Faster ICA by preconditioning with Hessian approximations

Pierre Ablin, Jean-Francois Cardoso, Alexandre Gramfort

Faster ICA by preconditioning with Hessian approximations

Pierre Ablin [*], Jean-Francois Cardoso [†], and Alexandre Gramfort [‡]

July 2, 2017

Abstract

Independent Component Analysis (ICA) is a powerful technique for unsupervised data exploration that is widely used across fields such as neuroscience, astronomy, chemistry or biology. Linear ICA is a linear latent factor model, similar to sparse dictionary learning, that aims at discovering statistically independent sources from multivariate observations. ICA is a probabilistic generative model for which inference is classically done by maximum likelihood estimation. Estimating sources by maximum likelihood leads to a smooth non-convex optimization problem where the unknown is a matrix called the separating or unmixing matrix. As the gradient of the likelihood is available in closed form, first order gradient methods, stochastic or non-stochastic, are often employed despite a slow convergence such as in the Infomax algorithm. While the Hessian is known analytically, the cost of its computation and inversion makes Newton method unpractical for a large number of sources. We show how sparse and positive approximations of the true Hessian can be obtained and used to precondition the L-BFGS algorithm. Results on simulations and two applied problems (EEG data and image patches) demonstrate that the proposed technique leads to convergence that can be orders of magnitude faster than algorithms commonly used today even when looking for hundred of sources.

Keywords : Independent Component Analysis, Blind source separation, quasi-Newton methods, maximum likelihood estimation, second order methods, preconditioning.

[*]P. Ablin works at Inria, Parietal Team, Université Paris-Saclay, Saclay, France; e-mail: pierre.ablin@inria.fr

[†]J. F. Cardoso is with the Institut d'Astrophysique de Paris, CNRS, Paris, France; e-mail: cardoso@iap.fr

[‡]A. Gramfort is with Inria, Parietal Team, Université Paris-Saclay, Saclay, France; e-mail: alexandre.gramfort@inria.fr

1 Introduction

Independent Component Analysis (ICA) [1, 2] is a multivariate data exploration tool massively used across scientific disciplines such as neuroscience [3, 4, 5, 6], astronomy [7, 8, 9], chemistry [10, 11] or biology [12, 13]. The underlying assumption of ICA is that the data are obtained by combining latent components which are statistically independent. The linear ICA problem addresses the case where latent variables and observations are linked by a linear transform. In the linear case, ICA boils down to the estimation of a linear transform of the input signals to obtain statistically independent output signals, which are called *sources*. The matrix that transforms the sources into the measured signals is called the *mixing matrix*, and its inverse, which turns the input signals into sources, is called the *unmixing matrix*.

The strength and wide applicability of ICA come from its limited number of assumptions. For ICA to become a well-posed problem it is only required that all sources except one are non-Gaussian and statistically independent [1]. The generality of this concept explains the usefulness of ICA in many domains. In neuroscience, ICA is typically used to find artifacts in signals, because artifacts are by nature independent from the brain signal and also tend to be strong and transient with non-Gaussian distributions [14]. In image processing, ICA has also been shown to extract meaningful features, that can be used for instance for denoising using image patches [15]. This last application reveals the similarity between a popular patch-based method known as sparse dictionary learning [16] and ICA that can be seen as a probabilistic formulation of this learning problem [17].

Pham [18] explained that inferring the unmixing matrix in linear ICA can be formulated as a likelihood maximization problem. Interestingly, it has been shown respectively by Hyvarinen [19] and Cardoso [20] that two of the most used algorithms, FastICA [21] and Infomax [22], can be derived from this principle. In particular, the learning rule of Infomax boils down to a stochastic gradient method for maximization of the log-likelihood of the model. Maximum likelihood is therefore a central concept for ICA, which later motivated new works such as the AMICA algorithm [23].

The Hessian of the likelihood function has been thoroughly studied in [24, 25], and it was proposed in [18, 26] to use this second order information in iterative solvers. In [27], Palmer *et al.* reused these ideas and reported important speed ups compared to first order gradient methods used in the literature. To the best of our knowledge, this technique constitutes the state-of-the-art for optimizing the likelihood function. While theory and experiments show that convergence of this second-order algorithm is quadratic on simulations when the ICA model holds, on real signals which are not exactly a mixture of independent sources the rate of convergence falls back to linear. This article addresses this issue by building on the classical optimization algorithm L-BFGS.

The practical implications of this work are important as computational costs of ICA estimation can be a bottleneck in data exploration. In the neuroscience community, Infomax is the default ICA algorithm used in the popular EEGLab

toolbox [28]. By nature, this algorithm requires careful tuning of the learning rate parameter and fails to perfectly reach a stationary point of the likelihood [29]. The algorithm proposed here does not suffer from these limitations while exhibiting significantly faster convergence.

This article is organized as follows: In section 2, we show how the maximum likelihood principle leads to a matrix-valued unconstrained and smooth optimization problem. We then detail how to find two good and low cost approximations of the Hessian of the likelihood function, and propose an efficient algorithm based on L-BFGS that can exploit these approximations to improve the local conditioning of the optimization problem. Finally, in section 3, we show on synthetic signals, on multiple electroencephalography (EEG) datasets and on images that these algorithms can greatly accelerate the convergence of maximum-likelihood based ICA algorithms, and that it can improve the reliability of ICA decompositions.

Notation

We denote by $\mathcal{E} \in \mathbb{R}^{N \times N}$ a matrix with small norm. Let M be a square $N \times N$ matrix, and B a fourth order tensor of size $N \times N \times N \times N$. We can apply B to M: it gives another matrix denoted as BM whose entries are $(BM)_{ij} = \sum_{k,l} B_{ijkl} M_{kl}$. The Frobenius matrix scalar product is denoted as $\langle M | M' \rangle = \mathrm{Tr}(M^\top M') = \sum_{i,j} M_{ij} M'_{ij}$, and $\|M\| = \sqrt{\langle M | M \rangle}$ is the associated Frobenius matrix norm. We also denote $\langle M' | B | M \rangle = \langle M' | BM \rangle = \sum_{i,j,k,l} B_{ijkl} M'_{ij} M_{kl}$. The scalars α or α_k are step sizes in the optimization algorithms. We denote by $\det(M)$ the determinant of M and by I the identity matrix. Let f be a real-valued function. When talking about the complexity of an operation, we say that it is in $\Theta(f(N,T))$ if there exist two constants $0 < c_1 < c_2$ such that the cost of that operation is in the interval $[c_1 f(N,T), c_2 f(N,T)]$ for all T, N. Finally, δ_{ij} is the Kronecker symbol, equal to 1 when $i = j$ and 0 otherwise.

2 Methods

First, let us precisely write the ICA probabilistic model considered in this article and then derive the maximum likelihood estimation.

