Application of Selected Projected Gradient Algorithms to Nonnegative Matrix Factorization

Rafal Zdunek[a] and Andrzej Cichocki[b*]

[a]Institute of Telecommunications, Teleinformatics, and Acoustics,
Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland

[b]Laboratory for Advanced Brain Signal Processing,
RIKEN Brain Science Institute, Wako-shi, Saitama 351-0198, Japan

Recently, a considerable growth of interest in Projected Gradient (PG) methods has been observed due to their high efficiency in solving large-scale minimization problems subject to linear constraints. Since the minimization problems underlying Nonnegative Matrix Factorization (NMF) of large matrices well matches this class of minimization problems, we test some recent PG methods in the context of their usefulness to NMF. In particular, the paper focuses on the following methods: projected Landweber, Barzilai-Borwein gradient projection, projected sequential subspace optimization (PSESOP), interior-point Newton (IPN), and sequential coordinate-wise. The methods are compared with respect to their performance in terms of Signal-to-Interference Ratio (SIR) and elapsed time, using the benchmark of mixed partially dependent nonnegative signals.

1. Introduction

Nonnegative Matrix Factorization (NMF) finds such nonnegative factors (matrices) $\boldsymbol{A} = [a_{ij}] \in \mathbb{R}^{I \times J}$ and $\boldsymbol{X} = [x_{jt}] \in \mathbb{R}^{J \times T}$ with $\forall i, j, t : a_{ij} \geq 0,\ x_{jt} \geq 0$ that $\boldsymbol{Y} \cong \boldsymbol{AX}$, given the observation matrix $\boldsymbol{Y} = [y_{it}] \in \mathbb{R}^{I \times T}$, the lower-rank J, and possibly other statistical information on the observed data or the factors to be estimated.

This method has found a variety of real-world applications in the areas such as blind separation of images and nonnegative signals [1–6], spectra recovering [7–10], pattern recognition and feature extraction [11–16], dimensionality reduction, segmentation and clustering [17–32], language modeling, text mining [25,33], music transcription [34], and neuro-biology (gene separation) [35,36].

Depending on an application, the estimated factors may have different interpretation. For example, Lee and Seung [11] introduced NMF as a method for decomposing an image (face) into parts-based representations (parts reminiscent of features such as lips, eyes, nose, etc.). In Blind Source Separation (BSS) [37], the matrix $\boldsymbol{Y}$ represents the observed mixed (superposed) signals or images, $\boldsymbol{A}$ is a mixing operator, and $\boldsymbol{X}$ is a matrix of true

*Also with Dept. of EE, Warsaw University of Technology; and IBS, Polish Academy of Science (PAN), Warsaw, Poland

source signals or images. Each row of $\boldsymbol{Y}$ or $\boldsymbol{X}$ is a signal or 1D image representation, where I is a number of observed mixed signals and J is a number of hidden (source) components. The index t usually denotes a sample (discrete time instant), where T is the number of available samples. In BSS, we usually have $T >> I \geq J$, and J is known or can be relatively easily estimated using SVD or PCA.

Our objective is to estimate the mixing matrix $\boldsymbol{A}$ and sources $\boldsymbol{X}$ subject to nonnegativity constraints of all the entries, given $\boldsymbol{Y}$ and possibly the knowledge on a statistical distribution of noisy disturbances.

The noise distribution is strongly application-dependent, however, in many BSS applications, a Gaussian noise is expected. Here our considerations are restricted to this case, however, the alternative NMF algorithms optimized to different distributions of the noise exist and can be found, e.g. in [38,37,39].

NMF was proposed by Paatero and Tapper [40,41] but Lee and Seung [42,11] highly popularized this method by using simple multiplicative algorithms to perform NMF. Unfortunately, the multiplicative algorithms are known to be very slowly-convergent for large-scale problems, and additionally they easily get stuck in local minima. Due to these reasons, there is a need to search more suitable algorithms for NMF. Many approaches have been proposed in the literature to relax these problems. One of them is to apply Projected Gradient (PG) algorithms [43–45] or projected Alternating Least-Squares (ALS) algorithms [33,46] instead of multiplicative ones. C.-J. Lin [45] suggested applying the Armijo rule to estimate the learning parameters in projected gradient updates for NMF. The gradient algorithms given in [47] also address the issue with selecting such a learning parameter that is the steepest descent and also keeps some distance to a boundary of the nonnegative orthant (subspace of real nonnegative numbers). Another very robust technique concerns exploiting the information from the second-order Taylor expansion term of a cost function to speed up the convergence. This approach was proposed in [39,48], where the mixing matrix $\boldsymbol{A}$ is updated with the projected Newton method, and the sources in $\boldsymbol{X}$ are computed with the projected least-squares method (the fixed point algorithm).

