Lefevre et al., 2011].

3.1.4 Experiments

In this section, we compare the proposed approach to other classical methods to minimize $\mathcal{L}$. The code for the proposed methods is available online at https://github.com/pierreablin/mmica.

Algorithm 3.2 Online MM algorithm for ICA

Input : A stream of samples X in dimension $\mathbb{R}^p$
Param: Number of iterations $t_{\max}$, mini-batch size n_b, number of coordinates to update per sample q
Init : Initialize $W = I_p$ and $A^i = 0 \in \mathbb{R}^{p\times p}$, $\forall i \in [1, p]$

for $t = 1, \dots, t_{max}$ **do**
  Fetch n_b samples from the stream
  for *each fetched sample* $\boldsymbol{x}$ **do** // Majorization
    Compute $\boldsymbol{u} = u^*(W\boldsymbol{x})$
    Compute q indices $i_1, \dots, i_q$ at random
    Update A^i for $i = i_1, \dots, i_q$ using Eq. (3.12)
  for $i = 1 \dots, p$ **do** // Minimization
    Update the i-th row of W using Eq. (3.10)
return W

Compared algorithms

Stochastic gradient descent (SGD). Given a mini-batch b containing n_b samples, the relative gradient $\nabla(\mathcal{L}_n)_{ik} = \frac{1}{n_b}\sum_{j\in b} G'([WX]_{ij})[WX]_{kj}$ is computed. Then, a descent move $W \leftarrow (I_n - \rho\nabla(\mathcal{L}_n))W$ is performed. The choice of the step size ρ is critical and difficult. The original article uses a constant step size, but more sophisticated heuristics can be derived. This method can be used both for the finite sum and the online problem. It is important to note that once WX and $G'(WX)$ are computed, it needs twice as many elementary operations to compute the gradient as it takes to update one matrix A^i (Eq. (3.11) and Eq.(3.12)) when $n_b \gg p$. The first computation requires $n_b \times p^2$ operations, while the second takes $n_b \times \frac{p(p+1)}{2}$ (since the matrices A^i are symmetric). When n_b is large enough, as it is the case in practice in the experiments, these computations are the bottlenecks of their respective methods. Hence, we take $q = 2$ in the experiments for the MM algorithms, so that the theoretical cost of one iteration of the proposed method matches that of SGD.

Variance reduced methods. One of the drawbacks of the stochastic gradient method is its sub-linear rate of convergence, which happens because the stochastic gradient is a very noisy estimate of the true gradient. Variance reduced methods such as SAG [Schmidt et al., 2017], SAGA [Defazio et al., 2014] or SVRG [Johnson and Zhang, 2013] reduce the variance of the estimated gradient, leading to better rates of convergence. However, these methods do not solve the other problem of SGD for ICA, which is the difficulty of finding a good step-size policy. We compare our approach to SAG, which keeps the past stochastic gradients in memory and performs a descent step in the averaged direction. This approach is however only relevant in the finite-sum setting.

Full batch second order algorithms. We compare our approach to the "Fast-Relative Newton" method (FR-Newton) [Zibulevsky, 2003] and the "Preconditioned ICA for Real Data" algorithm (Picard, Sec. 2.1) . The former performs quasi-Newton steps using a simple approximation of the Hessian of $\mathcal{L}_n$, which is as costly to compute as a gradient. The later refines the approximation by using it as a preconditioner for the L-BFGS algorithm. For both algorithms, one iteration requires to compute the gradient and the Hessian on the full dataset, resulting in a cost of $2 \times p^2 \times n$, and to evaluate the gradient and loss function for each point tested during the line search, so the overall cost is $(2 + n_{\text{ls}}) \times p^2 \times n$ where $n_{\text{ls}} \geq 1$ is the number of points tested during

the line-search. Thus, one epoch requires more than 3 times more computations than one of SGD or of the proposed algorithms. These algorithms cannot be used online.

Full batch MM. For the finite-sum problem, we also compare our approach to the full-batch MM, where the whole U is updated at the majorization step.

FastICA. FastICA [Hyvärinen, 1999] is a full batch fixed point algorithm for ICA. It *does not solve the same optimization problem* as the one presented in this paper (it does not minimize $\mathcal{L}_n$, see [Hyvärinen, 1999]). Hence, we do not include metrics involving $\mathcal{L}_n$ to benchmark it. However, it is one of the most widely used algorithms for ICA in practical applications, and is popular for its fast estimation speed. Furthermore, it is shown to have similar convergence properties as FR-Newton (see chapter 2)

Performance measures

The following quality measures are used to assess the performance of the different algorithms:

Loss on left-out data: It is the value of the loss on some data coming from the same dataset but that have not been used to train the algorithms. This measure, which boils down to the likelihood of left-out data, is similar to the *testing error* in machine learning, and can be computed in both the streaming and finite-sum settings.

Amari distance [Moreau and Macchi, 1998]: When the true mixing matrix A is available, for a matrix W, the product $R = WA$ is computed, and the *Amari distance* is given by:

$$\sum_{i=1}^{p} \left(\sum_{j=1}^{p} \frac{R_{ij}^2}{\max_l R_{il}^2} - 1 \right) + \sum_{i=1}^{p} \left(\sum_{j=1}^{p} \frac{R_{ji}^2}{\max_l R_{lj}^2} - 1 \right).$$

This distance measures the proximity of W and A^{-1} up to scale and permutation indetermination. It cancels if and only if R is a scale and permutation matrix, i.e., if the separation is perfect. This measure is relevant both for the online and finite-sum problems. It is the only metric for which it makes sense to compare FastICA to the other algorithms since it does not involve the loss function.

Relative gradient norm: The norm of the full-batch relative gradient of $\mathcal{L}_n$ is another measure of convergence. Since the problem is non-convex, the algorithms may converge to different local minima, which is why we favor this metric over the *train error*. It is however only relevant in the finite-sum setting. In this setting, a converging algorithm should drive the norm of the full-batch relative gradient to zero.

Parameters and initialization

The stochastic algorithms (SGD, SAG, and the proposed MM techniques) are used with a batch size of $n_b = 1000$. The proposed MM algorithms are run with a parameter $q = 2$, which ensures that each of their iterations is equivalent to one iteration of the SGD algorithm. In the online setting, we use a power $\alpha = 0.5$ to speed up the estimation. The step-sizes of SGD and SAG are chosen by trial and error on each dataset, by finding a compromise between convergence speed and accuracy of the final mixing matrix. In the online case, the learning rate is chosen as $\lambda \times n^{-0.5}$ for SGD. FR-Newton and Picard are run with its default parameters.

Regarding initialization, it is common to initialize an ICA algorithm with an approximate *whitening* matrix. A whitening matrix W is such that the signals $W\boldsymbol{x}$ are decorrelated. It is interesting to start from such a point in ICA because decorrelation is a necessary condition for independence.

Denoting $C_{\boldsymbol{x}}$ the correlation matrix of the signals, the whitening condition writes $WC_{\boldsymbol{x}}W^\top = I_p$. Hence, the whitening matrices are the $W = RC_{\boldsymbol{x}}^{-\frac{1}{2}}$ where R is a rotation ($R^\top R = I_p$). In practice, we take $R = I_p$. The covariance matrix needs to be estimated. In the case of a fixed dataset $X \in \mathbb{R}^{p\times n}$, we can use the empirical covariance $\tilde{C}_X = \frac{1}{n}XX^\top$ as an approximation. However, the cost of such a computation, $O(p^2 \times n)$, gets prohibitively large as n grows. Since the whitening is only an initialization, it needs not be perfectly accurate. Hence, in practice, we compute the empirical covariance on a sub-sampled version of X of size $n = 10^4$. The same goes for the online algorithm: we fetch the first 10^4 samples to compute the initial approximate whitening matrix.

Datasets

Synthetic datasets: For this experiment, we generate a matrix $S \in \mathbb{R}^{p\times n}$ with $p = 10$ and $n = 10^6$ of independent sources following a super-Gaussian Laplace distribution: $d(x) = \frac{1}{2}\exp(-|x|)$. Note that this distribution does not match the Huber function used in the algorithms, but estimation is still possible since the sources are super-Gaussian. Then, we generate a random mixing matrix $A \in \mathbb{R}^{p\times p}$ of normally distributed coefficients. The algorithms discussed above are then run on $X = AS$, and the sequence of iterates produced is recorded. Finally, the different quality measures are computed on those iterates. We repeat this process 100 times with different random realizations, in order to increase the robustness of the conclusions. The averaged quality measures are displayed in Fig. 3.1. In order to compare different random initializations, the loss evaluated on left-out data is always shifted so that its plateau is at 0.

To observe the effect of the greedy gap selection, we generate another dataset in the same way with $p = 30$, $n = 10^5$. Results are displayed in Fig. 3.4.

Real datasets: The algorithms are applied on classical ICA datasets, covering a wide range of dimensions p. The first experiment is in the spirit of [Hoyer and Hyvärinen, 2000].

We extract a big 32GB dataset of $n = 4 \times 10^7$ square patches of size 10×10 from natural images. Each patch is vectorized into an array of dimension $p = 100$. Only the online algorithms are used to process this dataset since it does not fit into RAM. The results on this dataset are displayed in Fig. 3.2.

We also generate smaller datasets in the same fashion, of size $n = 10^6$, and 10×10 patches. The dimension is reduced to $p = 10$ using PCA.

Finally, an openly available EEG dataset [Delorme et al., 2012] of dimension $p = 71$, $n = 10^6$ is used without dimension reduction. Each signal matrix is multiplied by a $p \times p$ random matrix. The different algorithms are applied on these datasets with 10 different random initializations, and for 50 epochs in the finite sum setting. Results are displayed in Fig. 3.3.

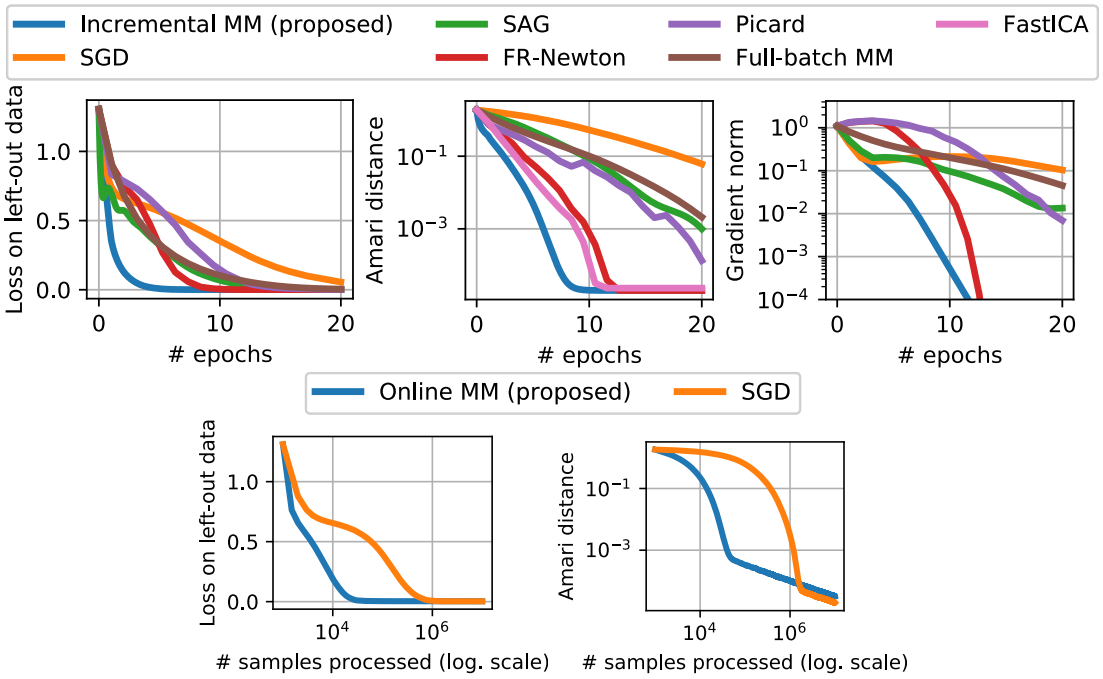


Figure 3.1 – Results on synthetic data. Top: finite-sum problem. 100 datasets of size $n = 10^6$ and $p = 10$ are generated, each algorithm performs 20 epochs (passes on the dataset). Bottom: online problem. 100 datasets of size $n = 10^7$ and $p = 10$ are generated, each algorithm performs one pass on each dataset. Metrics are displayed with respect to epochs/number of passes.

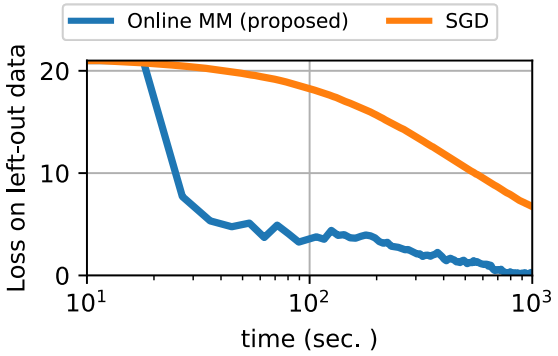


Figure 3.2 – Online algorithms applied on a 32 GB real dataset with $p = 100$ and $n = 4 \times 10^7$. Time is in logarithmic scale. Values of the loss on left out data greater than its initial value are truncated.

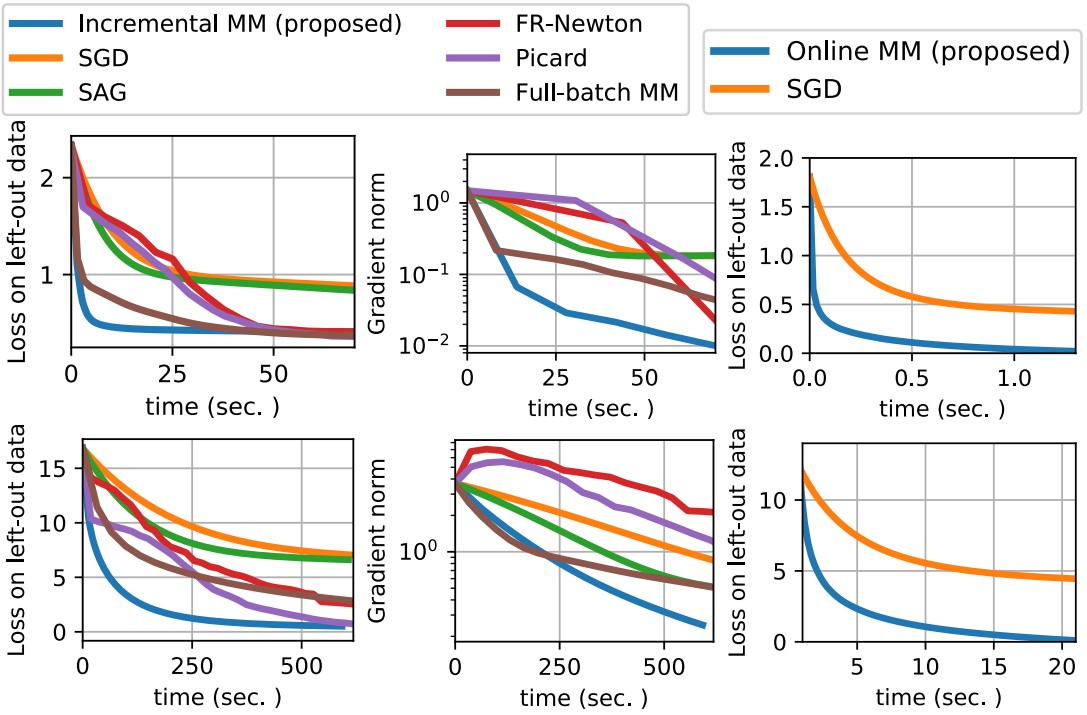


Figure 3.3 – Behavior of different algorithms on real data. Top: 15 image patch datasets of size 10×10^6 are generated, and the averaged results are displayed. Bottom: same with 15 EEG datasets of size 71×10^6. Left and middle: finite sum problem. Right: online problem. Metrics are displayed with respect to time.