2.1 Maximum likelihood for ICA

Given a set of N signals $x_1, \ldots, x_N$ with T samples each, that we can note as $X = [x_1, .., x_N]^\top \in \mathbb{R}^{N \times T}$, we want to find an unmixing matrix $W \in \mathbb{R}^{N \times N}$ such that $WX = Y = [y_1, .., y_N]^\top$ contains mutually independent signals. Here, independence is meant in the statistical sense: the joint probability density function of $[y_1, .., y_N]$, p_Y, is equal to the product of the marginal densities of the y_i's, p_i:

$$p_Y(y_1, .., y_N) = \prod_{i=1}^{N} p_i(y_i) \ . \tag{1}$$

The inference can be done by maximum likelihood in the following way [18]. We postulate a model for X: there is a matrix $A \in \mathbb{R}^{N\times N}$ and independent and identically distributed signals S such that $X = AS$. If such a matrix exists for empirical data, we will say that the ICA model *holds*. Under this model, for each sample t, $S(t)$ has a factorized density $\prod_{i=1}^{N} p_i(s_i)$. The density of the corresponding data sample is computed by a linear change of variables, giving $p_X(X(t)) = |\det(A)|^{-1} \prod_{i=1}^{N} p_i((A^{-1}X(t))_i)$. Finally, we obtain the averaged log-likelihood of the data:

$$\frac{1}{T}\log(p_X(X)) = -\log|\det(A)| + \hat{E}\left[\sum_{i=1}^{N}\log(p_i((A^{-1}X)_i))\right]$$

where $\hat{E}$ denotes the empirical mean. This log-likelihood should be maximized with respect to the variable A. It is however simpler to rewrite it as a function of $W = A^{-1}$ and $Y = WX$. For a given matrix W, the negative averaged log-likelihood of the data X with parameter W is:

$$\mathcal{L}(W) = -\log|\det(W)| - \hat{E}\left[\sum_{i=1}^{N}\log(p_i(y_i))\right] \quad . \tag{2}$$

The maximum likelihood ICA problem is a smooth non-convex optimization problem where $\mathcal{L}$ is the objective function that has to be minimized with respect to W (note that Y depends on W).

Before we go any further, it is important to note that the log-likelihood depends on the density of the sources, which is unknown. There are different ways of dealing with this problem. The simplest one is to use fixed densities as it is the case in standard Infomax, where $-\log(p_i) = 2\log(\cosh(./2))$ (plus an irrelevant normalization constant). This choice may seem arbitrary with strong consequences on the results, but it is shown in [24] that an exact choice of the density functions is not critical for recovering the unmixing matrix in the presence of only super-Gaussians sources. However, it consistently fails at recovering sub-Gaussian sources. The purpose of this contribution is to improve the optimization part of algorithms for maximum likelihood based ICA, this is why we will only consider the simple, yet general, case of fixed densities as in standard Infomax. In our experiments, we will use the exact same density used in Infomax.

2.2 Properties of the objective function

In this section, we detail a number of properties of the objective function (2). First, notice that this optimization problem is not convex: there is not a single optimum. Indeed, if W^* minimizes the objective function, any permutation of the columns of W^* gives another minimizer. Second, one should note that the variable W belongs to the manifold of invertible matrices, with no further specification. This set has a geometry well-suited to matrix multiplications (because the product of two invertible matrices is still invertible), and not so

much to matrix addition (as the sum of two invertible matrices can be singular). This has strong consequences on the gradient of this function.

2.2.1 Gradient

Since the optimization is performed on square matrices, the associated gradient can be viewed as a matrix of the same size. The absolute gradient of this function is the quantity G^A such that for any matrix $\mathcal{E}$, $\mathcal{L}(W+\mathcal{E}) = \mathcal{L}(W) + \langle G^A|\mathcal{E}\rangle + \mathcal{O}(||\mathcal{E}||^2)$. Note that this corresponds to the classical gradient definition $G^A_{ij} = \frac{\partial \mathcal{L}}{\partial W_{ij}}$. However, given the multiplicative geometry of the problem, it is better to work with the relative gradient G such that $\mathcal{L}((I+\mathcal{E})W) = \mathcal{L}(W) + \langle G|\mathcal{E}\rangle + \mathcal{O}(||\mathcal{E}||^2)$. These two gradients are related by the formula $G = G^A W^\top$. From now on we will only use the relative gradient G which is given for (2) by (the computation is derived in the appendix, see also [30]):

$$G_{ij} = \hat{E}[\psi_i(y_i)y_j] - \delta_{ij} \ , \tag{3}$$

where $\psi_i = -\frac{p_i'}{p_i}$ is called the *score function* (which is equal to $\tanh(\cdot/2)$ with standard Infomax densities). It can be rewritten compactly in matrix form: $G(Y) = \frac{1}{T}\psi(Y)Y^\top - I$.

2.2.2 Hessian

Let us define the four following quantities:

$$\begin{cases} \hat{h}_{ijl} = \hat{E}[\psi_i'(y_i)y_j y_l] \text{ , for } 1 \leq i,j,l \leq N \\ \hat{h}_{ij} = \hat{E}[\psi_i'(y_i)y_j^2] \text{ , for } 1 \leq i,j \leq N \\ \hat{h}_i = \hat{E}[\psi_i'(y_i)] \text{ , for } 1 \leq i \leq N \\ \hat{\sigma}_i^2 = \hat{E}[y_i^2] \text{ , for } 1 \leq i \leq N \end{cases} . \tag{4}$$

We define the relative Hessian H for the problem as the fourth order tensor such that: $\mathcal{L}((I+\mathcal{E})W) = \mathcal{L}(W) + \langle G|\mathcal{E}\rangle + \frac{1}{2}\langle \mathcal{E}|H|\mathcal{E}\rangle + \mathcal{O}(||\mathcal{E}||^3)$. After some manipulations that are detailed in the appendix, we obtain:

$$H_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik}\hat{h}_{ijl} \ . \tag{5}$$

There are some interesting things about this expression. First, it is sparse since it has only of the order of N^3 non-zero coefficients: $\delta_{il}\delta_{jk} \neq 0$ for $i = l$ and $j = k$ which corresponds to N^2 coefficients, and $\delta_{ik} \neq 0$ for $i = k$ which happens N^3 times. This means that for a practical application with 100 sources the Hessian easily fits in memory. However, obtaining the Hessian is costly as it requires the computation of $\hat{h}_{ijl}$ for each i, j and l, and hence it takes $\Theta(N^3 \times T)$ operations. Another problem is that the Hessian must be positive definite in order to have a working Newton method, but unfortunately it is not guaranteed here since it is a non-convex problem. While it can be a natural idea to regularize the Hessian with a damping factor as used in the

Levenberg-Marquardt algorithm, it is in practice challenging because computing the smallest eigenvalue of the Hessian is quite expensive given the cubic size of the Hessian matrix. Provided the Hessian can be well regularized, a Newton algorithm remains costly as it involves solving a large linear system at each iteration. Overall, it is perfectly possible to set up a Newton method using the true Hessian, but the cost of the different operations involved makes it slow. This is what motivates the use of Hessian approximations.

2.2.3 Hessian Approximations

We define a first approximation of H replacing $\hat{h}_{ijl}$ by $\delta_{jl}\hat{h}_{ij}$. We denote that approximation by $\tilde{H^2}$:

$$\tilde{H^2}_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik}\delta_{jl}\hat{h}_{ij} \ . \tag{6}$$

A second approximation, $\tilde{H^1}$, is obtained by replacing $\hat{h}_{ij}$ by $\hat{h}_i\hat{\sigma}_j^2$ for $i \neq j$:

$$\begin{cases} \tilde{H^1}_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik}\delta_{jl}\hat{h}_i\hat{\sigma}_j^2 \text{ , for } i \neq j \\ \tilde{H^1}_{iiii} = 1 + \hat{h}_{ii} \end{cases} . \tag{7}$$

It is always true that $\hat{h}_{iii} = \hat{h}_{ii}$, and when the signals are independent, $\hat{h}_{ijl} = \delta_{jl}\hat{h}_{ij} = \delta_{jl}\hat{h}_i\hat{\sigma}_j^2$ asymptotically for $i \neq j$. This means that **the two approximations asymptotically match the true Hessian if the signals are independent**. In particular, if an iterative algorithm converges to a solution on a problem where the ICA model holds, the sequence of Hessian approximations converges towards the true Hessian of the objective function at the optimum.