In this paper, we extend our approach to NMF that was initialized in [47]. Now we test more advanced and recent PG algorithms that are very efficient for solving large-scale minimization problems subject to linear constraints. In the next Section, we briefly discuss the PG approach to NMF. Section 3 describes the tested algorithms. The experimental results are illustrated in Section 4. Finally, some conclusions are given in Section 5.

2. Projected Gradient Algorithms

In contrast to the multiplicative algorithms, the class of PG algorithms has additive updates. The algorithms discussed here approximately solve Non-Negative Least Squares (NNLS) problems with the basic alternating minimization technique that is used in NMF:

$$\min_{\mathbf{x}_t \geq 0} D_F(\boldsymbol{y}_t || \boldsymbol{A}\boldsymbol{x}_t) = \frac{1}{2}\|\boldsymbol{y}_t - \boldsymbol{A}\boldsymbol{x}_t\|_2^2, \quad t = 1, \ldots, T \tag{1}$$

$$\min_{\underline{\mathbf{a}}_i \geq 0} D_F(\boldsymbol{y}_i || \boldsymbol{X}^T \underline{\boldsymbol{a}}_i) = \frac{1}{2}\|\boldsymbol{y}_i - \boldsymbol{X}^T \underline{\boldsymbol{a}}_i\|_2^2, \quad i = 1, \ldots, I \tag{2}$$

or in the equivalent matrix form

$$\min_{x_{jt}\geq 0} D_F(\boldsymbol{Y}||\boldsymbol{AX}) = \frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{AX}\|_F^2, \tag{3}$$

$$\min_{a_{ij}\geq 0} D_F(\boldsymbol{Y}^T||\boldsymbol{X}^T\boldsymbol{A}^T) = \frac{1}{2}\|\boldsymbol{Y}^T-\boldsymbol{X}^T\boldsymbol{A}^T\|_F^2, \tag{4}$$

where $\boldsymbol{A} = [\boldsymbol{a}_1,\ldots,\boldsymbol{a}_J] \in \mathbb{R}^{I\times J}$, $\boldsymbol{A}^T = [\underline{\boldsymbol{a}}_1,\ldots,\underline{\boldsymbol{a}}_I] \in \mathbb{R}^{J\times I}$, $\boldsymbol{X} = [\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T] \in \mathbb{R}^{J\times T}$, $\boldsymbol{X}^T = [\underline{\boldsymbol{x}}_1,\ldots,\underline{\boldsymbol{x}}_J] \in \mathbb{R}^{T\times J}$, $\boldsymbol{Y} = [\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T] \in \mathbb{R}^{I\times T}$, $\boldsymbol{Y}^T = [\underline{\boldsymbol{y}}_1,\ldots,\underline{\boldsymbol{y}}_I] \in \mathbb{R}^{I\times T}$, and usually $I \geq J$. The matrix $\boldsymbol{A}$ is assumed to be a full-rank matrix, so there exists a unique solution $\boldsymbol{X}^* \in \mathbb{R}^{J\times T}$ for any matrix $\boldsymbol{Y}$ since the NNLS problem is strictly convex (with respect to one set of variables $\{\ \boldsymbol{X}\}$).

