Stochastic algorithms with descent guarantees for ICA

Pierre Ablin
INRIA
Université Paris-Saclay

Alexandre Gramfort
INRIA
Université Paris-Saclay

Jean-François Cardoso
CNRS
Institut d'Astrophysique de Paris

Francis Bach
INRIA
École Normale Supérieure

Abstract

Independent component analysis (ICA) is a widespread data exploration technique, where observed signals are modeled as linear mixtures of independent components. From a machine learning point of view, it amounts to a matrix factorization problem with a statistical independence criterion. Infomax is one of the most used ICA algorithms. It is based on a loss function which is a non-convex log-likelihood. We develop a new majorization-minimization framework adapted to this loss function. We derive an online algorithm for the streaming setting, and an incremental algorithm for the finite sum setting, with the following benefits. First, unlike most algorithms found in the literature, the proposed methods do not rely on any critical hyperparameter like a step size, nor do they require a line-search technique. Second, the algorithm for the finite sum setting, although stochastic, guarantees a decrease of the loss function at each iteration. Experiments demonstrate progress on the state-of-the-art for large scale datasets, without the necessity for any manual parameter tuning.

1 Introduction

Independent component analysis (ICA) (Comon, 1994) is an unsupervised data exploration technique. In its classical and most popular form, it models a random vector $\boldsymbol{x} \in \mathbb{R}^{p\times 1}$ as a linear mixture of independent sources. This means that there exists a source vector $\boldsymbol{s} \in \mathbb{R}^{p\times 1}$ of statistically independent features and a mixing matrix $A \in \mathbb{R}^{p\times p}$, such that $\boldsymbol{x} = A\boldsymbol{s}$. The aim of ICA is to recover A from some realizations of $\boldsymbol{x}$ without any assumption or constraint on A.

Despite being a linear and shallow model, ICA is widely used in many observational sciences. Indeed, many physical phenomena are well modeled by ICA. For example, in neuroscience, the physics driving the measurement process of electrical signals in the brain is linear following Maxwell's equations (Makeig et al., 1997). In astronomy (Morello et al., 2015), mechanics (Yang and Nagarajaiah, 2014), neuroscience (O'Muircheartaigh and Jbabdi, 2017), biology (Biton et al., 2014) and several other fields, ICA algorithms are used daily to process ever-increasing amounts of data[1]. In some data processing pipelines, ICA can be a computational bottleneck for large datasets, calling for more scalable algorithms. It is thus of importance to develop ICA solvers which are fast, easy to use and with strong convergence guarantees.

One of the first and most employed ICA algorithms is Infomax (Bell and Sejnowski, 1995). The Infomax objective function is equivalent to a likelihood criterion in which each feature of $\boldsymbol{s}$ follows a super-Gaussian distribution with density $d(\cdot)$ (roughly speaking, a super-Gaussian distribution is heavy-tailed; a rigorous definition is given in Section 2.1). The likelihood of $\boldsymbol{x}$ given A then is (Pham and Garat, 1997):

$$p(\boldsymbol{x}|A) = \frac{1}{|\det(A)|}\prod_{i=1}^{p} d([A^{-1}\boldsymbol{x}]_i). \tag{1}$$

It is more convenient to work with the unmixing matrix $W := A^{-1}$ and the negative log-likelihood, yielding a cost function $\ell(\boldsymbol{x}, W) := -\log(p(\boldsymbol{x}|W^{-1}))$:

$$\ell(\boldsymbol{x}, W) = -\log|W| - \sum_{i=1}^{p}\log(d([W\boldsymbol{x}]_i)) \ . \tag{2}$$

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

[1] Two of the most used ICA algorithms (Bell and Sejnowski, 1995; Hyvärinen, 1999a) have been cited over 1500 times in 2017 according to Google Scholar

The underlying expected risk is then:

$$\mathcal{L}(W) := \mathbb{E}_{\boldsymbol{x}}[\ell(\boldsymbol{x}, W)] \tag{3}$$
$$= -\log|W| - \sum_{i=1}^{p} \mathbb{E}[\log(d([W\boldsymbol{x}]_i))] \ .$$