Discussion

Experiments run on both synthetic and real data of various dimensions demonstrate that the proposed methods consistently perform best when quantifying the loss on left-out data (test error), for large scale datasets with many samples. This metric is arguably the most important from a statistical machine learning standpoint. This is also validated by the Amari distance in the simulated case: the proposed method shows similar convergence as FR-Newton and FastICA, and outperforms other algorithms. Regarding the gradient norm metric (similar to training error), in the simulated and image patch experiment, the proposed algorithm is in the end slower than FR-Newton. This behavior is expected: the incremental algorithm has a linear convergence, while second order methods are quadratic algorithm.

However, FR-Newton catches up with the proposed algorithm well after the testing error plateaus, so when the error of the model is dominated by the *estimation error* [Bottou and Bousquet, 2008] rather than the optimization error.

Effect of the greedy update rule: On the 30×10^5 dataset (Fig. 3.4), we run the incremental algorithm with the greedy coordinate update rule discussed in Sec. 3.1.3 with $q = 1$ and $q = 3$. We compare it to a random approach (where q random sources are updated at each iteration) for the same values of q, and to the more costly full-selection algorithm, where each source is updated for each sample. The greedy approach only adds a negligible computational overhead linear in p compared to the random approach, while leading to much faster estimation. In terms of generalization error, it is only slightly outperformed by the full selection approach ($q = p$).

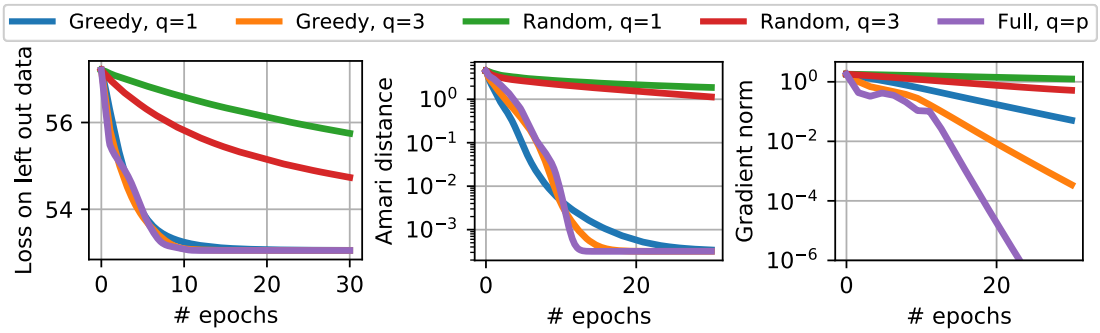


Figure 3.4 – Effect of the greedy update rule, on a synthetic problem of size $p = 30$, $n = 10^5$. For a similar complexity, the greedy approach gives much faster convergence than the random approach.

3.1.5 Conclusion

In this part, we have introduced a new majorization-minimization framework for ICA, and have shown that it is equivalent to an EM approach for Gaussian scale mixtures. Our method has the valuable advantage of guaranteeing a decrease of the surrogate loss function, which enables stochastic methods with descent guarantees. This is, to the best of our knowledge, a unique feature for a stochastic ICA algorithm. We have proposed both an incremental and an online algorithm for the finite-sum and online problems, with the same complexity as SGD thanks to an efficient greedy coordinate descent update. Experiments show progress on current state-of-the-art, without the need for tedious manual setting of any parameter.

Appendix

3.1.6 Proof of equivalence of EM

Given the Gaussian scale mixture formulation of d, as $d(y) = \int_0^{+\infty} g(y, \eta) q(\eta) d\eta$, an EM algorithm would do the following:

$$\begin{aligned}
\mathcal{L}_n(W) &= -\log|W| - \frac{1}{n}\sum_{i=1}^{p}\sum_{j=1}^{n}[\log(d([WX]_{ij})) \\
&= -\log|W| - \frac{1}{n}\sum_{i=1}^{p}\sum_{j=1}^{n}\log(\int_0^{\infty} g([WX]_{ij}, \eta) q(\eta) d\eta) \\
&\leqslant -\log|W| - \frac{1}{n}\sum_{i=1}^{p}\sum_{j=1}^{n}\int_0^{\infty} r_{ij}(\eta)\log\frac{g([WX]_{ij}, \eta) q(\eta)}{r_{ij}(\eta)} d\eta
\end{aligned}$$

where the r_{ij} are any density functions. The equality if and only if $r_{ij}(\eta) \propto g([WX]_{ij}, \eta) q(\eta)$. The upper bound in the last equation can be written as:

$$-\log|W| + \frac{1}{2n}\sum_{i=1}^{p}\sum_{j=1}^{n}\tilde{U}_{ij}\,[WX]_{ij}^2 \;+\; c \;,$$

where $\tilde{U}_{ij} = \int_0^{\infty} r_{ij}(\eta)\eta^{-1} d\eta$ and where c is a remaining term which does not depend on W. Thus, the upper bound has the same dependence on W as the loss $\tilde{\mathcal{L}}_n(W)$. The E-step thus computes $\tilde{U}_{ij} = \int_0^{\infty} r_{ij}(\eta)\eta^{-1} d\eta$ using a density $r_{ij} \propto g([WX]_{ij}, \eta) q(\eta)$.

This exactly corresponds to the majorization step described in our algorithm. The proof, from [Palmer et al., 2006], is as follows. Dropping the indices for readability, and denoting $y = [WX]_{ij}$, normalizing r gives:

$$r(\eta) = \frac{g(y,\eta)q(\eta)}{d(y)}$$

so that

$$\tilde{U}(y) = \frac{1}{d(y)} \int_0^\infty g(y,\eta)\eta^{-1}q(\eta)d\eta.$$

Hence, using $\frac{\partial}{\partial y}g(y,\eta) = -g(y,\eta)y/\eta$, we get

$$\tilde{U}(y) = -\frac{1}{yd(y)} \int_0^\infty \frac{\partial}{\partial y}g(y,\eta)q(\eta)d\eta = -\frac{d'(y)}{yd(y)} = u^*(y) \ .$$

This demonstrates that, although not formulated in this fashion, the proposed method is indeed equivalent to an EM algorithm.

3.1.7 The EM algorithm for noisy mixtures is stuck in the noise-free limit

We follow the update rules given in [Bermond and Cardoso, 1999, Girolami, 2001]. The model is $\boldsymbol{x} = A\boldsymbol{s} + \boldsymbol{n}$ where $\boldsymbol{n} \sim \mathcal{N}(0, \Sigma)$. Key quantities for the update rule are the following expectations:

$$\mathbb{E}[\boldsymbol{s}|\boldsymbol{x}] = (A^\top\Sigma^{-1}A + \Lambda^{-1})^{-1}A^\top\Sigma^{-1}\boldsymbol{x} \tag{3.15}$$

$$\mathbb{E}[\boldsymbol{s}\boldsymbol{s}^\top|\boldsymbol{x}] = (A^\top\Sigma^{-1}A + \Lambda^{-1})^{-1} + \mathbb{E}[\boldsymbol{s}|\boldsymbol{x}]\mathbb{E}[\boldsymbol{s}|\boldsymbol{x}]^\top \ , \tag{3.16}$$

where Λ is a diagonal matrix. In the case considered here, A is square and invertible and $\Sigma = 0$. Basic algebra shows that in that case, the above formula simplifies to:

$$\mathbb{E}[\boldsymbol{s}|\boldsymbol{x}] = A^{-1}\boldsymbol{x} \text{ and } \mathbb{E}[\boldsymbol{s}\boldsymbol{s}^\top|\boldsymbol{x}] = A^{-1}\boldsymbol{x}\boldsymbol{x}^\top A^{-\top} \ . \tag{3.17}$$

The EM update for A based on n samples $\boldsymbol{x}_1, \cdots \boldsymbol{x}_n$: then is

$$A^{\text{new}} = (\sum_{i=1}^n \boldsymbol{x}_i\mathbb{E}[\boldsymbol{s}|\boldsymbol{x}_i])(\sum_{i=1}^n \mathbb{E}[\boldsymbol{s}\boldsymbol{s}^\top|\boldsymbol{x}_i])^{-1} \ , \tag{3.18}$$

which yields $A^{\text{new}} = A$ by using Eq. (3.17). The EM algorithm is thus frozen in the case of no noise.

3.1.8 Proof of guaranteed descent

Let us demonstrate that one iteration of the incremental algorithm 25 decreases $\tilde{\mathcal{L}}_n$. At the iteration t, let W be the current unmixing matrix, U^{mem} the state of the memory, and the A^i's the current sufficient statistics. As said in Section 3.1.2, E-step, we have $A^i_{kl} = \frac{1}{n}\sum_{j=1}^n U^{\text{mem}}_{ij}X_{kj}X_{kl}$. Therefore, the algorithm is in the state (W, U^{mem}), and the corresponding loss is $\tilde{\mathcal{L}}_n(W, U^{\text{mem}})$. After the majorization step, the memory on the mini-batch is updated to minimize $\tilde{\mathcal{L}}_n$. Hence, the majorization step diminishes $\tilde{\mathcal{L}}_n$. Then, each descent move in the minimization step guarantees a decrease of $\tilde{\mathcal{L}}_n$. Both steps decrease $\tilde{\mathcal{L}}_n$, the incremental algorithm overall decreases the surrogate loss function.

3.1.9 Fast minimization step using conjugate gradient

The minimization step (3.10) involves computing the i-th row of the inverse of a given $p \times p$ matrix K. This amounts to finding $\boldsymbol{z}$ such that $K\boldsymbol{z} = \boldsymbol{e}_i$. Exact solution can be found by Gauss-Jordan elimination with a complexity $O(p^3)$. However, expanding the expression of K yields $K_{kl} = \frac{1}{n}\sum_j U_{ij}Y_{kj}Y_{lj}$, where $Y = WX$.

It follows that:

Lemma 3.4. *Assume that W is such that the rows of $Y = WX$ are independent. Then, $K_{kl} = O(\frac{1}{\sqrt{n}})$ for $k \neq l$.*

Proof In that case, $\mathbb{E}[\mathbf{u}_i\mathbf{y}_k\mathbf{y}_l] = 0$ since $\mathbf{y}_k$ and $\mathbf{y}_l$ are independent. Hence, the central limit theorem yields the advertised result. ■

Therefore, the matrix K is well approximated by its diagonal provided that n is large enough, and that the current signals Y are close enough from independence. As such, we use the diagonal of K as a *preconditioner* to the conjugate gradient technique for solving $K\mathbf{z} = \mathbf{e}_i$.

This gives an excellent approximation of the solution in a fraction of the time taken to obtain the exact solution.

Part II

SMICA: Spectral Matching Independent Component Analysis for M/EEG Brain Rhythms Separation

4

SMICA: spectral matching ICA for M/EEG processing

Contents

4.1 SMICA

This section contains unpublished work.

4.1.1 Introduction

Magnetoencephalography and Electroencephalography (M/EEG) are popular non-invasive techniques to record brain activity [Hämäläinen et al., 1993, Niedermeyer and Lopes da Silva, 2005]. They capture respectively the magnetic and electric signals produced by active neurons from the scalp surface or close to it. Due to the distance between the brain and the M/EEG sensors, these signals are a mixture of *sources*. The physics of the mixing is well understood: it is a linear process and can be considered instantaneous. It means that each sensor sees the sum of the contributions of all sources, and that there is no propagation delay.

What characterizes most neural sources at the origin of M/EEG signals is their spectral signature. Typically, brain sources are rhythmic signals, containing what is often referred to as *oscillations* [Buzsáki and Llinás, 2017]. Brain rhythms are an important concept in neuroscience: they are modulated during cognitive tasks, during sleep and can be used to design novel biomarkers (e.g. age, cognitive disorders) [Buzsáki, 2006].

Independent Component Analysis (ICA) [Hyvärinen et al., 2004] is extensively used in neuroscience for processing M/EEG signals [Makeig et al., 1997]. It is used to separate meaningful brain signals from artifacts (eye blinks, heartbeats, line noise, muscle, . . .) [Jung et al., 2000, Makeig et al., 1997]. ICA is in fact the most widely used method for EEG artifact rejection in the literature [Urigüen and Garcia-Zapirain, 2015]. What makes ICA quite remarkable it that it can identify those contamination sources 'blindly', that is, without prior knowledge of the underlying physics of the system (except linearity). Besides EEG, it is also widely used for the same purpose in MEG studies [Mantini et al., 2008, Vigário et al., 2000, Ikeda and Toyama, 2000, Dammers et al., 2008].

Beyond artifact removal, ICA is also used to reveal and study brain activity. In [Makeig et al., 2004], ICA is successfully applied to recover evoked and induced event-related dynamics in EEG signals. In [Gómez-Herrero et al., 2008], ICA is used to extract brain sources, on which causal relations are exhibited, uncovering directional coupling. In [Subasi and Gursoy, 2010], independent EEG sources are used in a machine learning pipeline, predicting epileptic seizures. ICA is used on MEG signals to identify links between function and structure in the brain in [Stephen et al., 2013]. It can also be used on MEG data, coupled with Hilbert filtering, to uncover resting state networks [Brookes et al., 2011].

Finally, linear ICA solutions can be used for source *localization*. Indeed, the linear coefficients needed to form a source can be represented as a topography on the scalp. An equivalent current dipole (ECD) can then be fitted to the topography [Scherg and Von Cramon, 1985], yielding at the same time an estimate of the source location, and its *dipolarity* (how close it can be explained by a focal activity in the brain modeled with a single dipole).

In order to unmix the M/EEG signals, ICA algorithms rely on an *independence* assumption between the sources. However, independence is a statistical principle which is difficult to quantify on real data. In practice, an ICA algorithm finds a linear transform of the sensor signals by optimizing some statistical criterion which can be shown to be maximized when the recovered sources are independent. Such criteria are designed to express some particular aspect of the statistical independence. These criteria are usually 'consistent' in the sense that their optimization solve the ICA problem when the data actually follow the model and a large number of samples is available. However, on real data of finite size, various criteria, capturing different aspects of independence, lead to different estimated sources.