The solution $\boldsymbol{x}_t^*$ to (1) satisfies the Karush-Kuhn-Tucker (KKT) conditions:

$$\boldsymbol{x}_t^* \geq 0, \quad \boldsymbol{g}_X(\boldsymbol{x}_t^*) \geq 0, \quad \boldsymbol{g}_X(\boldsymbol{x}_t^*)^T\boldsymbol{x}_t^* = 0, \tag{5}$$

or in the matrix notation:

$$\boldsymbol{X}^* \geq 0, \quad \boldsymbol{G}_X(\boldsymbol{X}^*) \geq 0, \quad \mathrm{tr}\{\boldsymbol{G}_X(\boldsymbol{X}^*)^T\boldsymbol{X}^*\} = 0, \tag{6}$$

where $\boldsymbol{g}_X$ and $\boldsymbol{G}_X$ are the corresponding gradient vector and gradient matrix:

$$\boldsymbol{g}_X(\boldsymbol{x}_t) = \nabla_{\mathbf{x}_t} D_F(\boldsymbol{y}_t||\boldsymbol{A}\boldsymbol{x}_t) = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x}_t - \boldsymbol{y}_t), \tag{7}$$

$$\boldsymbol{G}_X(\boldsymbol{X}) = \nabla_{\mathbf{X}} D_F(\boldsymbol{Y}||\boldsymbol{AX}) = \boldsymbol{A}^T(\boldsymbol{AX} - \boldsymbol{Y}). \tag{8}$$

Similarly, the KKT conditions for the solution $\underline{\boldsymbol{a}}^*$ to (2), and the solution $\boldsymbol{A}^*$ to (4) are as follows:

$$\underline{\boldsymbol{a}}_i^* \geq 0, \quad \boldsymbol{g}_A(\underline{\boldsymbol{a}}_i^*) \geq 0, \quad \boldsymbol{g}_A(\underline{\boldsymbol{a}}_i^*)^T\underline{\boldsymbol{a}}_i^* = 0, \tag{9}$$

or in the matrix notation:

$$\boldsymbol{A}^* \geq 0, \quad \boldsymbol{G}_A(\boldsymbol{A}^*) \geq 0, \quad \mathrm{tr}\{\boldsymbol{A}^*\boldsymbol{G}_A(\boldsymbol{A}^*)^T\} = 0, \tag{10}$$

where $\boldsymbol{g}_A$ and $\boldsymbol{G}_A$ are the gradient vector and gradient matrix of the objective function:

$$\boldsymbol{g}_A(\underline{\boldsymbol{a}}_t) = \nabla_{\underline{\mathbf{a}}_i} D_F(\underline{\boldsymbol{y}}_i||\boldsymbol{X}^T\underline{\boldsymbol{a}}_i) = (\boldsymbol{X}^T\underline{\boldsymbol{a}}_i - \underline{\boldsymbol{y}}_i), \tag{11}$$

$$\boldsymbol{G}_A(\boldsymbol{A}) = \nabla_{\mathbf{A}} D_F(\boldsymbol{Y}^T||\boldsymbol{X}^T\boldsymbol{A}^T) = (\boldsymbol{AX} - \boldsymbol{Y})\boldsymbol{X}^T. \tag{12}$$

There are many approaches to solve the problems (1) and (2), or equivalently (3) and (4). In this chapter, we discuss selected projected gradient methods that can be generally expressed by iterative updates:

$$\boldsymbol{X}^{(k+1)} = P_\Omega[\boldsymbol{X}^{(k)} - \eta_X^{(k)}\boldsymbol{P}_X^{(k)}], \tag{13}$$

$$\boldsymbol{A}^{(k+1)} = P_\Omega[\boldsymbol{A}^{(k)} - \eta_A^{(k)}\boldsymbol{P}_A^{(k)}], \tag{14}$$

where $P_\Omega[\xi]$ is a projection of ξ onto a convex feasible set $\Omega = \{\xi \in \mathbb{R} : \xi \geq 0\}$ – namely, the nonnegative orthant $\mathbb{R}_+$ (the subspace of nonnegative real numbers), $\boldsymbol{P}_X^{(k)}$ and $\boldsymbol{P}_A^{(k)}$ are descent directions for $\boldsymbol{X}$ and $\boldsymbol{A}$, and $\eta_X^{(k)}$ and $\eta_A^{(k)}$ are learning rules, respectively.