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:

$$\mathcal{L}_n(W) := \frac{1}{n}\sum_{j=1}^{n} \ell(\boldsymbol{x}_j, W) \tag{4}$$
$$= -\log|W| - \frac{1}{n}\sum_{i=1}^{p}\sum_{j=1}^{n} \log(d([WX]_{ij}))] \ .$$

This article focuses 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 4. 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. More recently, Ablin et al. (2018b) proposed to use the L-BFGS algorithm with the Hessian approximations. 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 square matrix W, $|W|$ is the determinant of W. For a matrix M, $M_{i:}$ denotes its i-th row, and $M_{:j}$ denotes its j-th column. Given a function u from $\mathbb{R}$ to $\mathbb{R}$ and a matrix $Y \in \mathbb{R}^{p\times n}$, $u(Y)$ denotes the matrix of element-wise operations: $\forall i, j,\ \ u(Y)_{ij} = u(Y_{ij})$. For complexity analysis, we say that a quantity Q is $O(\phi(n, p))$ if $\frac{Q}{\phi(n,p)}$ is bounded.

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.

2.1 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:

$$G(y) := -\log(d(y)) = \min_{u\geq 0} \frac{uy^2}{2} + f(u), \tag{5}$$

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:

$$\tilde{\ell}(\boldsymbol{x}, W, \boldsymbol{u}) := -\log|W| + \sum_{i=1}^{p}[\frac{1}{2}u_i\,[W\boldsymbol{x}]_i^2 + f(u_i)], \quad (6)$$

and the associated empirical risk, for $U = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n] \in \mathbb{R}_+^{p\times n}$:

$$\tilde{\mathcal{L}}_n(W, U) := \frac{1}{n}\sum_{j=1}^{n}\tilde{\ell}(\boldsymbol{x}_j, W, \boldsymbol{u}_j) \quad (7)$$

$$= -\log|W| + \frac{1}{n}\sum_{i=1}^{p}\sum_{j=1}^{n}[\frac{1}{2}U_{ij}\,[WX]_{ij}^2 + f(U_{ij})]. \quad (8)$$

Following Eq. (5), we have:

Lemma 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 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. (5), 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. (5) 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, $U_{ij} = u^*([WX]_{ij})$. □

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 2.

2.2 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 supplementary material). 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:

$$\boldsymbol{x} = A\boldsymbol{s} + \boldsymbol{n}\ , \quad (9)$$

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 article 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.

2.3 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 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 1 and 2. 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.

3.1 Minimization step: Descent in W

Expanding $[WX]_{ij}^2$, the middle term in the new loss function (6) is quadratic in the rows of W:

$$\tilde{\mathcal{L}}_n = -\log|W| + \frac{1}{2}\sum_{i=1}^{p} W_{i:}A^i W_{i:}^\top + \frac{1}{n}\sum_{i=1}^{p}\sum_{j=1}^{n} f(U_{ij}), \tag{10}$$

where $W_{i:}$ denotes the i-th row of W, and the A^i's are $p \times p$ matrices given by:

$$A^i_{kl} := \frac{1}{n}\sum_{j=1}^{n} U_{ij}X_{kj}X_{lj} \ . \tag{11}$$

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 (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,

$$\arg\min_{\boldsymbol{m}\in\mathbb{R}^{1\times p}} \tilde{\mathcal{L}}_n(\Theta_i(\boldsymbol{m})W, U) = \frac{1}{\sqrt{(K^{-1})_{ii}}}(K^{-1})_{i:} \ . \tag{12}$$

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): denoting by $\mathcal{A}$ the "algorithm operator" which maps input signals X (be it a stream or a finite set) to the estimated mixing matrix, for any invertible matrix B, $\mathcal{A}(BX) = B\mathcal{A}(X)$.