They are less costly to obtain than H. $\tilde{H^2}$ requires the computation of $\hat{h}_{ij}$ for all (i, j), so its computation is in $\Theta(N^2 \times T)$. We refer to it as $\tilde{H^2}$ because of the exponent 2 in the complexity. $\tilde{H^1}$ is even faster to obtain, because is only requires the computation of the $\hat{h}_i$ and $\hat{\sigma}_i$. Its computation cost is thus in $\Theta(N^1 \times T)$.

These approximations are block diagonal: For a pair (i, j) with $i \neq j$, the only nonzero coefficients in $\tilde{H}_{ijkl}$ are for $(k, l) = (i, j)$ and $(k, l) = (j, i)$. Because of that structure, it is simpler to invert them than H: it amounts to inverting each 2×2 block on the diagonal, so the complexity of inversion is $\Theta(N^2)$. Finally, they are sparser than the true Hessian, and hence can be stored more efficiently.

Unfortunately, these approximations can be quite far from H. In most practical applications, the ICA model does not hold and then even at the likelihood maximum, $\hat{h}_{ijl} \neq \delta_{jl}\hat{h}_{ij} \neq \delta_{jl}\hat{h}_i\hat{\sigma}_j^2$. Even though $\tilde{H^2}$ more expensive to compute that $\tilde{H^1}$, it is also a better approximation in the sense that it is equal to H on the diagonal blocks, while $\tilde{H^1}$ is not.

2.2.4 Regularization of Hessian Approximations

It is possible to show that if the density model were perfect, $\tilde{H^1}$ would asymptotically be (when the number of samples $T \to \infty$) a positive definite matrix if

Algorithm 1: Regularization procedure

Input : Eigenvalue threshold $\lambda_{min} > 0$, approximate Hessian H_k
for *Each pair* (i, j) **do**
  Compute λ using (9);
  if $\lambda < \lambda_{min}$ **then**
    Add $(\lambda_{min} - \lambda)I_2$ to the block (i, j) of H_k ;
  end
end
Output: Regularized H_k

at most one source is Gaussian (see [18]). In our case, we have three problems that prevent the positiveness of the approximate Hessian. First, the previous result is only true asymptotically. Second, it needs to have the true score model, which is not the case with fixed scores. Finally, it is common to get data where there are more than one Gaussian signal in the mixture.

Let us study what happens when there are two Gaussian signals of indexes i and j in the mixture. Suppose that we have managed to set the gradient to 0. Looking at the diagonal values, we get that $\hat{E}[\psi_i(y_i)y_i] = \hat{E}[\psi_j(y_j)y_j] = 1$. Now, if we call σ_i the standard deviation of the i^{th} signal, an integration by parts shows that $\mathbb{E}[\psi_i(y_i)y_i] = \mathbb{E}[\psi_i'(y_i)]\sigma_i^2$ if (y_i) is Gaussian, regardless of ψ_i. Now, if both i and j are Gaussian signals, the corresponding 2×2 block in the Hessian (or approximate Hessian) is:

$$\begin{pmatrix} \frac{\sigma_j^2}{\sigma_i^2} & 1 \\ 1 & \frac{\sigma_i^2}{\sigma_j^2} \end{pmatrix} \ , \tag{8}$$

which is singular (its determinant cancels). This means that as the gradient goes to 0, if there are at least two Gaussian sources, one eigenvalue of the Hessian (or approximate Hessian) will tend towards 0. This naturally leads to erratic behavior when computing $H^{-1}G$. Regularization is needed to prevent this.

A good thing about the two Hessian approximations is that they can be diagonalized without effort: it amounts to diagonalizing each of the 2×2 blocks. Elementary linear algebra shows that the smallest eigenvalue associated to the block (i, j) is given by:

$$\lambda_{ij} = \frac{1}{2}(a_{ij} + a_{ji} - \sqrt{(a_{ij} - a_{ji})^2 + 4}) \ , \tag{9}$$

with $a_{ij} = \tilde{H}_{ijij}$, 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 1. Note that the blocks with positive and big enough eigenvalues are left untouched, while the other blocks have their spectrum shifted so that their smallest eigenvalue is equal to the minimum value λ_{min}.

2.3 First order methods

In the following, every optimization algorithm relies on a line search, which will be specified later in subsection 2.5 after we have introduced the main algorithmic ideas.

2.3.1 Gradient descent

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

$$W \leftarrow \left(I - \alpha \left(\frac{1}{T} \psi(Y) Y^\top - I \right) \right) W \ ,$$

where $\alpha > 0$ is a step size found by line search or an annealing policy.

One of the advantage of gradient descent is that it can cope with the non-convexity of the problem: provided that α is small enough, each step will effectively decrease the objective function and it will converge to a point with zero-gradient (cf. Prop. 1.2.1 in [31]). The downside is that its convergence speed is at most linear, and that it requires a fine-tuned annealing policy or line search to ensure that α is well-chosen. These two problems are generally solved by second order methods where the Hessian captures the local curvature of the minimized function.

2.3.2 Infomax

We now give a brief explanation on how the Infomax algorithm runs in practice. The philosophy of the algorithm is to use the gradient as a descent direction. However, the main difference with the algorithm that we have just shown is that it is a stochastic gradient algorithm: the gradient G in the previous formula is not computed using all the samples, but rather on a mini-batch of size T'. This means that at each iteration of the algorithm, the samples are randomly split in groups of length T'. Then, the gradient for each group G' is computed and a stochastic gradient step $W \leftarrow (I - \alpha G')W$ is performed.

The stochasticity of Infomax has benefits and drawbacks. For a thorough review about what stochasticity brings, see [32]. We here present a summary of the properties of stochastic gradient. On the good side, it makes the algorithm very fast for the first few passes on the full data. This happens 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 set might be too costly, and then resorting to stochastic techniques is one way of coping with the issue.

Unfortunately, it also comes with some disadvantages. The first one is that a plain stochastic gradient method with fixed batch size cannot exactly converge to a local minimum of the objective function. Another way to put it is that across iterations, the true gradient computed with the full set will not go to 0, but instead will reach a plateau.

Another difficulty that arises with stochastic algorithms is 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 [32]. 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 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 is in practice problematic [29].

2.4 Second order methods

Second order methods exploit the curvature information carried in the Hessian and its approximations in order to converge in fewer iterations.

2.4.1 Empirical Quasi-Newton method

The simplest way to take advantage of the Hessian approximations is to use them as replacement of H in Newton algorithm, for which the descent direction is given by $-\tilde{H}^{-1}G$. We will refer to this as the elementary quasi-Newton method, which is detailed in Algorithm 2. Note that any Hessian approximation can be used as long as it is guaranteed to be positive definite. This is the optimization procedure performed by the AMICA algorithm [27], which uses the crudest approximation $\tilde{H}^1$.