In neuroscience, the most widely used algorithms are Infomax [Bell and Sejnowski, 1995] and FastICA [Hyvärinen, 1999]. These algorithms quantify independence on the marginal (instantaneous) distribution of the data: they ignore any time correlation and focus entirely on the non-Gaussianity of the data. Another route to ICA is to leverage the time correlation of the source signals. Blind separation is then possible if the sources show spectral diversity. Among these algorithms, Second Order Blind Identification (SOBI) [Belouchrani et al., 1997] is one of the most widely used. It jointly-diagonalizes of a set of time correlation matrices. Another approach closely related to our work consists in the joint diagonalization of spectral covariances [Pham and Garat, 1997]. Finally, In order to leverage both non-Gaussianity and spectral diversity, several methods based on short-time Fourier transform (STFT) have been proposed. For instance, Fourier-ICA [Hyvärinen et al., 2010] leverages both non-Gaussianity and spectral diversity with a hybrid method, consisting of the non-Gaussian ICA of concatenated short-time Fourier transforms. Other approaches consist in joint diagonalization of cospectral matrices (covariance matrices of STFT frames) [Congedo et al., 2008] or work in the wavelet domain [Pham and Cardoso, 2003].

While all these algorithms rely on various independence measures, they make the strong assumption that there are as many sources as sensors. The number of sensors is dictated by hardware, while the number of brain sources is a biological parameter; there is no reason to assume that they are equal. This is why, when fewer sources than sources are expected to represent the data, a dimension reduction technique like Principal Component Analysis (PCA) is often applied before preforming an ICA. However, this two stage approach, consisting of first PCA and then ICA, is heuristic. It is argued in [Ar-

toni et al., 2018] that applying PCA before ICA degrades the quality of the recovered sources. It is also sometimes suggested that, in order to estimate less sources than sensors, the data dimension should be reduced by discarding some channels rather than resorting to PCA. A natural but difficult question is the selection of the channels to preserve.

In this article, we study Spectral Matching ICA (SMICA) for M/EEG processing. This ICA model has been first investigated in astronomy for separation of the cosmic microwave background [Cardoso et al., 2002, Delabrouille et al., 2003]. SMICA measures independence in the frequency domain: It finds sources that are spectrally colored. This assumption makes it potentially well suited for brain rhythms and artifacts extraction as they are known to have prototypical spectra. Brain sources tend to exhibit so-called "1/f" power spectral densities, while the spectra of artifacts are often localized in certain frequency bands (e.g. muscle artifacts or line noise). Another potential benefit of SMICA is that it can naturally estimate fewer sources than sensors. Therefore, no preprocessing for dimension reduction is required. Finally, as SMICA only requires computing second-order statistics (spectral covariance matrices), it can handle gracefully very long recordings. This suggests that SMICA is well suited to uncover different generators of brain rhythms.

The article is organized as follows. Section 4.1.2 contains a brief review of usual ICA methods for M/EEG processing. In subsection 4.1.3, the SMICA statistical model is introduced and the estimation strategy based on an Expectation-Maximisation (EM) algorithm is described. In subsection 4.1.4, the usefulness of SMICA is demonstrated on various MEG and EEG datasets.

Notation The trace of a matrix $M \in \mathbb{R}^{p\times p}$ is $\mathrm{Tr}(M)$, and its determinant is $|M|$. A matrix is invertible when $|M| \neq 0$, and we write $M \in \mathrm{GL}_p$. Given a vector $u \in \mathbb{R}^p$, the matrix $\mathrm{diag}(u) \in \mathbb{R}^{p\times p}$ is the matrix containing the elements of u on its diagonal, and 0 elsewhere. If M is a $p \times p$ matrix, then $\mathrm{diag}(M)$ is the diagonal matrix with the same diagonal as M. The Moore-Penrose pseudo inverse of a tall matrix $A \in \mathbb{R}^{p\times q}$ is $A^\dagger = (A^\top A)^{-1}A^\top$. Finally, we define the Kullback-Leibler (KL) divergence between two positive definite matrices of size $p \times p$, C_1 and C_2, as

$$\mathrm{KL}(C_1, C_2) = \mathrm{Tr}(C_1 C_2^{-1}) - \log|C_1 C_2^{-1}| - p. \tag{4.1}$$

4.1.2 Background on Independent Component Analysis

The three routes to Independent Component Analysis

Let $X \in \mathbb{R}^{p\times T}$ a signal matrix, where each row of X corresponds to a signal. Here, p is the number of M/EEG sensors and T is the number of time samples. Independent Component Analysis, in its classical form, models the signals X as:

$$X = AS \ , \ A \in \mathbb{R}^{p\times p}, \ S \in \mathbb{R}^{p\times T} \ , \tag{4.2}$$

where A is called the *mixing matrix* and S the *source matrix*. The key assumption of ICA is that the rows of the source matrix S, the *sources*, are independent. The goal of ICA is to recover both A and S from the signals X, using only the independence assumption.

This model shows some natural indeterminacies: multiplying a source by a factor and dividing the corresponding column in A leads to the same signal matrix X. This is a scaling indeterminacy. Similarly, swapping two sources and the corresponding

columns in A leads to the same signal: this is a permutation indeterminacy. Scale and permutation ambiguities are intrinsic to ICA. Therefore, an ICA model is called *identifiable* if these are the only ambiguities to consider. Importantly, assuming the simplest model for the sources, i.e. that they are Gaussian i.i.d., leads to many more ambiguities. The model is then non-identifiable. Indeed, with independent Gaussian i.i.d. signals any rotation applied to the data still gives independent Gaussian i.i.d. signals: there is then a rotation indeterminacy.

There are three elementary assumptions that one can add on the sources in order to ensure identifiability [Cardoso, 2001]:

- *Non-Gaussianity*: the sources are still assumed i.i.d, but their probability density function can deviate from Gaussianity. In [Comon, 1994], it is shown that non-Gaussianity provides identifiability, under the assumption that at most one source is Gaussian. It is the most standard setting for ICA: FastICA [Hyvärinen, 1999] and Infomax [Bell and Sejnowski, 1995] both fall in this category. Most ICA algorithms for M/EEG signal processing use this source model, because most brain sources are far from Gaussian.

- *Non-Stationarity*: the sources are Gaussian, but no longer stationary. The variance of the sources varies with time. The problem is identifiable if no two variance profiles are proportional [Pham and Cardoso, 2001].

- *Time correlation*: the sources are Gaussian stationary, but temporally correlated. It implies that sources typically do not have non-flat spectra. The problem is identifiable if no two spectra are proportional [Pham, 2001a], i.e. if the sources are 'spectraly' diverse.

Figure 4.1 illustrates the three routes to ICA. This article focuses on the last model. In practice, these assumptions can seem too strong for real signals: indeed, no M/EEG brain source perfectly matches either of the above assumptions. They usually look quite different from the prototypical sources of figure 4.1. Still, if they verify the type of diversity required by any of the three routes, they can still be blindly recovered. For instance, following the third route, ICA algorithms which aim to recover mixture of time correlated Gaussian stationary sources will recover sources which are not stationary nor Gaussian, provided that they have non-proportional spectra. Of course, this comes with a drop in statistical efficiency with respect to a model capturing the full statistical structure, but identifiability is still guaranteed.

Standard algorithms for recovering time-correlated sources

We now review two of the most usual ICA algorithms based on time correlation.

Second Order Blind Indentification (SOBI)

Second Order Blind Identification (SOBI) [Belouchrani et al., 1997] aims at separating time-correlated sources using correlation matrices:

$$R(\tau) = \frac{1}{T-\tau} \sum_{i=0}^{T-\tau} X(t) X^\top (t+\tau) \tag{4.3}$$

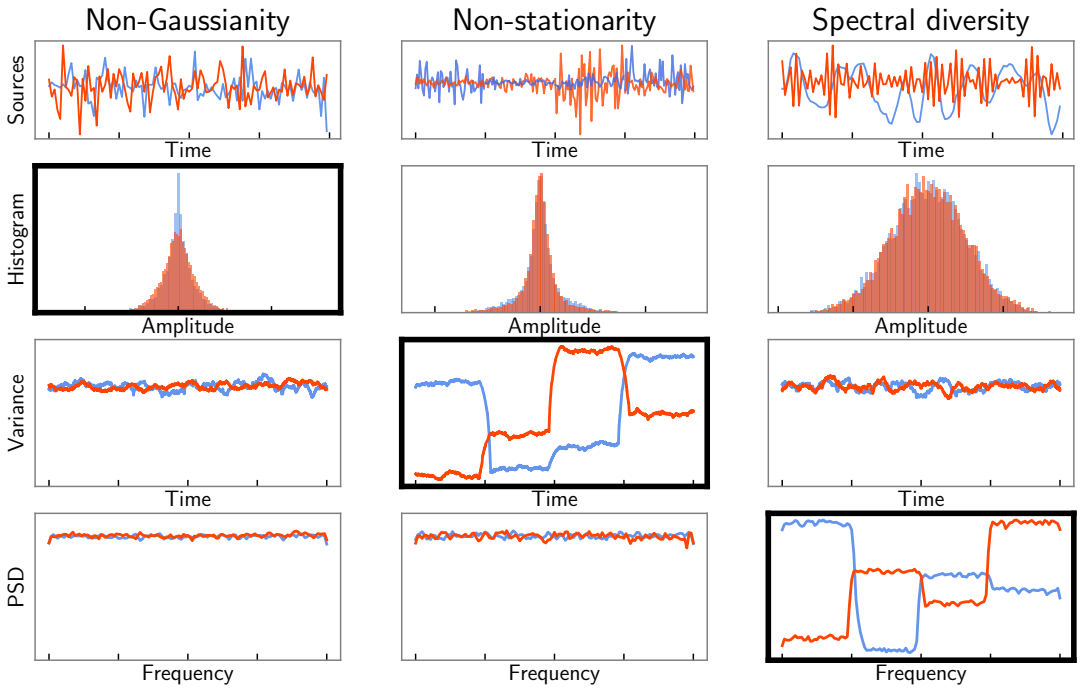


Figure 4.1 – This figure illustrates the different routes to achieve blind deconvolution under independence assumptions. The first row shows different "prototypical" source profiles. The second row shows the corresponding histograms. The third row shows the instantaneous variance of the sources across time. The last row shows the power spectral density (PSD) of the source. These signals are synthetic, therefore the scales are arbitrary and irrelevant. The first route (left column) is the non-Gaussian ICA model where each source (except maybe one) distribution is assumed to be non-Gaussian (either sub- or super-Gaussian). These sources can be independent and identically distributed, leading to constant variance and power spectra. The second route (middle) relies on non-stationary assumptions: each source is assumed to have a time-varying variance that evolves independently for each source. Finally, the third route (right) to ICA explored in this work assumes that each source have different spectral contents. Sources following the second route -non-stationarity- are usually also non-Gaussian as illustrated by the histogram, hence non-Gaussian algorithms can recover them. These algorithms fail on the last type of sources, which can be Gaussian. The only source signals that cannot be separated by any ICA methods are Gaussian white noise, which have a Gaussian density, a constant variance profile and a flat power spectrum density.

computed for a set of time lags $\mathcal{T} = \{\tau_1, \ldots, \tau_B\}$. Under the ICA model $X = AS$, these matrices have the structure:

$$R(\tau) = A\Lambda(\tau)A^\top, \tag{4.4}$$

where $\Lambda(\tau) = \frac{1}{T-\tau}\sum_{i=0}^{T-\tau} S(t)S^\top(t+\tau)$ is (close to) a diagonal matrix for independent sources. Hence, the mixing matrix A appears as an *joint-diagonalizer* of the set of matrices $R(\tau)$ for $\tau \in \mathcal{T}$. In other words, the mixing matrix A is such that the matrices $A^{-1}R(\tau)A^{-\top}$ for $\tau \in \mathcal{T}$ are all diagonal.

In practice, SOBI proceeds in two steps: a whitening step (which enforces the decorrelation of the recovered sources) followed by a rotation. Hence, a separation matrix $B = A^{-1}$ is found in the form $B = UW$ where W is a whitening matrix, *i.e.*, any matrix such that $WR(0)W^\top = I$, and where the rotation U is determined by minimizing a joint diagonalization criterion of the whitened matrices $WR(\tau_i)W^\top$

$$U = \underset{U^\top U = I}{\arg\min}\, \mathcal{L}_{SOBI}(U) \triangleq \sum_{i=1}^{B} \mathrm{Off}(UWR(\tau_i)W^\top U^\top) \ , \tag{4.5}$$

where the Off of a matrix is defined by: $\mathrm{Off}(M) = \sum_{a\neq b} M_{ab}^2$. This method is appealing for the simplicity of its implementation, because there exists a variety of approximate joint-diagonalization algorithms that aim at solving exactly problem (4.5) [Ziehe et al., 2004, Cardoso and Souloumiac, 1996]. However, it is not derived from a clear statistical framework based on a probabilistic model. For instance, the whiteness constraint can have adverse effects on the quality of the decomposition [Cardoso, 1994]. Also, the simple whitening process by an inverse sqare root of $R(0)$ would be biased for a significant noise level. Probably more importantly, the off-diagonal errors involved in the joint-diagonalization criterion 4.5 are simply co-added even though those terms are far from being statistically independent. In section 4.1.2, we examine a frequency-domain approach which greatly mitigates the later problem and we show in section 4.1.3 how it is easily adapted to properly take into account noise contamination.

Joint-diagonalization (JDIAG)

In [Pham, 2001a], it is proposed to separate time-correlated sources by exploiting *spectral diversity*. It assumes a noiseless ICA model $X = AS$ with A of size $p \times p$. The sources are assumed independent and stationary. Let $f_1 < \cdots < f_{B+1}$ some frequencies, and denote X_i (resp. S_i) the signals X (resp the sources S) filtered in the frequency band $[f_i, f_{i+1}]$. Since filtering is a linear operations, the ICA equation (4.2) yields $X_i = AS_i$. JDIAG assumes a Gaussian i.i.d. model for the filtered sources:

$$S_i \sim \mathcal{N}(0, P_i) \ , \tag{4.6}$$

where P_i is a diagonal matrix. The negative log-likelihood of the model then writes:

$$\mathcal{L}_{JDIAG} = \sum_{i=1}^{B} n_i \,\mathrm{KL}(\hat{C}_i, C_i) \ , \tag{4.7}$$

where n_i is the number of DFT frequencies in the band $[f_i, f_{i+1}]$, $\hat{C}_i = \frac{1}{T}X_iX_i^\top$ is the empirical covariance in the i-th band, $C_i = AP_iA^\top$ is the predicted covariance from the model, and the KL divergence, defined in eq. (4.1), measures the distance between two Gaussians $\mathcal{N}(0, \hat{C}_i)$ and $\mathcal{N}(0, C_i)$.

This model forms the basis of the SMICA approach detailed in the next section; it will be more detailed there.

For A fixed, it is a simple matter to show that $\mathcal{L}_{JDIAG}$ is minimized for $P_i = \mathrm{diag}(A^{-1}\hat{C}_i A^{-\top})$; therefore, minimization of $\mathcal{L}_{JDIAG}$ is obtained, after a little of calculus, by finding:

$$A \in \underset{A \in \mathrm{GL}_p}{\arg\min} \sum_{i=1}^{B} n_i \log \left(\frac{|\mathrm{diag}(D_i)|}{|D_i|} \right), \quad \text{with} \quad D_i = A^{-1}\hat{C}_i A^{-\top} \ . \tag{4.8}$$

This is a classical joint-diagonalization criterion [Pham, 2001b], for which fast algorithms exist [Ablin et al., 2019a]. Although this method is backed by stronger statistical arguments than SOBI, and aims at recovering the same type of sources, it is seldom used in neuroscience.