The projection $P_\Omega[\xi]$ can be performed in many ways. One of the simplest techniques is to replace all negative entries in ξ by zero-values, or in practical cases, by a small positive number ϵ to avoid numerical instabilities. Thus

$$P[\xi] = \max\{\epsilon, \xi\}. \tag{15}$$

However, this is not the only way to carry out the projection $P_\Omega(\xi)$. It is typically more efficient to choose the learning rates $\eta_X^{(k)}$ and $\eta_A^{(k)}$ so as to preserve nonnegativity of the solutions. The nonnegativity can be also maintained by solving least-squares problems subject to the constraints (6) and (10). Here we present the exemplary PG methods that work for NMF problems quite efficiently, and we implemented them in the Matlab toolbox: NMFLAB/NTFLAB for Signal and Image Processing [49]. For simplicity, we focus our considerations on updating the matrix $\boldsymbol{X}$, assuming that the updates for $\boldsymbol{A}$ are be obtained in a similar way.

3. Algorithms

3.1. Projected Landweber method

The Landweber method performs gradient descent minimization with the following iterative scheme:

$$\boldsymbol{X}^{(k+1)} = \boldsymbol{X}^{(k)} - \eta \boldsymbol{G}_X^{(k)}, \tag{16}$$

where the gradient is given by (8), and the learning rate $\eta \in (0, \eta_{max})$. The updating formula assures an asymptotical convergence to the minimal-norm least squares solution for the convergence radius defined by

$$\eta_{max} = \frac{2}{\lambda_{max}(\boldsymbol{A}^T\boldsymbol{A})}, \tag{17}$$

where $\lambda_{max}(\boldsymbol{A}^T\boldsymbol{A})$ is the maximal eigenvalue of $\boldsymbol{A}^T\boldsymbol{A}$. Since $\boldsymbol{A}$ is a nonnegative matrix, we have $\lambda_{max}(\boldsymbol{A}^T\boldsymbol{A}) \leq \max_j[\boldsymbol{A}^T\boldsymbol{A}\mathbf{1}_J]_j$, where $\mathbf{1}_J = [1, \ldots, 1]^T \in \mathbb{R}^J$. Thus the modified Landweber iterations can be expressed by the formula:

$$\boldsymbol{X}^{(k+1)} = \left[\boldsymbol{X}^{(k)} - \mathrm{diag}\{\eta_j\}\boldsymbol{G}_X^{(k)}\right]_+, \quad \text{where} \quad \eta_j = \frac{2}{(\boldsymbol{A}^T\boldsymbol{A}\mathbf{1}_J)_j}, \tag{18}$$

In the Projected Landweber (PL) method, which can be regarded as a particular case of the PG iterative formula (13)–(14), the solution obtained with (16) in each iterative step is projected onto the feasible set.

Finally, we have the PL-NMF algorithm:

Algorithm 1 *(***PL-NMF***)*

Set $\boldsymbol{A}$, $\boldsymbol{X}$, *% Initialization*
For $s = 1, 2, \ldots,$ *% Alternating*
Step 1: $\eta_j^{(X)} = \dfrac{2}{(\boldsymbol{A}^T \boldsymbol{A} \boldsymbol{1}_J)_j}$,
For $k = 1, 2 \ldots,$ *% Inner iterations for X*
$\boldsymbol{G}_X = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{X} - \boldsymbol{Y})$, *% Gradient with respect to X*
$\boldsymbol{X} \leftarrow [\boldsymbol{X} - \operatorname{diag}\{\eta_j\}\boldsymbol{G}_X]_+$, *% Updating*
End
Step 2: $\eta_j^{(A)} = \dfrac{2}{(\boldsymbol{X}\boldsymbol{X}^T \boldsymbol{1}_J)_j}$,
For $k = 1, 2 \ldots,$ *% Inner iterations for A*
$\boldsymbol{G}_A = (\boldsymbol{A}\boldsymbol{X} - \boldsymbol{Y})\boldsymbol{X}^T$, *% Gradient with respect to A*
$\boldsymbol{A} \leftarrow \left[\boldsymbol{A} - \boldsymbol{G}_A \operatorname{diag}\{\eta_j^{(A)}\}\right]_+$, *% Updating*
End
End *% Alternating*

3.2. Barzilai-Borwein gradient projection

The Barzilai-Borwein gradient projection method is motivated by the quasi-Newton approach, i.e. the inverse of the Hessian is replaced with an identity matrix $\boldsymbol{H}_k = \alpha_k \boldsymbol{I}$, where the scalar α_k is selected so that the inverse Hessian has similar behavior as the true Hessian in the recent iteration. Thus

$$\boldsymbol{X}^{(k+1)} - \boldsymbol{X}^{(k)} \approx \alpha_k \left(\nabla_X D(\boldsymbol{Y} || \boldsymbol{A}^{(k)} \boldsymbol{X}^{(k+1)}) - \nabla_X D(\boldsymbol{Y} || \boldsymbol{A}^{(k)} \boldsymbol{X}^{(k)}) \right).$$