Algorithm 2: Elementary quasi-Newton

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 relative gradient G_k using (3);
- Compute Hessian approximation $\tilde{H}_k$ using (6) or (7);
- Regularize $\tilde{H}_k$ using algorithm 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 $Y \leftarrow (I + \alpha_k p_k) Y$;

end

Output: Y, W_k

One particularly clear limitation of first order gradient methods is the *oscillating* behavior of the successive directions. Indeed, in *valleys*, the next descent step tends to undo the benefit from the last step. To illustrate this, we ran a gradient descent algorithm on a simple synthetic ICA problem with $N = 30$ sources, all of which come from a Laplace density ($p_i(x) \propto \exp(-|x|)$). This

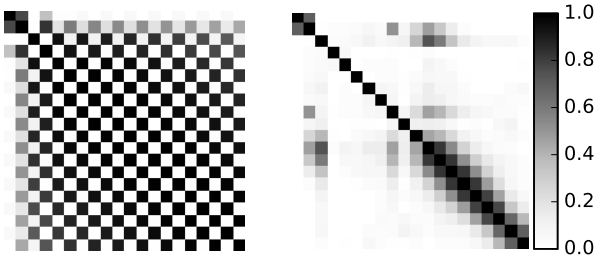

Figure 1: Cosine of the angles between successive directions across iterations of the algorithms. Left: simple gradient descent algorithm. Right: elementary quasi-Newton (algorithm 2). A black pixel at the coordinate (i, j) means that the i^{th} and j^{th} direction used by the algorithm at each step are almost aligned. A white pixel means that they are orthogonal.

density is super-Gaussian, and thus the sources can be recovered with our score model. We ran the algorithm for only 20 iterations, using a costly line-search that finds a step α almost optimal: $\alpha \simeq \arg\min \mathcal{L}((I + \alpha G)W)$ at each step. Then, we compute the cosine of angles between each descent direction. Denoting D_i the descent direction at iteration i, we compute $\cos(\widehat{(D_i, D_j)}) = \frac{\langle D_i | D_j \rangle}{||D_i|| ||D_j||}$ for all (i, j) between 1 and 20 . This gives us a 20×20 symmetric matrix. From Cauchy-Schwartz inequality, the entries in the matrix are less than 1, and are close to 1 when D_i and D_j are almost aligned. In Figure 1, this matrix is displayed on the left. We can see that the directions D_i, D_{i+2}, $D_{i+4} \dots$ are essentially the same and that it takes quite a few iterations to find new descent directions. This means that the algorithm wastes a lot of time searching in the same directions. On the right, we repeated the same experiment with the elementary Quasi-Newton method. We can see that the pattern changes: each descent direction is new and has not been explored before. This highlights the well-documented "zig-zagging" behavior of the gradient descent algorithm with an intricate objective function.

2.4.2 Preconditioned L-BFGS method

Most widely used quasi-Newton methods manage to estimate the local curvature of the objective function without explicitly computing its Hessian [33]. Indeed, popular methods such as DFP [34, 35, 36] or BFGS [37, 35, 38, 39] build an approximation of the Hessian using solely function and gradient evaluations performed during optimization.

For most problems, at the beginning of the iterations no approximation of the Hessian (or of the inverse of the Hessian) is available and it is commonly set to a scaled identity matrix. It is therefore natural to wonder if having an inexpensive way to compute approximations of the Hessian can be of any help here. We consider in the following the popular L-BFGS [40] algorithm.

Rather than storing all the updates of the Hessian approximation leading to

Algorithm 3: Preconditioned L-BFGS

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 (3);

Compute Hessian approximation $\tilde{H}_k$ using (6) or (7);

Regularize $\tilde{H}_k$ using algorithm 1;

Compute the search direction $p_k = -(\tilde{H}_k^m)^{-1} G_k$ using L-BFGS formula in algorithm 4;

Compute the step length α_k using a line search;

Set $W_{k+1} = (I + \alpha_k p_k) W_k$;

end

Output: Y, W_k

a dense matrix potentially too big to fit in memory like BFGS does, the L-BFGS algorithm only stores the last m updates and then relies on recursive inversion algorithm. The integer m is referred to as the memory parameter. From the m previous iterates and gradient evaluations, it builds a low rank approximation of the Hessian, which we denote H_k^m. Let us define y_i as $G_i - G_{i-1}$, the s_i as $a_i p_i$ (this is the "relative" update of the unmixing matrix between two iterations) and the ρ_i as $1/\langle s_i | y_i \rangle$. These notations are the same that are used in [33]. Our preconditioned L-BFGS algorithm that exploits the Hessian approximations to initialize the recursive formula of L-BFGS is summarized in algorithms 3 and 4. 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 $\tilde{H}_k$.

2.5 Line search

These algorithms rely 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 [41, 33] in order to prove convergence. The line search procedure proposed by Moré and Thuente [42] is generally considered to be a very 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, we must compute the value of the objective function as well as the gradient, which can be costly.

A simpler line search strategy is backtracking. We start from $\alpha = 1$. If for that value, there is a decrease in the objective, we use this step size. Else, we divide α by 2 and start again. This method only requires a loss evaluation at each step size, but it does not enforce Wolfe conditions.

In practice, we allow to each algorithm a number of attempts (candidate values for α) in order to reach a suitable value. If that number of attempts is

Algorithm 4: 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 $\forall i \in \{k-m, \dots, k-1\}$.

Set $q = G_k$;

for *i=k-1,. . . ,k-m* **do**

  Compute $a_i = \rho_i \langle s_i | q \rangle$;

  Set $q = q - a_i y_i$;

end

Set $r = \tilde{H}_k^{-1} q$;

for *i=k-m,. . . ,k-1* **do**

  Compute $\beta = \rho_i \langle y_i | r \rangle$;

  Set $r = r + s_i(a_i - \beta)$;

end

Output: $r = p_k$

exceeded, that means that the objective function has a pathological behavior in the search direction. Through experiments, we observed that this scenario happens each time under the same circumstances, which is when the minimum of the objective function is reached for a very small value $\alpha_{min} \ll 1$, instead of the typical "Newton value" for a quadratic function $\alpha_{min} = 1$. If we were to select such a step size, then the algorithm would not move much, and might get stuck for a long time in that problematic zone. Instead, if the maximal number of attempts of the line search is reached, we fall back to using the gradient as descent direction. We do so because we have observed that the direction given by the gradient, although not optimal, is a direction along which the objective function is smooth. Thus, taking a step in that direction allows the algorithm to move away from the previous pathological zone.

Overall, for empirical quasi-Newton and L-BFGS, we found that the backtracking line search achieves better overall results, because it is slightly less costly, and less likely to find the correct minimum of a problematic direction. Another important motivation is that these algorithms make an underlying quadratic approximation of the objective function, for which the step $\alpha = 1$ is the best step. In practice, the step $\alpha = 1$ is indeed a good step in most cases.

3 Experiments

All the following experiments have been performed on the same computer using the Python programming language. For optimized numerical code we used Numpy [43] using Intel MKL as linear algebra backend library, and the numexpr package [1] to optimize CPU cache. It was particularly efficient to compute $\log\cosh(y_i(t)/2)$ and $\tanh(y_i(t)/2)$ $\forall i, t$. All experiments were run using only one core of an Intel Core i7-6600U.