Dimension reduction

The ICA model $X = AS$ does not include a noise term, and assumes that the mixing matrix A is square, i.e. that the number of sources equals the number of sensors. Arguably, the number of sources should be a biophysical parameter, while the number of sensors is dictated by the recording device. For this reason, it is customary to perform a pre-processing step to reduce the number of signals in the data matrix to the desired number of sources. The most widely used method for this task is Principal Component Analysis (PCA). It finds a matrix $W_{\mathrm{PCA}} \in \mathbb{R}^{q \times p}$, with $q < p$, such that the rows of $X_{PCA} \triangleq W_{PCA} X$ are decorrelated. Then, ICA is applied on X_{PCA}.

Although very popular, this two-stage approach is often criticized [Artoni et al., 2018]: it does not come from a principled statistical framework, and PCA returns signals of largest variance, hence it might destroy some low power but interesting sources that are present in the original dataset. Therefore, it is often advised to run ICA on the whole dataset, leading to long computations (see Sec. 2.1) and to some spurious sources looking like Gaussian noise.

ICA models with noise?

One must resort to PCA for finding fewer sources than sensors because the mixing matrix in ICA is assumed square. An ICA model with fewer sources than sensor ($A \in \mathbb{R}^{p \times q}$ with $q < p$) must assume that there is some additive noise *on the sensors*:

$$X = AS + N \ , A \in \mathbb{R}^{p \times q}, \ \ S \in \mathbb{R}^{q \times T}, \ \ N \in \mathbb{R}^{p \times T} \ , \tag{4.9}$$

where N is a noise matrix. Typically, one assumes that the noise is Gaussian i.i.d. Unfortunately, model (4.9) is notoriously hard to fit in the *non-Gaussian* setting, because the likelihood does not have a closed-form equation [Girolami, 2001, Bermond and Cardoso, 1999, Moulines et al., 1997]. Therefore, to the best of our knowledge, such model has never been applied in a neuroscience setting.

4.1.3 Spectral Matching Independent Component Analysis

We present the spectral matching ICA, which leverages spectral diversity in sources and includes a noise model. Contrary to the non-Gaussian setting, all random variables are modelled as Gaussian, yielding a tractable likelihood.

The SMICA model

In the following, let $X \in \mathbb{R}^{p \times T}$ be the M/EEG recording of interest, where p is the number of sensors and T is the number of samples. We assume a linear ICA model *with noise*, and with q sources:

$$X = AS + N \ , \tag{4.10}$$

where $A \in \mathbb{R}^{p \times q}$, $S \in \mathbb{R}^{q \times T}$ and $N \in \mathbb{R}^{p \times T}$. We assume that the time samples in S and N follow a Gaussian density, and that the sources (rows of S) and the noise channels (rows of N) are independent.

Define $\hat{S}$ the discrete Fourier transform of S:

$$\hat{S}_{jl} = \frac{1}{\sqrt{T}} \sum_{t=1}^{T} S_{jt} \exp(-\frac{2\pi i l t}{T}) \ \text{ for } \ j = 1 \dots q, \ \ l = 1 \dots T \ . \tag{4.11}$$

Under mild assumptions, these coefficients are Gaussian and independent as T goes to infinity, and their variance is given by the power spectrum $p_j(l) = \sum_\tau \mathbb{E}[S_j(t) S_j(t+\tau)] \exp(-\frac{2\pi i l t}{T})$: the $\hat{S}_{jl}$ are random independent variables following $\mathcal{N}(0, p_j(l))$. Although this result is asymptotic (valid only as T goes to infinity), we make the assumption that it holds for finite T: this is the *Whittle approximation*. We make the same assumption on the noise, denoting $\sigma_j^2(l)$ its power spectrum for the Fourier index l.

Following eq. (4.10), the spectral covariance of the signals X is given by:

$$C(l) = AP(l)A^\top + \Sigma(l) \ , \tag{4.12}$$

where $P(l) = \mathrm{diag}(p_1(l), \dots p_q(l))$ and $\Sigma(l) = \mathrm{diag}(\sigma_1^2(l), \dots, \sigma_p^2(l))$ are diagonal matrices. These powers are unknown and should be estimated: along with the mixing matrix A, they are the parameters of the model. However, in order to reduce the size of the model and to regularize it, we approximate the spectra as being *piecewise constant* over narrow spectral intervals. Let $B+1$ frequencies $f_1 < \dots < f_{B+1}$, and consider the frequency intervals $F_i = [f_i, f_{i+1}]$. Let f_s the sampling frequency. We assume that P and Σ are constant in each of these bands, i.e. that $P(l) = P_i = \mathrm{diag}(p_{i,1}, \dots, p_{i,q})$ and $\Sigma(l) = \Sigma_i = \mathrm{diag}(\sigma_{i,1}^2, \dots, \sigma_{i,p}^2)$ when $l \in [\frac{f_s}{f_{i+1}}, \frac{f_s}{f_i}]$. The model of the spectral covariance matrices then reads in each band:

$$C_i = AP_iA^\top + \Sigma_i \ . \tag{4.13}$$

In practice, the empirical covariance matrices $\hat{C}_i$ are computed and the theoretical formula (4.13) is adjusted to fit them. Figure 4.2 illustrates this *spectral matching* principle.

Statistical properties of the model

We now discuss a few statistical properties of the SMICA model.

Likelihood The negative log-likelihood of this model is given by:

$$\sum_l \mathrm{KL}(\hat{C}(l), C(l)) \ , \tag{4.14}$$

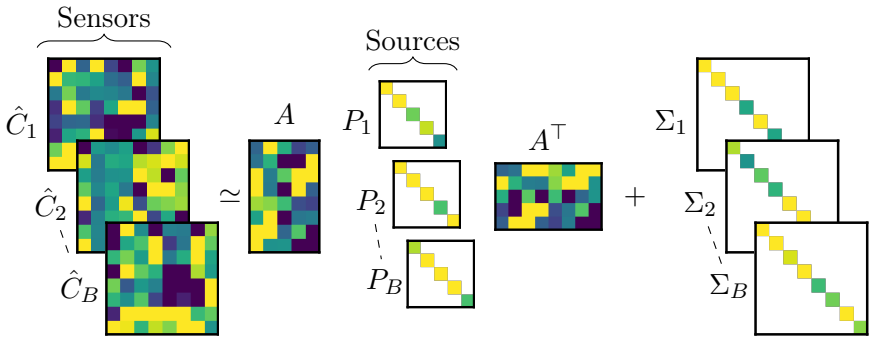


Figure 4.2 – The spectral matching principle: empirical spectral covariances $\hat{C}_1, \ldots, \hat{C}_B$ are computed from the M/EEG data. SMICA matches those covariances with the model $\hat{C}_i \simeq AP_iA^\top + \Sigma_i$, where the P_i and Σ_i are diagonal positive matrices. Matching is performed in a statistically sound way, by maximizing the likelihood of the model.

where $\hat{C}(l)$ is the empirical power spectrum computed from the data, and KL is the Kullback-Leibler divergence. Following the piecewise-constant spectra assumption, it is written as a sum over the B frequency bands:

$$\mathcal{L} = \sum_{i=1}^{B} n_i \, \mathrm{KL}(\hat{C}_i, C_i) \ , \tag{4.15}$$

where n_i is the number of Fourier frequency bins in the i-th band. As such, it is a measure of mismatch between the spectral covariance matrices obtained from the data $\hat{C}_i$ and predicted by the model C_i. Since it decomposes as a sum over frequency bands, it can be used for model fit inspection, for instance to determine if some spectral domain exhibit a poor mismatch.

Source estimation: Wiener filtering Noiseless ICA models, $X = AS$, have a simple way of computing the estimated sources: $\hat{S} = \hat{A}^{-1}X$.

SMICA can employ the same technique for recovering the source, albeit replacing A^{-1} by $A^\dagger$ (the pseudo-inverse of A). Still, the model of SMICA allows for a finer source recovery, through *Wiener filtering* ([Brown et al., 1992], Chapter 4). It is more appealing to compute the expected value of the sources given the parameters: $\hat{S} = \mathbb{E}[S|X, \Theta]$, which is given by Wiener filtering:

$$\hat{S}_i = (A^\top \Sigma_i^{-1} A + P_i^{-1})^{-1} A^\top \Sigma_i^{-1} X_i \ . \tag{4.16}$$

This operation is linear in X, and is adaptive to the level of noise: if in frequency band i the estimated noise Σ_i for a sensor is large, then its contribution in the source estimate shrinks towards 0. Note that the standard ICA source estimation formula is recovered when the noise is equal on all channels and tends to 0 (i.e. $\Sigma_i = \lambda_i I_p$ with $\lambda_i \to 0$), yielding at the limit $\hat{S} = A^\dagger X$.

Signal denoising using SMICA Like any ICA algorithm, SMICA estimate sources which can be marked as spurious / non-biological by specialists. The remaining sources can then be projected back in the signal space, giving cleaned M/EEG signals. Thanks to its noise modeling, SMICA makes these two operations statistically sound. Furthermore, SMICA offers another data-cleaning opportunity, which boils down to removing sources identified as noise like for any ICA procedure.

Given sources $\hat{S}$ obtained by Wiener filtering, some sources can be manually or automatically marked as artifactual. By marking source i as an artifact, we mean that we set $\hat{S}_i = 0$. Finally, the cleaned signals are computed as $X_{cleaned} = A\hat{S}$.

Combining SMICA with non-Gaussian ICA As explained in section 4.1.2, SMICA cannot separate sources which have proportional spectra. In practice, it may happen that several brain sources have very similar spectra. Therefore, one can use non-Gaussian ICA on the sources that have similar spectra to further separate them.

SMICA can also be used as a *source subspace* identifier, by applying non-Gaussian ICA on the whole estimated source matrix. The hope is that the resulting dimension reduction is more meaningful than PCA, and that Wiener filtering helps cleaning the signals. The practical benefits of such approach are demonstrated in the experiments presented in section 4.1.4.

Fitting the model: EM algorithm

The model is fitted using the Expectation-Maximization (EM) algorithm [Dempster et al., 1977]. This algorithm is appealing because it does not require hyperparameters like learning rates, and is guaranteed to decrease the loss function at each iteration. Still, this approach is generally slow (it might require many iterations to reach a satisfactory set of parameters), and other optimization techniques could be investigated for the fast minimization of $\mathcal{L}$.

The EM algorithm for SMICA is described in [Cardoso et al., 2002] with a slightly more constrained noise model. For completeness, we derive it in the appendix (section 4.1.5).

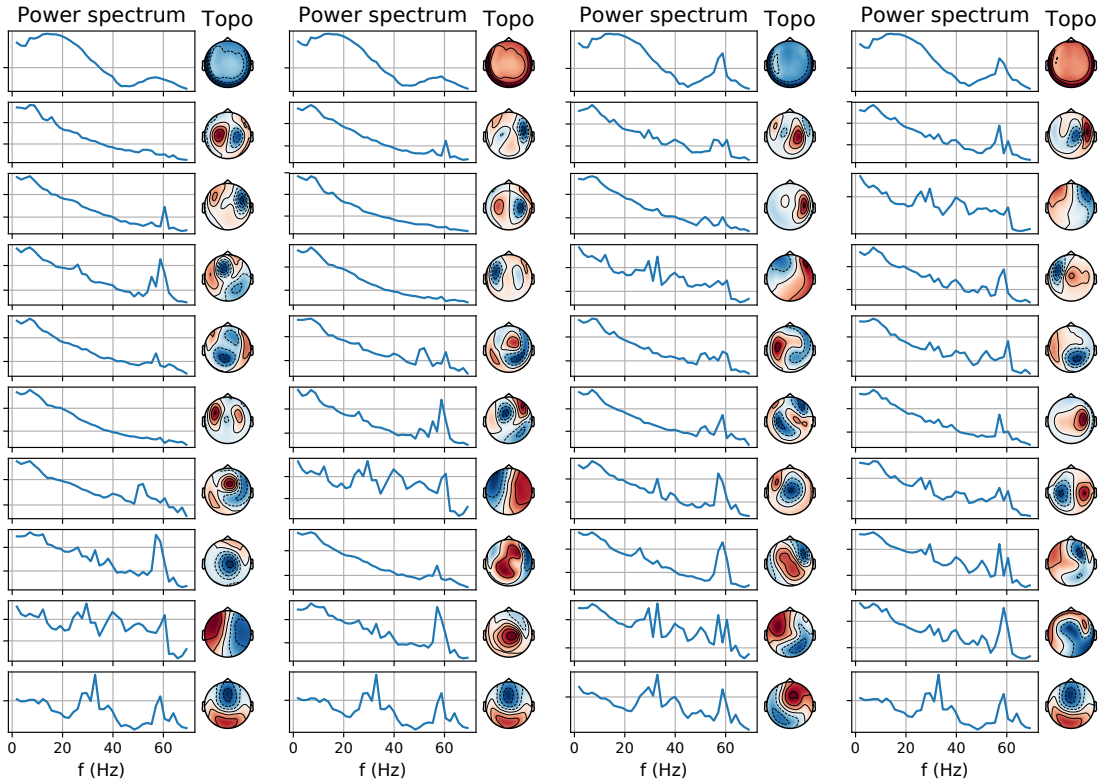
Complexity of SMICA SMICA only needs the spectral covariances $C_1, \ldots, C_B$ to infer the parameters of the model. Therefore, the complexity of the EM algorithm does not depend on the number of samples T; only the computation of the covariances does. This differs from non-Gaussian ICA algorithms, for which the fitting time depends a lot on the length of the recordings.

In practice, we found that fitting SMICA on a 102 sensors MEG dataset with 40 frequency bins and 102 sources takes about 15 minutes using one CPU of a recent laptop. The cost is much higher than that of JDIAG, for which this operation usually takes less than a minute.

4.1.4 Experiments

In this section, we apply SMICA on different datasets, in order to illustrate its application for analysis of M/EEG data. We use Infomax as a reference non-Gaussian ICA method but we use the Picard algorithm(Sec. 2.1) for fast and robust optimization of the Infomax objective. This algorithm is run with the tanh non-linearity. The joint-diagonalization algorithm for SOBI is a combination of [Ziehe et al., 2004] with a backtracking line-search. The joint-diagonalization algorithm for JDIAG uses the fast implementation described in [Ablin et al., 2019a]. The M/EEG analysis is carried out using the MNE package [Gramfort et al., 2014].

In all experiments we set the frequency bins F_i of SMICA and JDIAG as uniform in the range $1 - 70Hz$, with 40 bins.