3.2 Majorization step : Descent in U

For a fixed unmixing matrix W, Lemma 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), 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:

$$A^i \leftarrow A^i + \frac{1}{T}(U^{\text{new}}_{ij} - U^{\text{mem}}_{ij})\boldsymbol{x}_j\boldsymbol{x}_j^\top \ . \tag{13}$$

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:

$$A^i \leftarrow (1-\rho(n))A^i + \rho(n)U_{ij}\boldsymbol{x}_j\boldsymbol{x}_j^\top \ , \tag{14}$$

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.

3.3 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 $K_{i:}^{-1}$. 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 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.

3.4 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. (5), it is given by what we call gap : a positive quantity measuring the decrease 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:

$$\text{gap}(W, U^{\text{mem}}_{ij}) := \tilde{\mathcal{L}}_n(W, U^{\text{mem}}) - \tilde{\mathcal{L}}_n(W, \tilde{U}) \quad (15)$$

$$= \frac{1}{2} U^{\text{mem}}_{ij} \, [WX]^2_{ij} + f(U^{\text{mem}}_{ij}) - G([WX]_{ij}) \ . \quad (16)$$

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 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 (12) 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).

Algorithm 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, ..., t_max do
    Select a mini-batch b of size n_b at random
    for each index j ∈ b do                          // Majorization
        Select x = X_{:j}
        Compute u^new = u*(Wx)
        Compute the gaps (15)
        Find the q sources i_1, ..., i_q corresponding to the largest gaps
        Update A^i for i = i_1, ..., i_q using Eq. (13)
        Update the memory: U^mem_{:j} = u^new
    for i = 1, ..., p do                             // Minimization
        Update the i-th row of W using Eq. (12)
return W
```

Algorithm 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, ..., t_max do
    Fetch n_b samples from the stream
    for each fetched sample x do                     // Majorization
        Compute u = u*(Wx)
        Compute q indices i_1, ..., i_q at random
        Update A^i for i = i_1, ..., i_q using Eq. (14)
    for i = 1 ..., p do                              // Minimization
        Update the i-th row of W using Eq. (12)
return W
```

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`.

4.1 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 per-

formed. 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. (13) and Eq.(14)) 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) (Ablin et al., 2018b). 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, 1999a) 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, 1999b)). 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 (Ablin et al., 2018a)

4.2 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.

4.3 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.

4.4 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. 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. 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).

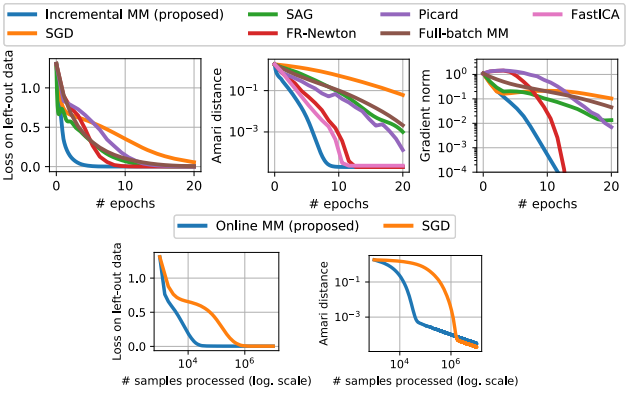

Figure 1: Results on synthetic data. Top: finite-sum problem. 100 datasets of size $n = 10^6$ and $p = 10$ are generated, each algorithm performs 20 epochs (passes on the dataset). Bottom: online problem. 100 datasets of size $n = 10^7$ and $p = 10$ are generated, each algorithm performs one pass on each dataset. Metrics are displayed with respect to epochs/number of passes.

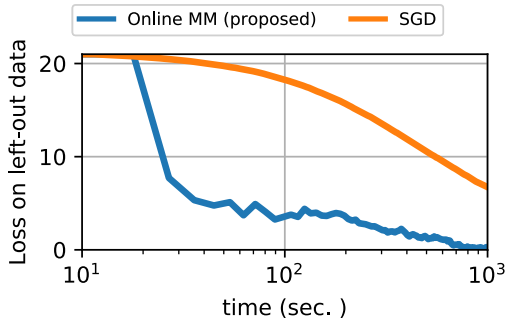

Figure 2: Online algorithms applied on a 32 GB real dataset with $p = 100$ and $n = 4 \times 10^7$. Time is in logarithmic scale. Values of the loss on left out data greater than its initial value are truncated.