[1] `https://github.com/pydata/numexpr`

3.1 Preprocessing

A good preprocessing of the data can help the algorithms by starting from a good initial point. The preprocessing we apply to all the signals is the standard preprocessing for ICA and is as follows. Given the input matrix X, we first set the mean of the signals to 0 by subtracting its mean to each row. Then, we whiten the signals. This means that we find a linear transform of the mixed signals such that the covariance matrix of the transformed signals is the identity matrix. In other words, we decorrelate the signals and scale them to have unit variance. Finding such a transform is done as follows. Let us denote by X the signal matrix. Its covariance matrix, $C = \frac{1}{T}XX^\top$, is symmetric positive definite, and thus admits the eigenvector decomposition $C = U^\top DU$ where D is diagonal with positive entries, and U is an orthogonal matrix : $UU^\top = I$. If we define $X_{white} = D^{-\frac{1}{2}}UX$, the covariance matrix of X_{white} is equal to identity. We call that particular transform, given by $D^{-\frac{1}{2}}U$, the *sphering whitener*. Note that any other whitening transform can be obtained by multiplying the sphering whitener by an orthogonal matrix. An alternative whitener is the Principal Component Analysis (PCA) whitener given by $U^\top D^{-\frac{1}{2}}U$.

3.2 Simulation study

In this section, we present results obtained on synthetic data. The general setup is the following: we choose the number of sources N, the number of samples T and a density for each source. For each of the N densities, we generate T random samples which gives the N simulated source signals. 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. Finally, the preprocessing explained above (3.1) is applied.

These signals are then fed to the algorithms, and the gradient infinite norm, which is defined as $\max_{ij}(|G_{ij}|)$, is tracked over C.P.U. time and iterations. Note that in practice, we do not have access to the minimum value of the loss function nor to one of its minimizers, hence it is impossible to track the difference between the current iterate and the final value, or the distance of the current objective value to its global minimum. This is why we only quantify the norm of the gradient. We repeat that experiment 100 times, changing each time the seed generating the random signals. For each algorithm, we end up with 100 curves of the gradient w.r.t. time and iterations. To obtain readable figures, we only display the median of these curves over the 100 experiments (50 were above the curve shown, and 50 below).

We performed 3 different experiments:

- **Experiment A**: $N = 40$ independent sources of $T = 10000$ samples. The density of each source is the same, given by $p(x) = \frac{1}{2}\exp(-|x|)$.
- **Experiment B**: $N = 15$ independent sources of $T = 1000$ samples. The density of the 5 first sources is given by $p(x) = \exp(-|x|)$, the 5 next

sources are Gaussian, and the density of the last 5 sources is given by $p(x) \propto \exp(-|x^3|)$.

- **Experiment C**: $N = 40$ independent sources of $T = 5000$ samples. The density of the source i is given by $p_i = \alpha_i \mathcal{N}(0,1) + (1-\alpha_i)\mathcal{N}(0,\sigma^2)$, where α_i is a sequence of linearly spaced values such that $\alpha_1 = 0.5$ and $\alpha_n = 1$. We set $\sigma = 0.1$.

In experiment A, the ICA model 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 cannot because they are Gaussian, and the last 5 cannot be recovered either because the sources are sub-Gaussian.

Finally, in experiment C, the ICA model holds, but the sources are becoming more and more Gaussian. Because of the limited number of samples, some of the most Gaussian sources cannot be distinguished from an actual Gaussian signal.

We try 6 algorithms for those experiments. We run the presented elementary Quasi-Newton and preconditioned L-BFGS. The first algorithm is run only with the approximation $\tilde{H}^1$, while we try the two different approximations $\tilde{H}^1$ and $\tilde{H}^2$ for L-BFGS. We also use the unpreconditioned versions of these algorithms: a simple gradient descent, and the regular L-BFGS method. The memory size for L-BFGS has been set to $m = 7$. We empirically found that it had little effect on the performance in the range $3 \leq m \leq 15$. Finally, we also run Infomax, as described in section 2.3.2, with a mini-batch size set to a third of the number of samples (we found that the mini-batch size has little effect except when it is very close to the number of samples). Parameters such as the learning rate were set using their default values in the EEGLab matlab toolbox [28]. Note that the gradient norm displayed is the norm of the full gradient on the whole data, computed a posteriori after each run.

For the gradient descent, we used an oracle line-search: we run the algorithm using an expensive but accurate line-search to find the best step at each iterations, but do not take the line search time into account in the timing. In that way, we place the gradient descent under the best possible light, because any practical line search will perform worse than this one.

The results of the three experiments are shown in Figure 2. We can see that the three methods informed by the low-cost Hessian approximations perform better than their uninformed counterparts on the first experiment. In terms of number of iterations, the two different approximations yield very similar results, and we can expect them to follow quite similar trajectories. Thus, in terms of running time it is the simplest method with approximation $\tilde{H}^1$ which performs best, *i.e.,* the elementary quasi-Newton also used by the AMICA method. We observe that all quasi-Newton algorithms have a quadratic convergence. In particular, the elementary quasi-Newton's convergence rate is quadratic because at the optimum when the ICA model holds, the approximate Hessian asymptotically coincides with the true Hessian. The most logical explanation behind the

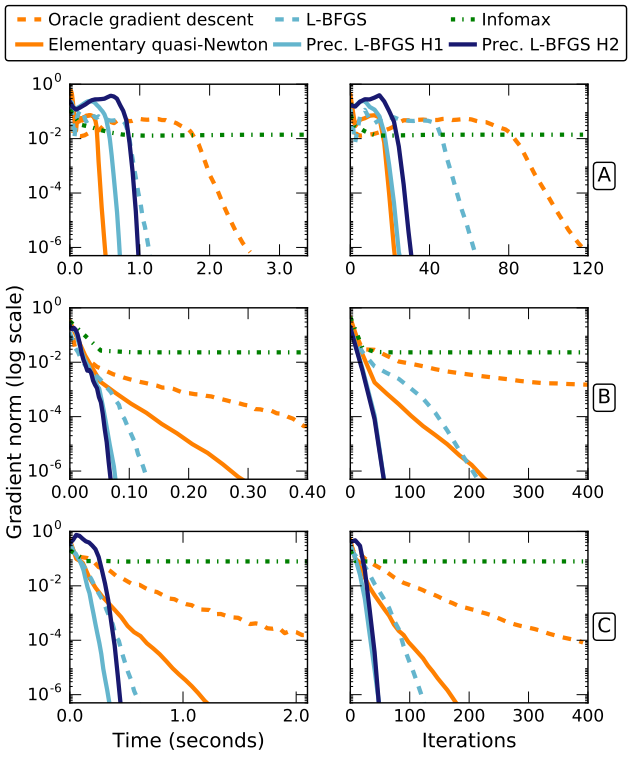

Figure 2: Comparison of the optimization algorithms on three synthetic experiments. Top: experiment A, middle: experiment B, bottom: experiment C. Left: gradient as a function of time, right: gradient as a function of iterations (pass on the full set for Infomax). Solid lines correspond to algorithms informed of the approximate Hessian, dashed lines are their standard counterparts. Lines of the same color correspond to algorithms of the same family.

superiority of elementary quasi-Newton is that the approximation of the Hessian is such a good approximation in that case that L-BFGS fails to produce a better curvature. Finally, as expected, Infomax quickly diminishes the gradient in the first steps, but then reaches a plateau at a quite high value of the gradient norm.

Experiments B and C put the algorithms in a difficult situation, and both yield similar conclusions. We observe that the gradient descent progresses really slowly. The standard L-BFGS algorithm manages to achieve honorable convergence time, while being totally uninformed by the problem geometry. The elementary quasi-Newton is no longer in good position. In particular, its convergence rate drops from quadratic to linear. This happens simply because the ICA model does not hold anymore, hence the sequence of approximate Hessians does not converge to the true Hessian as the gradient goes to 0. This approximation is still much better than the identity, so the elementary quasi-Newton is faster than the gradient descent, but we can see that preconditioned L-BFGS manages to find much better curvature approximations. Indeed, the convergence of this algorithm still seems quadratic.