(a) SMICA (b) JDIAG (c) SOBI (d) Infomax

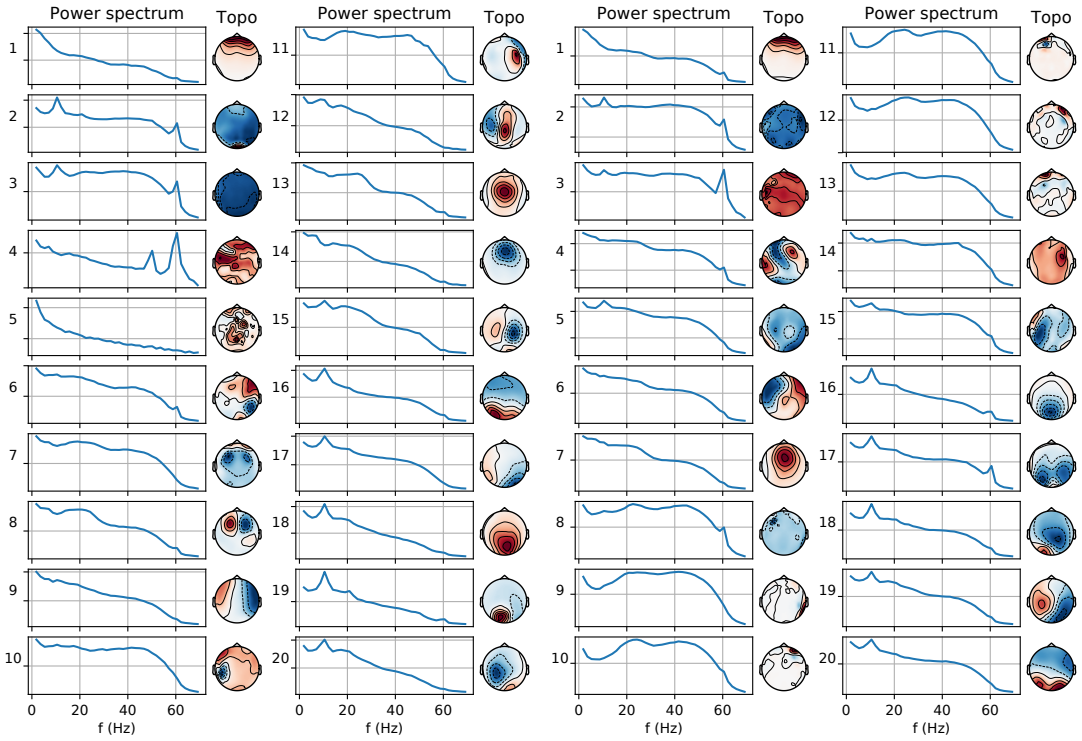
Figure 4.3 – Different ICA decompositions on MEG data. Most sources found by SOBI and Infomax are contaminated by environmental noise, which is especially visible in the 20-40 Hz band. There is also a strong peak around 60 Hz for most sources, including in the heartbeat source (first row). Sources found by SMICA and JDIAG are very similar, and both well separate ECG and environmental noise from the other brain sources.

Decomposition example on a MEG dataset

We start by showing the decomposition found by SMICA, JDIAG, SOBI and Infomax on a MEG dataset, where the subject was presented checkerboard patterns into the left and right visual field, as well as monaural auditory tones to the left or right ear. Stimuli occurred every 750 ms (See [Gramfort et al., 2014] for a description of the dataset). MEG is acquired with 102 magnetometers and 204 gradiometers.

For this experiment, we only consider the 102 magnetometer channels. Each ICA algorithm returns 10 sources (after PCA for Infomax, SOBI or JDIAG). The power spectrum and topography of these sources are displayed in figure 4.3.

For this dataset, the decompositions of SMICA and JDIAG are quite similar. They successfully isolate environmental noise (last two components of SMICA). It also appears that the components of SOBI and Infomax are of lesser quality, since they seem artifacted by noise in the band 20-40 Hz. All algorithms successfully isolate heartbeats, but these components are slightly contaminated by 60 Hz line noise for SOBI and Infomax.



(a) SMICA (b) JDIAG

Figure 4.4 – Comparison of SMICA and JDIAG on an EEG dataset. Both algorithms are asked to recover 20 sources. Both algorithms recover the eye blinks (source 1). SMICA manages to isolate what seems to be line signal (source 4, peak at 60 Hz). The other peak at 50 Hz may be due to aliasing after resampling of the raw data. JDIAG does not recover a source with such a high 60 Hz peak. As a consequence, more sources found by JDIAG present a small peak at 60 Hz. JDIAG also isolates some noisy source (sources 8 - 13, right), which have topographies centered around one sensor. SMICA does not find such sources, and instead recovers more biologically plausible brain sources (sources 8-20, left).

Example on an EEG dataset

We run SMICA and JDIAG on a 69-channel EEG data coming from the dataset described in [Delorme et al., 2012]. Both algorithms return 20 sources, which are displayed in figure 4.4. Differences between SMICA and JDIAG are now more striking which might be caused by the greater noise level compared to the MEG recording.

Although decompositions differ in many aspects, we want to focus on source 4 recovered by SMICA, which is not found by JDIAG. There is a sharp peak at 60 Hz, indicating that it likely corresponds to line noise. The second peak at 50 Hz may seem spurious but is most probably due to spectral aliasing. Indeed the the fifth harmonic of a line at 60 Hz when sampled at 250 Hz appears at frequency 5*60-250 = 50 Hz. To test whether it is a plausible line source, we resort to a separate experiment: we determine the weight vector $w \in \mathbb{R}^{59}$ such that the time series wX is of power 1 with a maximal power at 60 Hz. We use a method in the spirit of [Nikulin et al., 2011]. To do so, we filter X in a narrow band around 60 Hz, yielding signals X_f. Then, we find w by maximizing the power of wX_f under the constraint that the power wX is 1. This is done by maximizing the Rayleigh quotient between the covariance of X_f and of X.

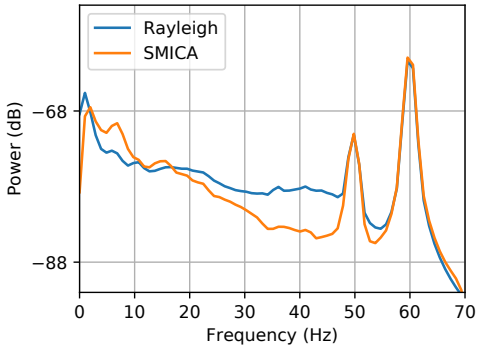


Figure 4.5 – Comparison between the spectra of source number 4 found by SMICA on an EEG dataset (figure 4.4) and a Rayleigh quotient method [Nikulin et al., 2011] aiming at recovering the source with maximal power at 60 Hz. Both sources exhibit another strong peak at 50 Hz, suggesting that this is not an artifact created by SMICA

The power spectrum of the corresponding source wX is displayed in figure 4.5, along with the power spectrum of the source number 4 found by SMICA in figure 4.4. The 50 Hz aliased harmonic is also recovered by Rayleigh quotient, suggesting that the source recovered by SMICA isolates correctly the line signal.

A data driven external interference suppression

Figure 4.3 suggests that the spectral methods (SMICA and JDIAG) remarkably isolate environmental noise from the rest of the signal, while other methods fail to do so. Therefore, spectral ICA can be used for environmental noise cancellation in MEG signals. We now compare it to two widely used methods for this task, Source Subspace Projection (SSP) [Uusitalo and Ilmoniemi, 1997] and Signal Space Separation (SSS), a.k.a. Maxwell filtering, [Taulu, 2006]. The idea behind this is that SMICA captures the sources that these methods are trying to suppress, thereby providing a clear statistical framework for noise suppression.

Figure 4.6 shows the global field power of raw and filtered signals with the compared methods, for four different experimental conditions. For SMICA and Infomax, the cardiac source and two sources corresponding to environmental noise are labeled by hand, and then suppressed using method described in section 4.1.3.

SMICA finds highly dipolar source subspaces

In this section, we illustrate the ability of SMICA to capture meaningful brain sources. We use the same dataset as in [Delorme et al., 2012]. It contains the EEG recording of 15 subjects, with 69 EEG channels. For a target number of independent sources, different ICA procedures described in the article are compared. If the number of sources is lower than the number of sensors (69), classical ICA methods must undergo a preprocessing step, typically PCA, yielding a matrix W_{PCA} of size $n_{sources} \times n_{sensors}$. Then, both Infomax and JDIAG are applied on the signals $X_{PCA} = W_{PCA}X$, yielding square unmixing matrices $W_{Infomax}$ and W_{JDIAG} of size $n_{sources} \times n_{sources}$. Finally, the estimated mixing matrix for these methods, linking the signals to the sources, are:

$$A_{Infomax} = (W_{Infomax}W_{PCA})^{\dagger}, \quad A_{JDIAG} = (W_{JDIAG}W_{PCA})^{\dagger} \ .$$

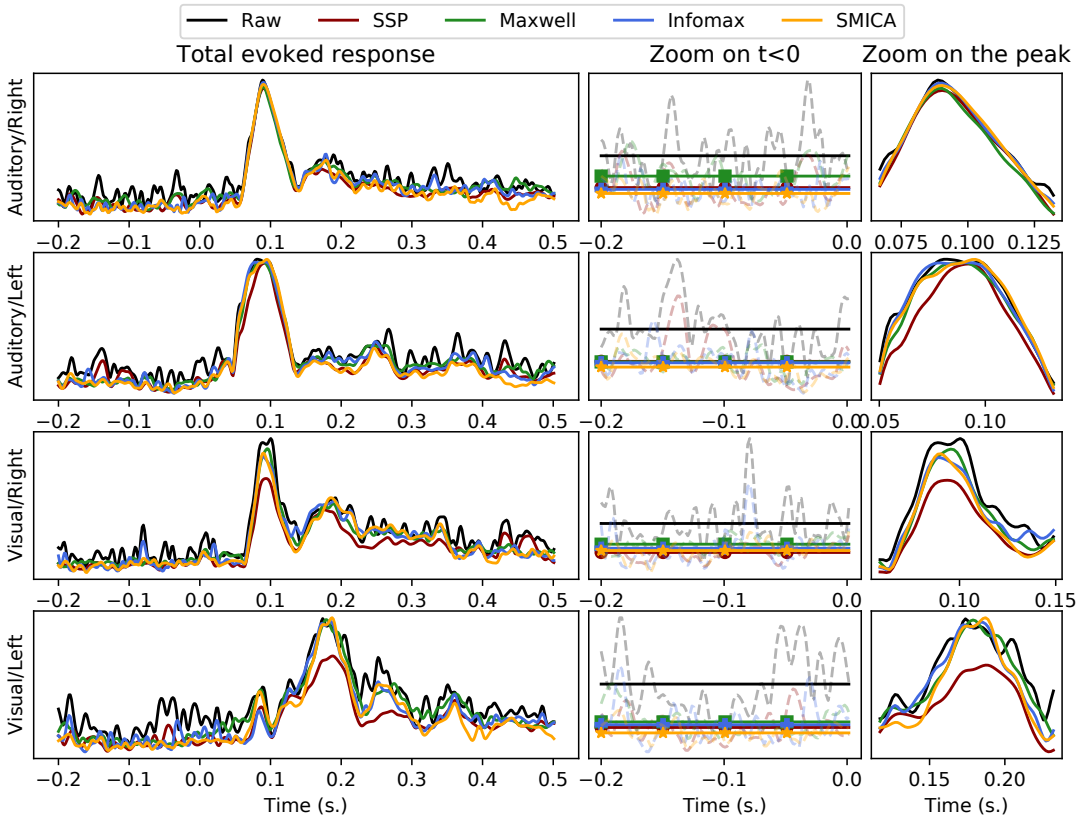


Figure 4.6 – Comparison of denoising methods. We display the global field powers (averaged evoked potential power) for four experimental conditions (from top to bottom): auditory right, left, and visual right, left stimulation. The raw averaged signal is displayed in black. We compare this with Signal Subspace Projection (SSP), Signal Space Separation (SSS), Infomax and SMICA. Left: averaged signals displayed during pre- and post-stimulation periods. Middle: zoom on the pre-stimulus time segment. Full lines display the average of the global field powers on these segments, while dashed lighter lines show the true signals. Right: zoom on the peak. For Infomax and SMICA, three sources have been marked as "bad": two sources coming from the noise room with spurious spectra (for SMICA, the two last sources of figure 4.3), and the ECG (first source of figure 4.3). SMICA offers slightly superior noise reduction in the $t < 0$ segment than SSS, SSP and Infomax, and preserves accurately the peak at around $100ms$, offering a higher signal-to-noise ratio.

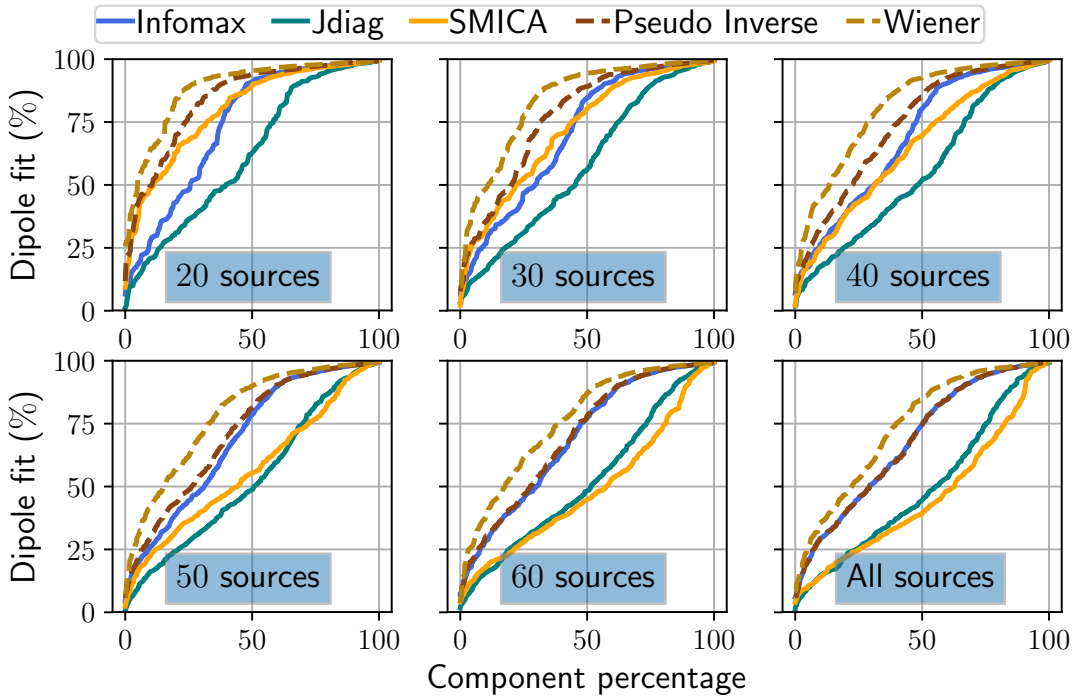


Figure 4.7 – Dipolarity of ICA sources found by different methods with 20, 30, 40, 50, 60, and 69 sources, on 15 EEG datasets of containing 69 sensors. Dipole fit corresponds to the goodness of fit obtained after fitting a dipole to the source location. For Infomax and JDIAG, a PCA is first applied on the data matrix to obtain the desired number of channels. SMICA works directly on the whole dataset. Pseudo-inverse and Wiener correspond to further applying Infomax on the sources recovered by SMICA, either by the pseudo-Inverse or Wiener filtering method. In this example, spectral diversity methods such as SMICA and JDIAG exhibit lower dipolarity. However, it appears that SMICA finds a better signal subspace than PCA, since performing non-Gaussian ICA in this subspace always yields higher dipolarity than out-of-the-box Infomax. Wiener filtering also helps finding better sources, and provides the overall best results.