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. 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

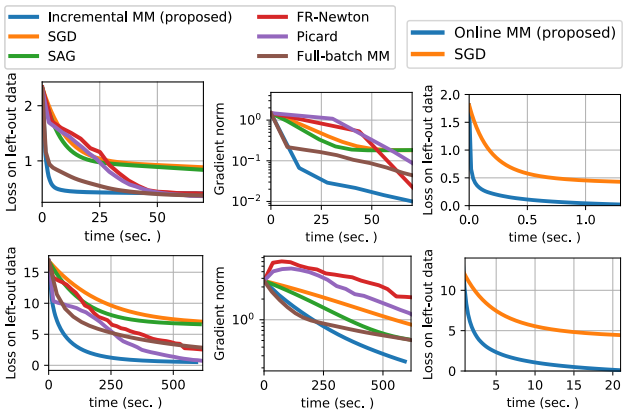

Figure 3: Behavior of different algorithms on real data. Top: 15 image patch datasets of size 10×10^6 are generated, and the averaged results are displayed. Bottom: same with 15 EEG datasets of size 71×10^6. Left and middle: finite sum problem. Right: online problem. Metrics are displayed with respect to time.

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.

4.5 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). 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. 4), we run the incremental algorithm with the greedy coordinate update rule discussed in Sec. 3.4 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

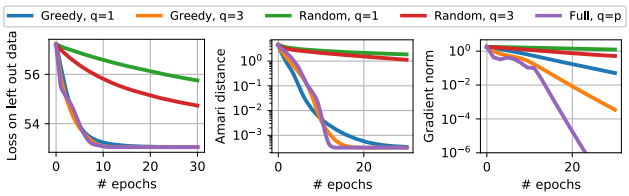

Figure 4: Effect of the greedy update rule, on a synthetic problem of size $p = 30$, $n = 10^5$. For a similar complexity, the greedy approach gives much faster convergence than the random approach.

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 ($q = p$).

5 Conclusion

In this article, 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.

Acknowledgments

We acknowledge support from the European Research Council (grants SEQUOIA 724063 and SLAB 676943).

References

Ablin, P., Cardoso, J.-F., and Gramfort, A. (2018a). Faster ica under orthogonal constraint. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4464–4468. IEEE.

Ablin, P., Cardoso, J.-F., and Gramfort, A. (2018b). Faster independent component analysis by preconditioning with hessian approximations. IEEE Transactions on Signal Processing, 66(15):4040–4049.

Amari, S.-I., Chen, T.-P., and Cichocki, A. (1997). Stability analysis of learning algorithms for blind source separation. Neural Networks, 10(8):1345–1351.

Bell, A. J. and Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. Neural Computation, 7(6):1129–1159.

Bermond, O. and Cardoso, J.-F. (1999). Approximate likelihood for noisy mixtures. In Proc. ICA, volume 99, pages 325–330.

Biton, A., Bernard-Pierrot, I., Lou, Y., Krucker, C., Chapeaublanc, E., Rubio-Pérez, C., López-Bigas, N., Kamoun, A., Neuzillet, Y., Gestraud, P., et al. (2014). Independent component analysis uncovers the landscape of the bladder tumor transcriptome and reveals insights into luminal and basal subtypes. Cell reports, 9(4):1235–1245.

Bottou, L. and Bousquet, O. (2008). The tradeoffs of large scale learning. In Advances in neural information processing systems, pages 161–168.

Bottou, L., Curtis, F. E., and Nocedal, J. (2016). Optimization methods for large-scale machine learning. Technical Report 1606.04838, arXiv.

Cappé, O. and Moulines, E. (2009). On-line expectation–maximization algorithm for latent data models. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 71(3):593–613.