Finally, we can wonder if using $\tilde{H}^2$ rather than $\tilde{H}^1$ is a waste of time: for those experiments the algorithms take about the same number of iterations to converge, but the former has a more expensive cost per iteration. However, as we will now see on real data, using $\tilde{H}^2$ can lead to some important convergence improvements.

3.3 Results on EEG data

We now present the results of our algorithms on 13 EEG datasets publicly available at `https://sccn.ucsd.edu/wiki/BSSComparison` [6] . Each recording contains $n = 72$ signals, of length $T \simeq 300000$ samples. EEG measures the changes of electric potential induced by brain activity. For such data, the ICA model does not perfectly hold. In addition, they are contaminated by a lot of noise and artifacts. Still, it has been shown that ICA succeeds at extracting meaningful and biologically plausible sources from these mixtures [44, 3, 6].

The signals are very long, requiring an excessive amount of time for the slower algorithms to process, so we first down-sampled the data by a factor 4, leaving us with on average $T \simeq 75000$ samples per recording. The signals are then preprocessed as explained in 3.1.

We then ran the two versions of preconditioned L-BFGS on the full non-downsampled data, to get a better sense of their performances on large datasets. Results are displayed on the top and middle row of Figure 3.

The conclusions are similar to the ones drawn with the synthetic experiments B and C: preconditioned L-BFGS is more efficient than the other algorithms. However, we can observe that $\tilde{H}^2$ brings an improvement compared to $\tilde{H}^1$.

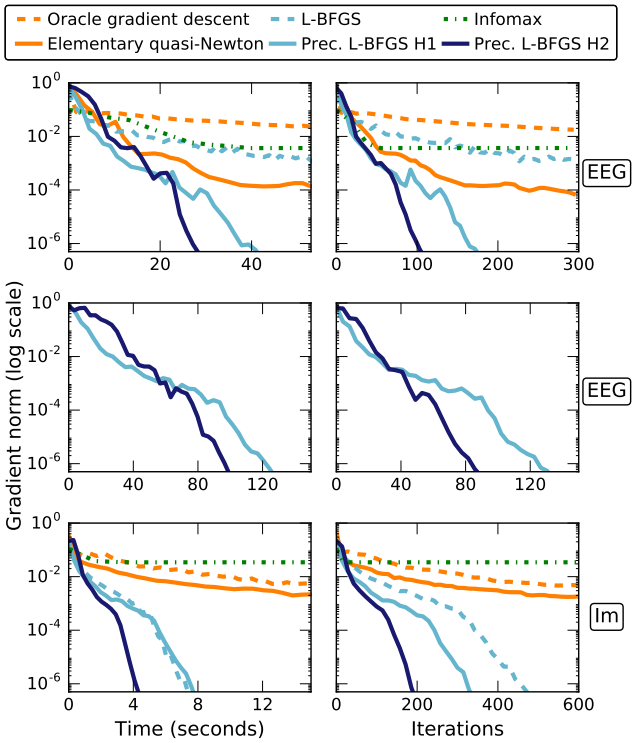

Figure 3: Comparison of the optimization algorithms on real data. Top: EEG dataset down-sampled by a factor 4 ($T \simeq 75000$ samples). Middle: full EEG dataset ($T \simeq 300000$ samples). Bottom: Image patches dataset. Left: gradient w.r.t. time, right: gradient w.r.t. iterations (pass on the full data for Infomax). Solid lines correspond to algorithms informed of the approximate Hessian, dashed lines are their standard counterparts. Lines of the same colors correspond to algorithms of the same family.

3.4 Results on natural images

Given a grayscale image, we extract T square patches (contiguous squares of pixels) of size (s, s). If we consider each patch as a vector, we end up with T samples of size s^2. We can then perform ICA on those data, assuming that each vectorized sample comes from a mixture of independent sources. The columns of the unmixing matrix can then be thought of as features, or dictionary atoms, learned from random patches.

We run all the previous algorithms on a set of 100 natural images of open country, available here: `http://cvcl.mit.edu/database.htm` [45]. We used $T = 30000$ patches of side 8×8 pixels, giving us a signal matrix of size 64×30000. The patches are all centered and scaled so that their mean and variance equal 0 and 1 respectively. The signals are then preprocessed (3.1).

Results are shown at the bottom of figure 3. We can see that the convergence is extremely slow for Infomax, for the gradient descent and for the elementary quasi-Newton method. Interestingly, we can see that while L-BFGS preconditioned with $\tilde{H}^1$ does less iterations than the unpreconditioned one, this does not compensate for the cost of computing $\tilde{H}^1$. The two algorithms perform indeed similarly in terms of time. However, using $\tilde{H}^2$ rather than $\tilde{H}^1$ provides an important gain, almost halving the number of iterations and the time needed to reach a stationary point.

3.5 Benefits of canceling the gradient

Preconditioned L-BFGS algorithm manages to quickly find a point at which the gradient vanishes. Infomax is able to decrease the gradient in a few steps, but then reaches a plateau where the gradient norm freezes. It is natural to ask ourselves to which extent does actually reaching a local minimum change the conclusions about the sources.

In EEG processing, the mixing matrix found by ICA carries important biological informations [6]. Recently, it was shown in [29] that on EEG data, the matrices returned by Infomax and the matrices obtained after pushing the convergence can lead to different neuroscientific interpretations. In this work, we propose another experiment to illustrate the importance of reaching a good local minimum. An algorithm that stops too early returns necessarily an estimate influenced by the initialization. In other words, one hopes that pushing the convergence reduces the role played by initialization in the scientific interpretation of the results.

To quantify this, we ran the preconditioned L-BFGS algorithm on the EEG signals twice, the first time after using a sphering whitener, the second time using a PCA whitener. We then compared the different mixing matrices given by the two runs, for various gradient norms. Denoting W_{PCA} and W_{sph} the two estimated matrix yielding a comparable gradient norm, we computed $\mathcal{T} = W_{sph}.W_{PCA}^{-1}$. If the sources recovered by the two algorithms were exactly the same up to order and scale, this matrix would be a permutation matrix times a diagonal matrix. In order to have readable matrix plots, we permute the

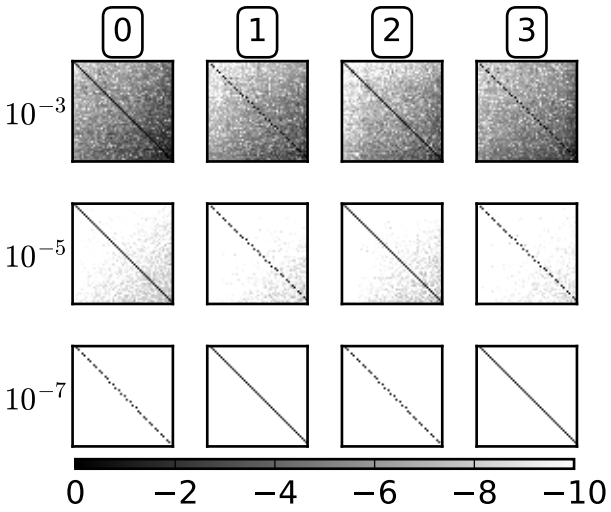

Figure 4: Comparison of the mixing matrices found by maximum likelihood maximization on EEG data with two different initializations, for different level of gradients. Each column correspond to one subject, each line to the gradient level indicated by the number on the left. The matrices are displayed in logarithmic scale: when the matrix is white with a black diagonal, it means that it is the identity and hence that the two initializations led to the exact same local maximum of the likelihood.

rows and columns of $\mathcal{T}$ to end up with a matrix with its larger coefficients on its diagonal. We then divide each row by the value of the diagonal. Finally, one last permutation is performed so that the rows with the largest off diagonal values are at the bottom. Thus, after this process, if $\mathcal{T}$ were exactly a permutation plus scale matrix, it would become the identity matrix.