For computation of $\boldsymbol{A}$, the assumption is similar. In comparison to, e.g. Lin's method [45], this method does not ensure that the objective function decreases at every iteration, but its general convergence has been proven analytically [50]. The general scheme of the Barzilai-Borwein gradient projection algorithm for updating $\boldsymbol{X}$ is as follows:

Algorithm 2 *(***GPSR-BB***)*

Set $\boldsymbol{A}$, $\boldsymbol{X}$, α_{min}, α_{max}, $\boldsymbol{\alpha}^{(0)} \in [\alpha_{min}, \alpha_{max}] \in \mathbb{R}^T$ *% Initialization*
For $k = 1, 2, \ldots,$ *% Inner iterations*
$\boldsymbol{\Delta}^{(k)} = P_\Omega[\boldsymbol{X}^{(k)} - \alpha^{(k)} \nabla_X D_F(\boldsymbol{Y} || \boldsymbol{A}\boldsymbol{X}^{(k)})] - \boldsymbol{X}^{(k)}$,
$\boldsymbol{\lambda}^{(k)} = \arg\min_{\lambda_t^{(k)} \in [0,1]} D_F(\boldsymbol{Y} || \boldsymbol{A}(\boldsymbol{X} + \boldsymbol{\Delta}^{(k)} \operatorname{diag}\{\boldsymbol{\lambda}\}))$, *where* $\boldsymbol{\lambda} = [\lambda_t] \in \mathbb{R}^T$,
$\boldsymbol{X}^{(k+1)} = \boldsymbol{X}^{(k)} + \boldsymbol{\Delta}^{(k)} \operatorname{diag}\{\boldsymbol{\lambda}\}$,
$\boldsymbol{\gamma}^{(k)} = \operatorname{diag}\{(\boldsymbol{\Delta}^{(k)})^T \boldsymbol{A}^T \boldsymbol{A} \boldsymbol{\Delta}^{(k)}\}$,
If $\gamma_t^{(k)} = 0$*:* $\alpha_t^{(k+1)} = \alpha_{max}$,
Else $\alpha_t^{(k+1)} = \min \left\{ \alpha_{max}, \max \left\{ \alpha_{min}, \dfrac{\left[(\boldsymbol{\Delta}^{(k)})^T \boldsymbol{\Delta}^{(k)}\right]_{tt}}{\gamma_t^{(k)}} \right\} \right\}$,
End
End *% Inner iterations*

Since $D_F(\boldsymbol{Y}||\boldsymbol{AX})$ is a quadratic function, the line search parameter $\boldsymbol{\lambda}^{(k)}$ can derived in the following closed-form formula:

$$\boldsymbol{\lambda}^{(k)} = \max\left\{0, \min\left\{1, \frac{\operatorname{diag}\left\{(\boldsymbol{\Delta}^{(k)})^T \nabla_X D_F(\boldsymbol{Y}||\boldsymbol{AX})\right\}}{\operatorname{diag}\left\{(\boldsymbol{\Delta}^{(k)})^T \boldsymbol{A}^T \boldsymbol{A}\boldsymbol{\Delta}^{(k)}\right\}}\right\}\right\} \tag{19}$$

The updates for $\boldsymbol{A}$ can be performed applying the above algorithm to the transposed system: $\boldsymbol{X}^T\boldsymbol{A}^T = \boldsymbol{Y}^T$, having $\boldsymbol{X}$ computed from the previous alternating step.