SMICA is applied on the raw signals X, and directly estimates a mixing matrix A_{SMICA} of size $n_{sensors} \times n_{sources}$. Then, the sources of SMICA are estimated with the pseudo-inverse and Wiener methods, yielding two sets of source signals S_{inv} and S_{Wiener} with $n_{sources}$ signals. Infomax is finally applied on these sources, yielding two square unmixing matrices W_{inv} and W_{Wiener}. The mixing matrices corresponding to the combination of SMICA and Infomax are obtained as:

$$A_{inv} = A_{SMICA} W_{inv}^{\dagger}, \quad A_{Wiener} = A_{SMICA} W_{Wiener}^{\dagger} \ .$$

The dipolarity corresponding to these 5 ICA mixing matrices ($A_{Infomax}$, A_{JDIAG}, A_{SMICA}, A_{pinv} and A_{Wiener}) are finally computed, yielding $n_{sources}$ scalars per subject. The dipole misfits are then aggregated across all subjects, and sorted. Results are displayed in figure 4.7.

We observe that when a small number of sources is required (e.g. 20), SMICA finds more dipolar sources than methods using PCA (JDIAG and Infomax). For a high number of sources, SMICA falls behind these methods. Applying Infomax on sources recovered by SMICA either by pseudo inversion or Wiener filtering always increases the

dipolarity of the decomposition. Overall, SMICA with Wiener yields greater dipolarity for all number of components, which makes it an algorithm of choice following [Delorme et al., 2012].

4.1.5 Discussion

Non-Gaussian ICA is routinely applied by many practitioners on EEG, MEG and even fMRI [Beckmann et al., 2005, Varoquaux et al., 2010]. In this article, we argued that using instead *spectral diversity* can offer an interesting alternative. Specifically, we proposed to use the SMICA algorithm, which leverages spectral diversity in a statistically sound manner. SMICA benefits from a noise model enabling accurate source estimation (through Wiener filtering), and the estimation of a smaller number of sources than sensors, without resorting to ad-hoc dimension reduction methods such as principal component analysis.

Results in figure 4.3 have shown that SMICA can better isolate stationary environmental noise than non-Gaussian ICA methods. This can be explained by the fact that noise and brain sources usually have very different spectra. Noise tends to have peaks or bumps in the spectra in higher frequencies while neural sources have spectra with power laws or exponential decays, so called $1/f$ spectral densities [Buzsáki and Draguhn, 2004, Dehghani et al., 2010, Rocca et al., 2018]. As a consequence, this method is well suited as noise suppressor, as demonstrated in figure 4.6. It also very clearly reveals physiological sources with $1/f$ power spectrum making decisions about what are artifacts and what are not easier.

Thanks to the noise model, we have a principled way to perform dimension reduction and to recover the sources time course. In [Artoni et al., 2018], it is argued that PCA is suboptimal for EEG data, and that even channel subsampling is to be preferred. Figure 4.7 shows that SMICA identifies a source subspace that contains more dipolar components than PCA. As such, SMICA can be a useful tool for dimension reduction.

As an important remark, as illustrated in figure 4.3, SMICA is very similar to JDIAG for clean data, since the noise subspace in this case is simple to find. Therefore, SMICA is more likely to be useful for noisy datasets such as clinical recordings.

This work is not the first using and highlighting the benefits of spectral ICA for M/EEG recordings. Some works have focused on convolutional mixtures (for which using the Fourier domain turns convolutions into products). In [Dyrholm et al., 2007], authors use convolutive ICA in time-domain, while in [Anemuller et al., 2003] they use a complex Infomax to find travelling waves. In the end, it boils down to estimating one mixing matrix for each frequency bin. However, independent models in each band might fail to recover brain rhythms with several frequency peaks. For instance, mu-rhythm is characterized by concurrent activity near 10 Hz and 20 Hz [Hari and Salmelin, 1997, Niedermeyer and Lopes da Silva, 2005].

Some improvements to the current SMICA algorithm can be investigated. First, since it comes from a principled statistical framework, it would be interesting to implement a data-driven way of computing the most likely number of sources in the data: an algorithm to automatically select the correct number of sources. However, preliminary experiments show that usual statistical criteria like Akaike Information Criterion or Bayesian Information Criterion are not satisfactory in this setting, likely because the model is not complex enough to fit fully such complex signals. The EM algorithm for fitting SMICA is also quite slow, some improvements could be possible by further

studying the geometry of the cost function and proposing quasi-Newton algorithms, as done recently for Infomax(Sec. 2.1).

Appendix

The EM algorithm for SMICA

The parameters are $\Theta = \{A, \Sigma_1, \ldots, \Sigma_B, P_1, \ldots, P_B\}$, and the latent variables are the sources in each frequency bands $S_1, \ldots, S_B$.

E-step At the E-step, the sufficient statistics of the model are computed. Since the model is Gaussian, they are simply the second-order statistics: $\mathbb{E}[S_i S_i^\top|\Theta], \mathbb{E}[S_i X_i^\top|\Theta]$ and $\mathbb{E}[X_i X_i^\top|\Theta]$. In the following, let $\Gamma_i = (A^\top \Sigma_i^{-1} A + P_i^{-1})^{-1}$ and $W_i = \Gamma_i A^\top \Sigma_i^{-1}$ the Wiener filter. We have:

$$R_i^{XX} \triangleq \mathbb{E}[X_i X_i^\top|\Theta] = \hat{C}_i \tag{4.17}$$

$$R_i^{SX} \triangleq \mathbb{E}[S_i X_i^\top|\Theta] = W_i \hat{C}_i \tag{4.18}$$

$$R_i^{SS} \triangleq \mathbb{E}[S_i S_i^\top|\Theta] = W_i \hat{C}_i W_i^\top + \Gamma_i \tag{4.19}$$

M-step At the M-step, the parameters of the model Θ should be modified in order to decrease the loss function, using the sufficient statistics obtained in the E-step. The EM functional writes:

$$Q = \sum_{i=1}^{B} n_i [\mathrm{Tr}((R_i^{XX} - 2AR_i^{SX} + AR_i^{SS}A^\top)\Sigma_i^{-1}) + \mathrm{Tr}(R_i^{SS}P_i^{-1}) + \log|\Sigma_i| + \log|P_i|] \ , \tag{4.20}$$

which should be minimized with respect to the parameters Θ.

- Optimizing P_i: the source powers are decoupled from the other parameters in (4.20). Minimization of Φ w.r.t. P_i is easily obtained by canceling the gradient, yielding:

$$P_i = \mathrm{diag}(R_i^{SS}) \ .$$

- Optimizing Σ_i: the mixing matrix A and the noise covariance are entangled in eq. (4.20), rendering the analytic minimization of Φ impossible. Therefore, we first minimize Φ w.r.t Σ_i, keeping A constant. This yields:

$$\Sigma_i = \mathrm{diag}(R_i^{XX} - 2AR_i^{SX} + AR_i^{SS}A^\top) \ .$$

- Optimizing A: keeping the noise levels fixed, minimizing Φ w.r.t. A yields, by canceling the gradient: $\sum_{i=1}^{B} n_i \Sigma_i^{-1}(R_i^{XS} - AR_i^{SS}) = 0$. This can be seen as a system of equations for the rows of A which, thanks to the diagonality of Σ_i, is easily seen to decouple across the rows. For each row, simple algebra yields the close form solution:

$$\text{for } r = 1 \ldots p, \ \ A_{r:} = Q_{r:} M_r^{-1} \quad \text{with} \quad Q = \sum_{i=1}^{B} n_i \Sigma_i^{-1} R_i^{XS} \quad M_r = \sum_{i=1}^{B} n_i \sigma_{i,r}^{-2} R_i^{SS}.$$

  Therefore, the EM update of A only requires solving p linear systems of size $q \times q$.

Implementation details The EM algorithm iterates the E and M step until a certain convergence criterion is reached. In practice, iterations are stopped when the difference between two consecutive values of the log-likelihood is below a threshold: $\mathcal{L}^{t+1} > \mathcal{L}^t - \varepsilon$. In order to have a good initialization for the algorithm, we first fit the model with a fixed noise level for each bin: we estimate Σ subject to $\Sigma_i = \Sigma$ for all i. In this setting, the M-step is much simpler and computationally quicker. Then, the core SMICA algorithm with free noise starts with Σ_i all equal to the estimated noise level, and A and P_i start from the same initial value.

Conclusion and Perspectives

In this thesis, we have proposed novel ICA algorithms, with brain signal processing in mind. First, we acknowledged that fast state-of-the-art quasi-Newton algorithms for ICA all rely heavily on the independence assumption. On real data, this assumption is only true to some extent. To remedy this problem, we proposed the Picard algorithm, which shows fast convergence even on real data. Picard is shipped with two of the most popular M/EEG processing toolboxes.

In the case of very long signals, stochastic methods that do not need the whole dataset to perform one iteration are appealing. So far, only stochastic gradient descent was proposed for ICA. The major drawback of this method is that the learning rate is a critical and hard to tune parameter. To overcome this difficulty, we propose stochastic algorithms with descent guarantee for ICA, resorting on a majorization-minimization scheme. These algorithms do not have such critical hyper-parameter, and the descent guarantee makes them safer than SGD.

Finally, we use an ICA algorithm developed in the context of astronomy, SMICA, and apply it to M/EEG signals. Contrary to most ICA algorithms used for brain signal processing, this algorithm has a noise model, which enables fine source estimation through Wiener filtering and statistically sound dimensionality reduction via ICA. Results on brain signals show that it extracts interesting brain sources.

The ICA theory shows that finer source modeling leads to better separation. Although this assertion is debatable in an applied setting where the ICA model is wrong anyway, it would still be handy to go beyond Picard's binary density model, and proposed an adaptive density estimation. A simple way to do so is to alternate density estimation and unmixing matrix W estimation steps. However, it would likely render the memory of L-BFGS useless, since with respect to W, the cost function would change at each step. Another possibility is to perform density and unmixing matrix estimation at the same time. Fortunately, the Hessian for this extended set of parameters also simplifies at source separation: using an L-BFGS method should also lead to good results. However, it is hard to derive a family of densities that has enough diversity and leads to tractable computations.

The ideas behind Picard have been successfully applied to joint diagonalization, and can be transfered to any statistical model where the Hessian simplifies greatly when the model holds. An interesting problem is to identify a broader class of problems with such property. Of course, there always exists a parametrization of the parameter space that leads to simple Hessians, but a specificity of ICA is that these simplifications occur with the simple multiplicative updates.

The majorization-minimization employed to derive stochastic algorithms with descent guarantees only works when we have a super-Gaussian prior on the sources. Employing a similar method with sub-Gaussian densities would lead to a min-max problem, and strategies employed in this thesis become ineffective. It would be of interest to develop a more general framework for stochastic ICA that does not restrict the choice of density.

It would also be interesting to develop fast algorithms for SMICA. So far, we use an EM algorithm which tends to be slow. Second order methods could accelerate estimation, even though the optimization problem is more complicated than regular ICA due to its intricate parameters. Another important unresolved problem in ICA for neuroscience is to estimate the number of sources in the mixture. The most widely employed approach is to reduce dimension by PCA and keep a number of sources that account for 85% of variance. This is an ad-hoc procedure, while SMICA may offer a principled way of choosing the number of sources. Since we have a likelihood, statistical criterion like Akaike Information Criterion come to mind, but our experiments show that this does not yield a realistic number of sources.

Bibliography

Pierre Ablin, Jean-François Cardoso, and Alexandre Gramfort. Faster independent component analysis by preconditioning with Hessian approximations. *IEEE Transactions on Signal Processing*, 66(15):4040–4049, 2018a.

Pierre Ablin, Jean-François Cardoso, and Alexandre Gramfort. Accelerating likelihood optimization for ICA on real signals. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 151–160. Springer, 2018b.

Pierre Ablin, Jean-François Cardoso, and Alexandre Gramfort. Faster ICA under orthogonal constraint. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018c.

Pierre Ablin, Jean-François Cardoso, and Alexandre Gramfort. Beyond Pham's algorithm for joint diagonalization. In *ESANN*, 2019a.

Pierre Ablin, Dylan Fagot, Herwig Wendt, Alexandre Gramfort, and Cédric Févotte. A quasi-Newton algorithm on the orthogonal manifold for NMF with transform learning. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 700–704. IEEE, 2019b.

Pierre Ablin, Alexandre Gramfort, Jean-François Cardoso, and Francis Bach. Stochastic algorithms with descent guarantees for ICA. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1564–1573, 2019c.

Pierre Ablin, Thomas Moreau, Mathurin Massias, and Alexandre Gramfort. Learning step sizes for unfolded sparse coding. *arXiv preprint arXiv:1905.11071*, 2019d.

Alexandre Abraham, Fabian Pedregosa, Michael Eickenberg, Philippe Gervais, Andreas Mueller, Jean Kossaifi, Alexandre Gramfort, Bertrand Thirion, and Gaël Varoquaux. Machine learning for neuroimaging with scikit-learn. *Frontiers in neuroinformatics*, 8, 2014.

Pierre-Antoine Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

Michal Aharon, Michael Elad, and Alfred Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

Elena A Allen, Erik B Erhardt, Eswar Damaraju, William Gruner, Judith M Segall, Rogers F Silva, Martin Havlicek, Srinivas Rachakonda, Jill Fries, Ravi Kalyanam, et al. A baseline for the multivariate comparison of resting-state networks. *Frontiers in systems neuroscience*, 5, 2011.

Elena A Allen, Eswar Damaraju, Sergey M Plis, Erik B Erhardt, Tom Eichele, and Vince D Calhoun. Tracking whole-brain connectivity dynamics in the resting state. *Cerebral cortex*, 24(3):663–676, 2014.

Shun-ichi Amari, Andrzej Cichocki, and Howard Hua Yang. A new learning algorithm for blind signal separation. In *Advances in neural information processing systems*, pages 757–763, 1996.

Shun-Ichi Amari, Tian-Ping Chen, and Andrzej Cichocki. Stability analysis of learning algorithms for blind source separation. *Neural Networks*, 10(8):1345–1351, 1997.

Jorn Anemuller, Terrence J. Sejnowski, and Scott Makeig. Complex independent component analysis of frequency-domain electroencephalographic data. *Neural Networks*, 16(9):1311 – 1323, 2003. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2003.08.003. URL http://www.sciencedirect.com/science/article/pii/S0893608003002442. Neuroinformatics.

Fiorenzo Artoni, Arnaud Delorme, and Scott Makeig. Applying dimension reduction to EEG data by principal component analysis reduces the quality of its subsequent independent component decomposition. *NeuroImage*, 175:176–187, 2018.

Michal Assaf, Kanchana Jagannathan, Vince D Calhoun, Laura Miller, Michael C Stevens, Robert Sahl, Jacqueline G O'boyle, Robert T Schultz, and Godfrey D Pearlson. Abnormal functional connectivity of default mode sub-networks in autism spectrum disorder patients. *Neuroimage*, 53(1):247–256, 2010.