We repeat that experiment for various gradient norms, and across all the 13 EEG recordings. For four subjects out of the 13, we obtain striking results, displayed in figure 4. For these subjects, we can clearly see that as the gradient goes to 0, the two algorithms initialized differently converge to the exact same solution.

It is also interesting to see that for a gradient norm of 10^{-3} as classically observed after running Infomax, the two transforms are quite different. This demonstrates that pushing the convergence of the ICA solvers favors the replicability of neuroscientific results.

4 Conclusion

In this work, two Hessian approximations of the objective function of maximum likelihood based ICA have been considered to accelerate estimation algo-

rithms. Experiments have revealed that these approximations can be too rough, especially when the ICA model does not hold, which is often the case on non-simulated data. To go beyond the elementary quasi-Newton, we introduced a preconditioned optimization algorithm based on the L-BFGS method. Through simulations and the analysis of EEG data and images, we have shown that this method outperforms off-the-shelf optimization algorithms such as regular L-BFGS but also the elementary quasi-Newton method which relies only on the Hessian approximation.

Appendix

Relative gradient and relative Hessian

The objective function is given by $\mathcal{L}(W) = -\log|\det(W)| - \hat{E}\left[\sum_{i=1}^{N} \log(p_i(y_i))\right]$ (2). The relative gradient G and the relative Hessian H at the point W are defined as the only matrix / fourth order tensor verifying:

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

Let us denote by ε_{ij} the coefficients of $\mathcal{E}$. We can write $\mathcal{L}((I+\mathcal{E})W) = -\log|\det((I+\mathcal{E})W)| - \hat{E}\left[\sum_{i=1}^{N} \log(p_i(y_i + \sum_{j=1}^{N} \varepsilon_{ij} y_j))\right]$. The $\log\det$ part can be taken care of as follows:

$$\log|\det((I+\mathcal{E})W)| = \log|\det(W)| + \log|\det(I+\mathcal{E})|$$

We thus have to develop to the second order $\log|\det(I+\mathcal{E})|$, which can be done by hand. We can also use the convenient relationship $\log|\det(\exp(\mathcal{E}))| = \mathrm{Tr}(\mathcal{E})$, which we can rewrite at the second order as $\log|\det(I+\mathcal{E}+\frac{1}{2}\mathcal{E}^2)| = \mathrm{Tr}(\mathcal{E})$. Now, using the change of variable $\mathcal{E} + \frac{1}{2}\mathcal{E}^2 \leftarrow \mathcal{E}$, which gives $\mathcal{E} \leftarrow \mathcal{E} - \frac{1}{2}\mathcal{E}^2$ at the second order, we obtain:

$$\log|\det(I+\mathcal{E})| = \mathrm{Tr}(\mathcal{E}) - \frac{1}{2}\,\mathrm{Tr}(\mathcal{E}^2)\ .$$

We can write the above expression in terms of scalar product: $\mathrm{Tr}(\mathcal{E}) = \sum_{i,j} \delta_{ij}\varepsilon_{ij} = \langle \mathcal{E}|I_N\rangle$, and $\mathrm{Tr}(\mathcal{E}^2) = \sum_{i,j} \varepsilon_{i,j}\varepsilon_{j,i} = \langle \mathcal{E}|H^{ld}|\mathcal{E}\rangle$ where $H^{ld}_{ijkl} = \delta_{il}\delta_{jk}$.

For the second part, we use the Taylor expansion $-\log(p_i(y_i+\varepsilon)) \simeq -\log(p_i(y_i)) + \psi_i(y_i)\varepsilon + \frac{1}{2}\psi_i'(y_i)\varepsilon^2$ which gives $-\log(p_i(y_i + \sum_{j=1}^{N} \varepsilon_{ij} y_j)) \simeq -\log(p_i(y_i)) + \sum_{j=1}^{N} \varepsilon_{ij}\psi_i(y_i)y_j + \frac{1}{2}\sum_{k,l=1}^{N} \varepsilon_{ik}\varepsilon_{il}\psi_i'(y_i)y_k y_l$.

Summing over all sources i and samples t:

$$\begin{aligned}
&- \hat{E}\left[\sum_{i=1}^{N} \log(p_i(y_i + \sum_{j=1}^{N} \varepsilon_{ij} y_j))\right] \\
= &- \hat{E}\left[\sum_{i=1}^{N} \log(p_i(y_i))\right] + \sum_{i,j} \varepsilon_{ij} \hat{E}\left[\psi_i(y_i) y_j\right] \\
&+ \frac{1}{2} \sum_{i,j,l} \varepsilon_{ij} \varepsilon_{il} \hat{E}\left[\psi_i'(y_i) y_j y_l\right]
\end{aligned}$$

We can then write this in a scalar product form: $\sum_{i,j} \varepsilon_{ij} \hat{E}\left[\psi_i(y_i) y_j\right] = \langle G^p | \mathcal{E} \rangle$ with $G^p_{ij} = \hat{E}\left[\psi_i(y_i) y_j\right]$, and $\sum_{i,j,l} \varepsilon_{ij} \varepsilon_{il} \hat{E}\left[\psi_i'(y_i) y_j y_l\right] = \langle \mathcal{E} | H^p | \mathcal{E} \rangle$ where $H^p_{ijkl} = \delta_{ik} \hat{E}\left[\psi_i'(y_i) y_j y_l\right]$. We can finally sum the two previous quantities to find:

$$\begin{cases} G_{ij} = -\delta_{ij} + \hat{E}\left[\psi_i(y_i) y_j\right] \\ H_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik} \hat{E}\left[\psi_i'(y_i) y_j y_l\right] \end{cases}.$$

Acknowledgment

This work was supported by the Center for Data Science, funded by the IDEX Paris-Saclay, ANR-11-IDEX-0003-02, and the European Research Council (ERC SLAB-YStG-676943).

References

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

[2] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural networks*, vol. 13, no. 4, pp. 411–430, 2000.

[3] S. Makeig, T.-P. Jung, A. J. Bell, D. Ghahremani, and T. J. Sejnowski, "Blind separation of auditory event-related brain responses into independentcomponents," *Proceedings of the National Academy of Sciences (PNAS)*, vol. 94, no. 20, pp. 10 979–10 984, 1997.

[4] V. Calhoun, T. Adali, G. Pearlson, and J. Pekar, "A method for making group inferences from functional MRI data using independent component analysis," *Human Brain Mapping*, vol. 14, no. 3, pp. 140–151, 2001.

[5] C. F. Beckmann and S. M. Smith, "Probabilistic independent component analysis for functional magnetic resonance imaging," *IEEE Transactions on Medical Imaging*, vol. 23, no. 2, pp. 137–152, Feb 2004.

[6] A. Delorme, J. Palmer, J. Onton, R. Oostenveld, and S. Makeig, "Independent EEG sources are dipolar," *PloS one*, vol. 7, no. 2, p. e30135, 2012.