Cardoso, J.-F. (1997). Infomax and maximum likelihood for blind source separation. IEEE Signal processing letters, 4(4):112–114.

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

Choi, H. and Choi, S. (2007). A relative trust-region algorithm for independent component analysis. Neurocomputing, 70(7):1502–1510.

Comon, P. (1994). Independent component analysis, a new concept? Signal processing, 36(3):287–314.

Defazio, A., Bach, F., and Lacoste-Julien, S. (2014). SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In Proc. NIPS, pages 1646–1654.

Delorme, A., Palmer, J., Onton, J., Oostenveld, R., and Makeig, S. (2012). Independent EEG sources are dipolar. PloS one, 7(2):e30135.

Girolami, M. (2001). A variational method for learning sparse and overcomplete representations. Neural Computation, 13(11):2517–2532.

Honorio, J., Samaras, D., Rish, I., and Cecchi, G. (2012). Variable selection for gaussian graphical models. In Artificial Intelligence and Statistics, pages 538–546.

Hoyer, P. O. and Hyvärinen, A. (2000). Independent component analysis applied to feature extraction from colour and stereo images. Network: Computation in Neural Systems, 11(3):191–210.

Hyvärinen, A. (1999a). Fast and robust fixed-point algorithms for independent component analysis. IEEE Transactions on Neural Networks, 10(3):626–634.

Hyvärinen, A. (1999b). The fixed-point algorithm and maximum likelihood estimation for independent component analysis. Neural Processing Letters, 10(1):1–5.

Johnson, R. and Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In Proc. NIPS, pages 315–323.

Lefevre, A., Bach, F., and Févotte, C. (2011). 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.

Mairal, J. (2015). Incremental majorization-minimization optimization with application to large-scale machine learning. SIAM Journal on Optimization, 25(2):829–855.

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2009). Online dictionary learning for sparse coding. In Proc. ICML, pages 689–696. ACM.

Makeig, S., Jung, T.-P., Bell, A. J., Ghahremani, D., and Sejnowski, T. J. (1997). Blind separation of auditory event-related brain responses into independent components. Proceedings of the National Academy of Sciences (PNAS), 94(20):10979–10984.

Montoya-Martínez, J., Cardoso, J.-F., and Gramfort, A. (2017). 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.

Moreau, E. and Macchi, O. (1998). Self-adaptive source separation. ii. comparison of the direct, feedback, and mixed linear network. IEEE Trans. on Signal Processing, 46(1):39–50.

Morello, G., Waldmann, I. P., Tinetti, G., Howarth, I. D., Micela, G., and Allard, F. (2015). Revisiting spitzer transit observations with independent component analysis: new results for the gj 436 system. The Astrophysical Journal, 802(2):117.

Neal, R. M. and Hinton, G. E. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In Learning in graphical models, pages 355–368. Springer.

O'Muircheartaigh, J. and Jbabdi, S. (2017). Concurrent white matter bundles and grey matter networks using independent component analysis. NeuroImage.

Palmer, J., Kreutz-Delgado, K., Rao, B. D., and Wipf, D. P. (2006). Variational EM algorithms for non-gaussian latent variable models. In Proc. NIPS, pages 1059–1066.

Pham, D. T. and Garat, P. (1997). Blind separation of mixture of independent sources through a quasi-maximum likelihood approach. IEEE Trans. on Signal Processing, 45(7):1712–1725.

Schmidt, M., Le Roux, N., and Bach, F. (2017). Minimizing finite sums with the stochastic average gradient. Mathematical Programming, 162(1-2):83–112.

Shewchuk, J. R. et al. (1994). An introduction to the conjugate gradient method without the agonizing pain.

Yang, Y. and Nagarajaiah, S. (2014). Blind identification of damage in time-varying systems using independent component analysis with wavelet transform. mechanical systems and signal processing, 47(1-2):3–20.

Zibulevsky, M. (2003). Blind source separation with relative newton method. In Proc. ICA, volume 2003, pages 897–902.