Francis Bach and Michael Jordan. Kernel independent component analysis. *Journal of machine learning research*, 3(Jul):1–48, 2002.

Hanna Becker, Pierre Comon, Laurent Albera, Martin Haardt, and Isabelle Merlet. Multi-way space–time–wave-vector analysis for eeg source separation. *Signal Processing*, 92(4):1021–1031, 2012.

Hanna Becker, Laurent Albera, Pierre Comon, Rémi Gribonval, Fabrice Wendling, and Isabelle Merlet. Localization of distributed eeg sources in the context of epilepsy: a simulation study. *IRBM*, 37(5-6):242–253, 2016.

Christian F Beckmann, Marilena DeLuca, Joseph T Devlin, and Stephen M Smith. Investigations into resting-state connectivity using independent component analysis. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1457):1001–1013, 2005. doi: 10.1098/rstb.2005.1634. URL https://royalsocietypublishing.org/doi/abs/10.1098/rstb.2005.1634.

Anthony J Bell and Terrence J Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159, 1995.

Adel Belouchrani, Karim Abed-Meraim, Jean-François Cardoso, and E. Moulines. A blind source separation technique using second-order statistics. *IEEE Transactions on Signal Processing*, 45(2):434–444, 1997.

Olivier Bermond and Jean-François Cardoso. Approximate likelihood for noisy mixtures. In *Proc. ICA*, volume 99, pages 325–330, 1999.

Wolfgang Bertram. Differential Geometry, Lie Groups and Symmetric Spaces over General Base Fields and Rings. *Memoirs of the American Mathematical Society*, 2008.

Leon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*, 2016.

Matthew J Brookes, Mark Woolrich, Henry Luckhoo, Darren Price, Joanne R Hale, Mary C Stephenson, Gareth R Barnes, Stephen M Smith, and Peter G Morris. Investigating the electrophysiological basis of resting state networks using magnetoencephalography. *Proceedings of the National Academy of Sciences*, 108(40):16783–16788, 2011.

Robert Grover Brown, Patrick YC Hwang, et al. *Introduction to random signals and applied Kalman filtering*, volume 3. Wiley New York, 1992.

Charles George Broyden. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970.

György Buzsáki. *Rhythms of the Brain*. Oxford University Press, 2006.

György Buzsáki and Andreas Draguhn. Neuronal oscillations in cortical networks. *Science*, 304(5679):1926–1929, 2004. ISSN 0036-8075. doi: 10.1126/science.1099745. URL https://science.sciencemag.org/content/304/5679/1926.

György Buzsáki and Rodolfo Llinás. Space and time in the brain. *Science*, 358 (6362):482–485, 2017. ISSN 0036-8075. doi: 10.1126/science.aan8869. URL https://science.sciencemag.org/content/358/6362/482.

Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

AC Cadavid, JK Lawrence, and A Ruzmaikin. Principal components and independent component analysis of solar and space data. *Solar Physics*, 248(2):247–261, 2008.

Olivier Cappé and Eric Moulines. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613, 2009.

Jean-François Cardoso. On the stability of some source separation algorithms. In *Proc. of the 1998 IEEE SP workshop on neural networks for signal processing (NNSP '98)*, pages 13–22, 1998a.

Jean-François Cardoso. On the performance of orthogonal source separation algorithms. In *Proceedings of EUSIPCO*, volume 94, pages 776–779, 1994.

Jean-François Cardoso. Infomax and maximum likelihood for blind source separation. *IEEE Signal processing letters*, 4(4):112–114, 1997.

Jean-François Cardoso. Blind signal separation: statistical principles. *Proceedings of the IEEE*, 86(10):2009–2025, 1998b.

Jean-François Cardoso. The three easy routes to independent component analysis; contrasts and geometry. In *Proc. ICA*, volume 2001, pages 1–6, 2001.

Jean-François Cardoso and Beate H Laheld. Equivariant adaptive source separation. *IEEE Transactions on Signal Processing*, 44(12):3017–3030, 1996.

Jean-François Cardoso and Antoine Souloumiac. Blind beamforming for non-gaussian signals. *IEE Proceedings F - Radar and Signal Processing*, 140(6):362–370, Dec 1993. ISSN 0956-375X. doi: 10.1049/ip-f-2.1993.0054.

Jean-François Cardoso and Antoine Souloumiac. Jacobi angles for simultaneous diagonalization. *SIAM journal on matrix analysis and applications*, 17(1):161–164, 1996.

Jean-François Cardoso, Hichem Snoussi, and Jacques Delabrouille. Blind separation of noisy Gaussian stationary sources. application to cosmic microwave background imaging. In *2002 11th European Signal Processing Conference*, pages 1–4. IEEE, 2002.

Pascal Chevalier, Laurent Albera, Pierre Comon, and Anne Ferréol. Comparative performance analysis of eight blind source separation methods on radiocommunications signals. In *Proc. of IEEE International Joint Conference on Neural Networks*, volume 1, pages 273–278, 2004.

Heeyoul Choi and Seungjin Choi. A relative trust-region algorithm for independent component analysis. *Neurocomputing*, 70(7):1502–1510, 2007.

Pierre Comon. Independent component analysis, a new concept? *Signal processing*, 36 (3):287–314, 1994.

Pierre Comon and Christian Jutten. *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic press, 2010.

Marco Congedo, Cédric Gouy-Pailler, and Christian Jutten. On the blind source separation of human electroencephalogram by approximate joint diagonalization of second order statistics. *Clinical Neurophysiology*, 119(12):2677–2686, 2008.

ADHD-200 Consortium et al. The ADHD-200 consortium: a model to advance the translational potential of neuroimaging in clinical neuroscience. *Frontiers in systems neuroscience*, 6, 2012.

Jurgen. Dammers, Michael Schiek, Frank Boers, Carmen Silex, Mikhail Zvyagintsev, Uwe Pietrzyk, and Klaus Mathiak. Integration of amplitude and phase statistics for complete artifact removal in independent components of neuromagnetic recordings. *IEEE Transactions on Biomedical Engineering*, 55(10):2353–2362, Oct 2008. ISSN 0018-9294. doi: 10.1109/TBME.2008.926677.

George Darmois. Analyse générale des liaisons stochastiques: etude particulière de l'analyse factorielle linéaire. *Revue de l'Institut international de statistique*, pages 2–8, 1953.

William C Davidon. Variable metric method for minimization. *SIAM Journal on Optimization*, 1(1):1–17, 1991.

Mike Davies. Identifiability issues in noisy ica. *IEEE Signal processing letters*, 11(5): 470–473, 2004.

Maarten De Vos, Anneleen Vergult, Lieven De Lathauwer, Wim De Clercq, Sabine Van Huffel, Patrick Dupont, Andre Palmini, and Wim Van Paesschen. Canonical decomposition of ictal scalp eeg reliably detects the seizure onset zone. *NeuroImage*, 37(3):844–854, 2007.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Proc. NIPS*, pages 1646–1654, 2014.

Nima Dehghani, Claude Bédard, Sydney S. Cash, Eric Halgren, and Alain Destexhe. Comparative power spectral analysis of simultaneous elecroencephalographic and magnetoencephalographic recordings in humans suggests non-resistive extracellular media. *Journal of Computational Neuroscience*, 29(3):405–421, Dec 2010. ISSN 1573-6873. doi: 10.1007/s10827-010-0263-2. URL https://doi.org/10.1007/s10827-010-0263-2.

Jacques Delabrouille, Jean-François Cardoso, and Guillaume Patanchon. Multidetector multicomponent spectral matching and applications for cosmic microwave background data analysis. *Monthly Notices of the Royal Astronomical Society*, 346(4):1089–1102, 2003.

Arnaud Delorme and Scott Makeig. EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *Journal of neuroscience methods*, 134(1):9–21, 2004.

Arnaud Delorme, Jason Palmer, Julie Onton, Robert Oostenveld, and Scott Makeig. Independent EEG sources are dipolar. *PloS one*, 7(2):e30135, 2012.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (methodological)*, pages 1–38, 1977.

Mads Dyrholm, Scott Makeig, and Lars Kai Hansen. Model selection for convolutive ICA with an application to spatiotemporal analysis of EEG. *Neural Computation*, 19 (4):934–955, 2007. doi: 10.1162/neco.2007.19.4.934.

Roger Fletcher. A new approach to variable metric algorithms. *The computer journal*, 13(3):317–322, 1970.

Roger Fletcher and Michael JD Powell. A rapidly convergent descent method for minimization. *The computer journal*, 6(2):163–168, 1963.

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

Donald Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26, 1970.

Germán Gómez-Herrero, Mercedes Atienza, Karen Egiazarian, and Jose L Cantero. Measuring directional coupling between EEG sources. *Neuroimage*, 43(3):497–508, 2008.

Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A. Engemann, Daniel Strohmeier, Christian Brodbeck, Lauri Parkkonen, and Matti S. Hämäläinen. MNE software for processing MEG and EEG data. *NeuroImage*, 86:446 – 460, 2014. ISSN 1053-8119.

Matti Hämäläinen, Riitta Hari, Risto J Ilmoniemi, Jukka Knuutila, and Olli V Lounasmaa. Magnetoencephalography-theory, instrumentation, and applications to noninvasive studies of the working human brain. *Reviews of modern Physics*, 65(2):413, 1993.

Riitta Hari and Riitta Salmelin. Human cortical oscillations: a neuromagnetic view through the skull. *Trends in Neurosciences*, 20(1):44 – 49, 1997. ISSN 0166-2236. doi: https://doi.org/10.1016/S0166-2236(96)10065-5. URL http://www.sciencedirect.com/science/article/pii/S0166223696100655.

Jean Honorio, Dimitris Samaras, Irina Rish, and Guillermo Cecchi. Variable selection for gaussian graphical models. In *Artificial Intelligence and Statistics*, pages 538–546, 2012.

Patrick O. Hoyer and Aapo Hyvärinen. Independent component analysis applied to feature extraction from colour and stereo images. *Network: Computation in Neural Systems*, 11(3):191–210, 2000.

Scott A Huettel, Allen W Song, Gregory McCarthy, et al. *Functional magnetic resonance imaging*, volume 1. Sinauer Associates Sunderland, MA, 2004.

Aapo Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, 1999.

Aapo Hyvärinen. The fixed-point algorithm and maximum likelihood estimation for independent component analysis. *Neural Processing Letters*, 10(1):1–5, 1999.

Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4):411–430, 2000.

Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent component analysis*, volume 46. John Wiley & Sons, 2004.

Aapo Hyvärinen, Pavan Ramkumar, Lauri Parkkonen, and Riitta Hari. Independent component analysis of short-time Fourier transforms for spontaneous EEG/MEG analysis. *NeuroImage*, 49(1):257–271, 2010.

Shiro Ikeda and Keisuke Toyama. Independent component analysis for noisy data-MEG data analysis. *Neural Networks*, 13(10):1063–1074, 2000.

Lianjun Jiang, Richard H Byrd, Elizabeth Eskow, and Robert B Schnabel. A preconditioned L-BFGS algorithm with application to molecular energy minimization. Technical report, Colorado Univ. at Boulder Dept. of Computer Science, 2004.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Proc. NIPS*, pages 315–323, 2013.

Tzyy-Ping Jung, Colin Humphries, Te-Won Lee, Scott Makeig, Martin J. McKeown, Vicente Iragui, and Terrence J. Sejnowski. Extended ICA removes artifacts from electroencephalographic recordings. In *Proceedings of the 10th International Conference on Neural Information Processing Systems*, NIPS'97, pages 894–900, Cambridge, MA, USA, 1997. MIT Press.

Tzyy-Ping Jung, Scott Makeig, Colin Humphries, Te-Won Lee, Martin J Mckeown, Vicente Iragui, and Terrence J Sejnowski. Removing electroencephalographic artifacts by blind source separation. *Psychophysiology*, 37(2):163–178, 2000.

Christian Jutten and Jeanny Herault. Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal processing*, 24(1):1–10, 1991.

Joseph B Kruskal. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear algebra and its applications*, 18(2):95–138, 1977.

Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

Su-In Lee and Serafim Batzoglou. Application of independent component analysis to microarrays. *Genome biology*, 4(11):R76, 2003.

Te-Won Lee, Mark Girolami, and Terrence J Sejnowski. Independent component analysis using an extended infomax algorithm for mixed subgaussian and supergaussian sources. *Neural computation*, 11(2):417–441, 1999.

Augustin Lefevre, Francis Bach, and Cédric Févotte. Online algorithms for nonnegative matrix factorization with the Itakura-Saito divergence. In *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2011 IEEE Workshop on*, pages 313–316. IEEE, 2011.

Michael S Lewicki and Terrence J Sejnowski. Learning overcomplete representations. *Neural computation*, 12(2):337–365, 2000.

Dong-Hui Li and Masao Fukushima. A modified bfgs method and its global convergence in nonconvex minimization. *Journal of Computational and Applied Mathematics*, 129 (1-2):15–35, 2001.

Lek-Heng Lim and Pierre Comon. Blind multilinear identification. *IEEE Transactions on Information Theory*, 60(2):1260–1280, 2013.

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

Davide Maino, A Farusi, Carlo Baccigalupi, Francesca Perrotta, AJ Banday, L Bedini, Carlo Burigana, Gianfranco De Zotti, KM Górski, and E Salerno. All-sky astrophysical component separation with fast independent component analysis (FASTICA). *Monthly Notices of the Royal Astronomical Society*, 334(1):53–68, 2002.

Julien Mairal. Optimization with first-order surrogate functions. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 783–791, 2013.

Julien Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.

Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proc. ICML*, pages 689–696. ACM, 2009.

Scott Makeig, Tzyy-Ping Jung, Anthony J. Bell, Dara Ghahremani, and Terrence J. Sejnowski. Blind separation of auditory event-related brain responses into independent components. *Proceedings of the National Academy of Sciences (PNAS)*, 94(20): 10979–10984, 1997. doi: 10.1073/pnas.94.20.10979.

Scott Makeig, Stefan Debener, Julie Onton, and Arnaud Delorme. Mining event-related brain dynamics. *Trends in cognitive sciences*, 8(5):204–210, 2004.

Dante Mantini, Raffaella Franciotti, Gian Luca Romani, and Vittorio Pizzella. Improving MEG source localizations: an automated method for complete artifact removal based on independent component analysis. *NeuroImage*, 40(1):160–173, 2008.

Kiyotoshi Matsuoka, Masahiro Ohoya, and Mitsuru Kawamoto. A neural net for blind separation of nonstationary signals. *Neural networks*, 8(3):411–419, 1995.

Martin J McKeown, Scott Makeig, Greg G Brown, Tzyy-Ping Jung, Sandra S Kindermann, Anthony J Bell, and Terrence J Sejnowski. Analysis of fmri data by blind separation into independent spatial components. Technical report, aval Health Research Center San Diego, 1997.

Fumikazu Miwakeichi, Eduardo Martınez-Montes, Pedro A Valdés-Sosa, Nobuaki Nishiyama, Hiroaki Mizuhara, and Yoko Yamaguchi. Decomposing eeg data into space–time–frequency components using parallel factor analysis. *NeuroImage*, 22(3): 1035–1045, 2004.