[7] D. Nuzillard and A. Bijaoui, "Blind source separation and analysis of multispectral astronomical images," *Astronomy and Astrophysics Supplement Series*, vol. 147, no. 1, pp. 129–138, 2000.

[8] D. Maino, A. Farusi, C. Baccigalupi, F. Perrotta, A. Banday, L. Bedini, C. Burigana, G. De Zotti, K. Górski, and E. Salerno, "All-sky astrophysical component separation with fast independent component analysis (FASTICA)," *Monthly Notices of the Royal Astronomical Society*, vol. 334, no. 1, pp. 53–68, 2002.

[9] A. Cadavid, J. Lawrence, and A. Ruzmaikin, "Principal components and independent component analysis of solar and space data," *Solar Physics*, vol. 248, no. 2, pp. 247–261, 2008.

[10] V. Vrabie, C. Gobinet, O. Piot, A. Tfayli, P. Bernard, R. Huez, and M. Manfait, "Independent component analysis of raman spectra: Application on paraffin-embedded skin biopsies," *Biomedical Signal Processing and Control*, vol. 2, no. 1, pp. 40 – 50, 2007.

[11] D. N. Rutledge and D. J.-R. Bouveresse, "Independent components analysis with the JADE algorithm," *TrAC Trends in Analytical Chemistry*, vol. 50, pp. 22–32, 2013.

[12] S.-I. Lee and S. Batzoglou, "Application of independent component analysis to microarrays," *Genome biology*, vol. 4, no. 11, p. R76, 2003.

[13] M. Scholz, S. Gatzek, A. Sterling, O. Fiehn, and J. Selbig, "Metabolite fingerprinting: detecting biological features by independent component analysis," *Bioinformatics*, vol. 20, no. 15, pp. 2447–2454, 2004.

[14] T.-P. Jung, S. Makeig, C. Humphries, T.-W. Lee, M. J. McKeown, V. Iragui, and T. J. Sejnowski, "Removing electroencephalographic artifacts by blind source separation," *Psychophysiology*, vol. 37, no. 2, pp. 163–178, 2000.

[15] A. Hyvärinen, P. O. Hoyer, and E. Oja, "Sparse code shrinkage: Denoising by nonlinear maximum likelihood estimation," *Advances in Neural Information Processing Systems*, pp. 473–479, 1999.

[16] J. Mairal, G. Sapiro, and M. Elad, "Learning multiscale sparse representations for image and video restoration," *Multiscale Modeling & Simulation*, vol. 7, no. 1, pp. 214–241, 2008.

[17] A. Hyvärinen and E. Oja, "Independent component analysis: Algorithms and applications," *Neural Netw.*, vol. 13, no. 4-5, pp. 411–430, May 2000.

[18] D. T. Pham and P. Garat, "Blind separation of mixture of independent sources through a quasi-maximum likelihood approach," *IEEE Transactions on Signal Processing*, vol. 45, no. 7, pp. 1712–1725, 1997.

[19] A. Hyvärinen, "The fixed-point algorithm and maximum likelihood estimation for independent component analysis," *Neural Processing Letters*, vol. 10, no. 1, pp. 1–5, 1999.

[20] J.-F. Cardoso, "Infomax and maximum likelihood for blind source separation," *IEEE Signal processing letters*, vol. 4, no. 4, pp. 112–114, 1997.

[21] A. Hyvarinen, "Fast and robust fixed-point algorithms for independent component analysis," *IEEE Transactions on Neural Networks*, vol. 10, no. 3, pp. 626–634, 1999.

[22] A. J. Bell and T. J. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural computation*, vol. 7, no. 6, pp. 1129–1159, 1995.

[23] J. A. Palmer, K. Kreutz-Delgado, and S. Makeig, "AMICA: An adaptive mixture of independent component analyzers with shared components," Tech. Rep., 2012.

[24] J.-F. Cardoso, "Blind signal separation: statistical principles," *Proceedings of the IEEE*, vol. 86, no. 10, pp. 2009–2025, 1998.

[25] S.-I. Amari, T.-P. Chen, and A. Cichocki, "Stability analysis of learning algorithms for blind source separation," *Neural Networks*, vol. 10, no. 8, pp. 1345–1351, 1997.

[26] M. Zibulevsky, "Blind source separation with relative newton method," in *Proc. ICA*, vol. 2003, 2003, pp. 897–902.

[27] J. A. Palmer, S. Makeig, K. Kreutz-Delgado, and B. D. Rao, "Newton method for the ICA mixture model," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2008, pp. 1805–1808.

[28] A. Delorme and S. Makeig, "Eeglab: an open source toolbox for analysis of single-trial eeg dynamics including independent component analysis," *Journal of neuroscience methods*, vol. 134, no. 1, pp. 9–21, 2004.

[29] J. Montoya-Martínez, J.-F. Cardoso, and A. Gramfort, "Caveats with stochastic gradient and maximum likelihood based ica for eeg," in *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2017, pp. 279–289.

[30] J.-F. Cardoso and B. H. Laheld, "Equivariant adaptive source separation," *IEEE Transactions on Signal Processing*, vol. 44, no. 12, pp. 3017–3030, 1996.

[31] D. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.

[32] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *arXiv preprint arXiv:1606.04838*, 2016.

[33] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, 1999.

[34] W. C. Davidon, "Variable metric method for minimization," *SIAM Journal on Optimization*, vol. 1, no. 1, pp. 1–17, 1991.

[35] R. Fletcher, "A new approach to variable metric algorithms," *The computer journal*, vol. 13, no. 3, pp. 317–322, 1970.

[36] R. Fletcher and M. J. Powell, "A rapidly convergent descent method for minimization," *The computer journal*, vol. 6, no. 2, pp. 163–168, 1963.

[37] C. G. Broyden, "The convergence of a class of double-rank minimization algorithms 1. general considerations," *IMA Journal of Applied Mathematics*, vol. 6, no. 1, pp. 76–90, 1970.

[38] D. Goldfarb, "A family of variable-metric methods derived by variational means," *Mathematics of computation*, vol. 24, no. 109, pp. 23–26, 1970.

[39] D. F. Shanno, "Conditioning of quasi-newton methods for function minimization," *Mathematics of computation*, vol. 24, no. 111, pp. 647–656, 1970.

[40] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[41] P. Wolfe, "Convergence conditions for ascent methods," *SIAM review*, vol. 11, no. 2, pp. 226–235, 1969.

[42] J. J. Moré and D. J. Thuente, "Line search algorithms with guaranteed sufficient decrease," *ACM Transactions on Mathematical Software (TOMS)*, vol. 20, no. 3, pp. 286–307, 1994.

[43] S. Van der Walt, S. Colbert, and G. Varoquaux, "The NumPy array: a structure for efficient numerical computation," *Comp. in Sci. & Eng.*, vol. 13, no. 2, pp. 22–30, 2011.

[44] T.-P. Jung, C. Humphries, T.-W. Lee, S. Makeig, M. J. McKeown, V. Iragui, and T. J. Sejnowsk, "Extended ica removes artifacts from electroencephalographic recordings," in *Proceedings of the 10th International Conference on Neural Information Processing Systems*, ser. NIPS'97. Cambridge, MA, USA: MIT Press, 1997, pp. 894–900.

[45] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International journal of computer vision*, vol. 42, no. 3, pp. 145–175, 2001.