Jair Montoya-Martínez, Jean-François Cardoso, and Alexandre Gramfort. Caveats with stochastic gradient and maximum likelihood based ICA for EEG. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 279–289. Springer, 2017.

Jorge J Moré and David J Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Transactions on Mathematical Software (TOMS)*, 20(3):286–307, 1994.

Eric Moreau and Odile Macchi. Self-adaptive source separation. ii. comparison of the direct, feedback, and mixed linear network. *IEEE Trans. on Signal Processing*, 46(1): 39–50, 1998.

Morten Mørup, Lars Kai Hansen, Christoph S Herrmann, Josef Parnas, and Sidse M Arnfred. Parallel factor analysis as an exploratory tool for wavelet transformed event-related eeg. *NeuroImage*, 29(3):938–947, 2006.

Eric Moulines, Jean-François Cardoso, and Elisabeth Gassiat. Maximum likelihood for blind separation and deconvolution of noisy signals using mixture models. In *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 3617–3620. IEEE, 1997.

Radford M. Neal and Geoffrey E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

Arkadii Semenovich Nemirovsky and David Borisovich Yudin. *Problem complexity and method efficiency in optimization.* J. Wiley & Sons, 1983.

Ernst Niedermeyer and F. H. Lopes da Silva. *Electroencephalography : basic principles, clinical applications, and related fields.* Philadelphia ; London : Lippincott Williams & Wilkins, 5th ed edition, 2005. ISBN 0781751268.

Vadim V Nikulin, Guido Nolte, and Gabriel Curio. A novel method for reliable and fast extraction of neuronal eeg/meg oscillations on the basis of spatio-spectral decomposition. *NeuroImage*, 55(4):1528–1535, 2011.

Jorge Nocedal and Stephen J Wright. *Numerical Optimization.* Springer, 1999.

Danielle Nuzillard and Albert Bijaoui. Blind source separation and analysis of multispectral astronomical images. *Astronomy and Astrophysics Supplement Series*, 147(1): 129–138, 2000.

Erkki Oja and Zhijian Yuan. The fastica algorithm revisited: Convergence analysis. *IEEE Transactions on Neural Networks*, 17(6):1370–1381, 2006.

Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3): 145–175, 2001.

Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607, 1996.

Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5 (2):111–126, 1994.

Jason Palmer, Ken Kreutz-Delgado, Bhaskar D. Rao, and David P. Wipf. Variational EM algorithms for non-gaussian latent variable models. In *Proc. NIPS*, pages 1059–1066, 2006.

Jason A. Palmer, Scott Makeig, Ken Kreutz-Delgado, and Bhaskar D. Rao. Newton method for the ICA mixture model. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1805–1808, March 2008. doi: 10.1109/ICASSP.2008.4517982.

Jason A Palmer, Ken Kreutz-Delgado, and Scott Makeig. AMICA: An adaptive mixture of independent component analyzers with shared components. Technical report, Swartz Center for Computatonal Neursoscience, University of California San Diego, Tech. Rep, 2012.

Dinh-Tuan Pham. Blind separation of instantaneous mixture of sources via the Gaussian mutual information criterion. *Signal Processing*, 81(4):855–870, 2001a.

Dinh-Tuan Pham. Joint approximate diagonalization of positive definite hermitian matrices. *SIAM Journal on Matrix Analysis and Applications*, 22(4):1136–1152, 2001b.

Dinh-Tuan Pham and Jean-François Cardoso. Blind separation of instantaneous mixtures of nonstationary sources. *IEEE Transactions on Signal Processing*, 49(9):1837–1848, 2001.

Dinh-Tuan Pham and Jean-Francois Cardoso. Source adaptive blind source separation: Gaussian models and sparsity. In *Wavelets: Applications in Signal and Image Processing, X, Proc. of SPIE*, volume 5207, San Diego, January 2003.

Dinh-Tuan Pham and Philippe Garat. Blind separation of mixture of independent sources through a quasi-maximum likelihood approach. *IEEE Transactions on Signal Processing*, 45(7):1712–1725, 1997.

Fabien Poncelet, Gaëtan Kerschen, J-C Golinval, and Damien Verhelst. Output-only modal analysis using blind source separation techniques. *Mechanical systems and signal processing*, 21(6):2335–2358, 2007.

Francesca Raimondi and Pierre Comon. Tensor decomposition of polarized seismic waves. In *GRETSI'2015*, 2015.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

Daria La Rocca, Nicolas Zilber, Patrice Abry, Virginie van Wassenhove, and Philippe Ciuciu. Self-similarity and multifractality in human brain activity: A wavelet-based analysis of scale-free brain dynamics. *Journal of Neuroscience Methods*, 309:175 – 187, 2018. ISSN 0165-0270. doi: https://doi.org/10.1016/j.jneumeth.2018.09.010. URL http://www.sciencedirect.com/science/article/pii/S0165027018302784.

Douglas N Rutledge and D Jouan-Rimbaud Bouveresse. Independent components analysis with the JADE algorithm. *TrAC Trends in Analytical Chemistry*, 50:22–32, 2013.

David Sabbagh, Pierre Ablin, Gaël Varoquaux, Alexandre Gramfort, and Denis A Engeman. Manifold-regression to predict from meg/eeg brain signals without source modeling. *arXiv preprint arXiv:1906.02687*, 2019.

Michael Scherg and Detlev Von Cramon. Two bilateral sources of the late AEP as identified by a spatio-temporal dipole model. *Electroencephalogr. Clin. Neurophysiol.*, 62(1):32–44, Jan. 1985.

Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

Matthias Scholz, S Gatzek, A Sterling, Oliver Fiehn, and Joachim Selbig. Metabolite fingerprinting: detecting biological features by independent component analysis. *Bioinformatics*, 20(15):2447–2454, 2004.

David F Shanno. Conditioning of quasi-Newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970.

Hao Shen, Martin Kleinsteuber, and Knut Huper. Local convergence analysis of FastICA and related algorithms. *IEEE Transactions on Neural Networks*, 19(6):1022–1032, 2008.

Jonathan Richard Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.

Nicholas D Sidiropoulos, Rasmus Bro, and Georgios B Giannakis. Parallel factor analysis in sensor array processing. *IEEE transactions on Signal Processing*, 48(8):2377–2388, 2000.

Julia M Stephen, Brian A Coffman, Rex E Jung, Juan R Bustillo, CJ Aine, and Vincent D Calhoun. Using joint ICA to link function and structure using MEG and DTI in schizophrenia. *Neuroimage*, 83:418–430, 2013.

Abdulhamit Subasi and M Ismail Gursoy. EEG signal classification using PCA, ICA, LDA and support vector machines. *Expert systems with applications*, 37(12):8659–8666, 2010.

Samu Taulu. Spatiotemporal Signal Space Separation method for rejecting nearby interference in MEG measurements. *Physics in Medicine and Biology*, 51(7):1759–1769, 2006.

Petr Tichavsky, Zbynek Koldovsky, and Erkki Oja. Performance analysis of the FastICA algorithm and cramer-rao bounds for linear independent component analysis. *IEEE transactions on Signal Processing*, 54(4):1189–1203, 2006.

Philippe Tillet, HT Kung, and David Cox. Infomax-ICA using Hessian-free optimization. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 2537–2541. IEEE, 2017.

Jose Antonio Urigüen and Begoña Garcia-Zapirain. EEG artifact removal - state-of-the-art and guidelines. *Journal of neural engineering*, 12(3):031001, 2015.

Mikko.A. Uusitalo and Risto.J. Ilmoniemi. Signal-space projection method for separating MEG or EEG into components. *Medical and Biological Engineering and Computing*, 35(2):135–140, 1997.

Stefan Van der Walt, S. Chri Colbert, and Gaël Varoquaux. The NumPy array: a structure for efficient numerical computation. *Comp. in Sci. & Eng.*, 13(2):22–30, 2011.

Gaël Varoquaux, Sepideh Sadaghiani, Philippe Pinel, Andreas Kleinschmidt, Jean-Baptiste Poline, and Bertrand Thirion. A group model for stable multi-subject ICA on fMRI datasets. *Neuroimage*, 51(1):288–299, 2010.

Ricardo Vigário, Jaakko Sarela, V Jousmiki, Matti Hamalainen, and Erkki Oja. Independent component approach to the analysis of EEG and MEG recordings. *IEEE Transactions on Biomedical Engineering*, 47(5):589–593, 2000.

Valeriu Vrabie, Cyril Gobinet, Olivier Piot, Ali Tfayli, Philippe Bernard, Régis Huez, and Michel Manfait. Independent component analysis of raman spectra: Application on paraffin-embedded skin biopsies. *Biomedical Signal Processing and Control*, 2(1): 40 – 50, 2007. ISSN 1746-8094. doi: https://doi.org/10.1016/j.bspc.2007.03.001.

Tianwen Wei. A convergence and asymptotic analysis of the generalized symmetric FastICA algorithm. *IEEE transactions on signal processing*, 63(24):6445–6458, 2015.

Martin Weis, Florian Romer, Martin Haardt, Dunja Jannek, and Peter Husar. Multi-dimensional space-time-frequency component analysis of event related eeg data using closed-form parafac. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 349–352. IEEE, 2009.

Jan R Wessel and Adam R Aron. It's not too late: The onset of the frontocentral p3 indexes successful response inhibition in the stop-signal paradigm. *Psychophysiology*, 52(4):472–480, 2015.

Philip Wolfe. Convergence conditions for ascent methods. *SIAM review*, 11(2):226–235, 1969.

Vicente Zarzoso and Pierre Comon. Robust independent component analysis by iterative maximization of the kurtosis contrast with algebraic optimal step size. *IEEE Transactions on Neural Networks*, 21(2):248–261, 2009.

Vicente Zarzoso, Pierre Comon, and Mariem Kallel. How fast is fastica? In *2006 14th European Signal Processing Conference*, pages 1–5. IEEE, 2006.

Michael Zibulevsky. Blind source separation with relative Newton method. In *Proc. ICA*, volume 2003, pages 897–902, 2003.

Andreas Ziehe, Pavel Laskov, Guido Nolte, and Klaus-Robert Muller. A fast algorithm for joint diagonalization with non-orthogonal transformations and its application to blind source separation. *Journal of Machine Learning Research*, 5(Jul):777–800, 2004.

Titre : Exploration de signaux M/EEG multivariés à l'aide de modèles non-stationnaires

Mots Clefs : Analyse en composantes indépendantes, optimisation non-convexe, données cérébrales

Résumé : L'Analyse en Composantes Indépendantes (ACI) modèle un ensemble de signaux comme une combinaison linéaire de sources indépendantes. Cette méthode joue un rôle clé dans le traitement des signaux de magnétoencéphalographie (MEG) et électroencéphalographie (EEG). L'ACI de tels signaux permet d'isoler des sources de cerveau intéressantes, de les localiser, et de les séparer d'artefacts. L'ACI fait partie de la boite à outils de nombreux neuroscientifiques, et est utilisée dans de nombreux articles de recherche en neurosciences. Cependant, les algorithmes d'ACI les plus utilisés ont été développés dans les années 90. Ils sont souvent lents lorsqu'ils sont appliqués sur des données réelles, et sont limités au modèle d'ACI classique. L'objectif de cette thèse est de développer des algorithmes d'ACI utiles en pratique aux neuroscientifiques. Nous suivons deux axes. Le premier est celui de la vitesse : nous considérons le problème d'optimisation résolu par deux des algorithmes les plus utilisés par les praticiens : Infomax et FastICA. Nous développons une nouvelle technique se basant sur un préconditionnement par des approximations de la Hessienne de l'algorithm L-BFGS. L'algorithme qui en résulte, Picard, est conçu pour être appliqué sur données réelles, où l'hypothèse d'indépendance n'est jamais entièrement vraie. Sur des données de M/EEG, il converge plus vite que les implémentations 'historiques'.

Les méthodes incrémentales, qui traitent quelques échantillons à la fois au lieu du jeu de données complet, constituent une autre possibilité d'accélération de l'ACI. Ces méthodes connaissent une popularité grandissante grace à leur faculté à bien passer à l'échelle sur de grands jeux de données. Nous proposons un algorithme incrémental pour l'ACI, qui possède une importante propriété de descente garantie. En conséquence, cet algorithme est simple d'utilisation, et n'a pas de paramètre critique et difficile à régler comme un taux d'apprentissage.

En suivant un second axe, nous proposons de prendre en compte du bruit dans le modèle d'ACI. Le modèle resultant est notoirement difficile et long à estimer sous l'hypothèse standard de non-Gaussianité de l'ACI. Nous nous reposons donc sur une hypothèse de diversité spectrale, qui mène à un algorithme facile d'utilisation et utilisable en pratique, SMICA. La modélisation du bruit permet de nouvelles possibilités inenvisageables avec un modèle d'ACI classique, comme une estimation fine des source et l'utilisation de l'ACI comme une technique de réduction de dimension statistiquement bien posée. De nombreuses expériences sur données M/EEG démontrent l'utilité de cette nouvelle approche.

Tous les algorithmes développés dans cette thèse sont disponibles en accès libre sur internet. L'algorithme Picard est inclus dans les librairies de traitement de données M/EEG les plus populaires en Python (MNE) et en Matlab (EEGlab).

Title : Exploration of multivariate M/EEG signals using non-stationary models

Keys words : Independent component analysis, non-convex optimization, brain data processing

Abstract : Independent Component Analysis (ICA) models a set of signals as linear combinations of independent sources. This analysis method plays a key role in electroencephalography (EEG) and magnetoencephalography (MEG) signal processing. Applied on such signals, it allows to isolate interesting brain sources, locate them, and separate them from artifacts. ICA belongs to the toolbox of many neuroscientists, and is a part of the processing pipeline of many research articles. Yet, the most widely used algorithms date back to the 90's. They are often quite slow, and stick to the standard ICA model, without more advanced features.

The goal of this thesis is to develop practical ICA algorithms to help neuroscientists. We follow two axes. The first one is that of speed. We consider the optimization problems solved by two of the most widely used ICA algorithms by practitioners: Infomax and FastICA. We develop a novel technique based on preconditioning the L-BFGS algorithm with Hessian approximation. The resulting algorithm, Picard, is tailored for real data applications, where the independence assumption is never entirely true. On M/EEG data, it converges faster than the 'historical' implementations.

Another possibility to accelerate ICA is to use incremental methods, which process a few samples at a time instead of the whole dataset. Such methods have gained huge interest in the last years due to their ability to scale well to very large datasets. We propose an incremental algorithm for ICA, with important descent guarantees. As a consequence, the proposed algorithm is simple to use and does not have a critical and hard to tune parameter like a learning rate.

In a second axis, we propose to incorporate noise in the ICA model. Such a model is notoriously hard to fit under the standard non-Gaussian hypothesis of ICA, and would render estimation extremely long. Instead, we rely on a spectral diversity assumption, which leads to a practical algorithm, SMICA. The noise model opens the door to new possibilities, like finer estimation of the sources, and use of ICA as a statistically sound dimension reduction technique. Thorough experiments on M/EEG datasets demonstrate the usefulness of this approach.

All algorithms developed in this thesis are open-sourced and available online. The Picard algorithm is included in the largest M/EEG processing Python library, MNE and Matlab library, EEGlab.