3.3. Projected sequential subspace optimization

The Projected SEquential Subspace OPtimization (PSESOP) method [51,52] carries out a projected minimization of a smooth objective function over a subspace spanned by several directions which include the current gradient and gradient from the previous iterations, and the Nemirovski directions. Nemirovski [53] suggested that convex smooth unconstrained optimization is optimal if the optimization is performed over a subspace which includes the current gradient $\boldsymbol{g}(\boldsymbol{x})$, the directions $\boldsymbol{d}_1^{(k)} = \boldsymbol{x}^{(k)} - \boldsymbol{x}^{(0)}$ and $\boldsymbol{d}_2^{(k)} = \sum_{n=0}^{k-1} w_n \boldsymbol{g}(\boldsymbol{x}_n)$ that should be orthogonal to the current gradient. Narkiss and Zibulevsky [51] also suggested to include another direction: $\boldsymbol{p}^{(k)} = \boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}$, which is motivated by a natural extension of the Conjugate Gradient (CG) method to a non-quadratic case. However, our practical observations showed that this direction does not have a strong impact on the NMF components, thus we neglected it in our NMF-PSESOP algorithm. Finally, we have the following algorithm for updating $\boldsymbol{x}_t$ which is a single column vector of $\boldsymbol{X}$:

Algorithm 3 *(***PSESOP***)*

Set $\boldsymbol{A}$, $\boldsymbol{x}_t^{(0)}$, p *% Initialization*
For $k = 1, 2, \ldots,$ *% Inner iterations*
$\boldsymbol{d}_1^{(k)} = \boldsymbol{x}^{(k)} - \boldsymbol{x}^{(0)}$,
$\boldsymbol{g}^{(k)} = \nabla_{\boldsymbol{x}_t} D_F(\boldsymbol{y}_t||\boldsymbol{A}\boldsymbol{x}_t)$,
$\boldsymbol{G}^{(p)} = \left[\boldsymbol{g}^{(k-1)}, \boldsymbol{g}^{(k-2)}, \ldots, \boldsymbol{g}^{(k-p)}\right] \in \mathbb{R}^{J\times p}$,
$w_k = \begin{cases} 1 & \text{if} \quad k = 1, \\ \frac{1}{2} + \sqrt{\frac{1}{4} + w_{k-1}^2} & \text{if} \quad k > 1, \end{cases}$
$\boldsymbol{w}^{(k)} = [w_k, w_{k-1}, \ldots, w_{k-p+1}]^T \in \mathbb{R}^p$,
$\boldsymbol{d}_2^{(k)} = \boldsymbol{G}^{(p)}\boldsymbol{w}^{(k)}$,
$\boldsymbol{D}^{(k)} = \left[\boldsymbol{d}_1^{(k)}, \boldsymbol{d}_2^{(k)}, \boldsymbol{g}^{(k)}, \boldsymbol{G}^{(p)}\right]$,
$\boldsymbol{\alpha}_*^{(k)} = \arg\min_\alpha D_F(\boldsymbol{y}_t||\boldsymbol{A}(\boldsymbol{x}_t^{(k)} + \boldsymbol{D}^{(k)}\boldsymbol{\alpha}^{(k)}))$,
$\boldsymbol{x}^{(k+1)} = P_\Omega[\boldsymbol{x}^{(k)} + \boldsymbol{D}^{(k)}\boldsymbol{\alpha}_*^{(k)}]$,
End *% Inner iterations*

The line search vector $\boldsymbol{\alpha}_*^{(k)}$ derived in a closed-form for the objective function $D_F(\boldsymbol{y}_t||\boldsymbol{A}\boldsymbol{x}_t)$ is as follows:

$$\boldsymbol{\alpha}_*^{(k)} = -((\boldsymbol{D}^{(k)})^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{D}^{(k)} + \lambda\boldsymbol{I})^{-1}(\boldsymbol{D}^{(k)})^T\nabla_{\boldsymbol{x}_t} D_F(\boldsymbol{y}_t||\boldsymbol{A}\boldsymbol{x}_t), \tag{20}$$

The regularization parameter can be set as a very small constant to avoid instabilities in inverting a rank-deficient matrix in case $\boldsymbol{D}^{(k)}$ has zero-value or dependent columns.

The algorithm for updating the row vectors of $\boldsymbol{A}$ is similar.

3.4. Interior point Newton algorithm

The Interior Point Newton (IPN) algorithm [54] solves the NNLS problem (1) by searching the solution satisfying the KKT conditions (5) which equivalently can be expressed by the nonlinear equations:

$$\boldsymbol{D}(\boldsymbol{x}_t)\boldsymbol{g}(\boldsymbol{x}_t) = 0, \tag{21}$$

where $\boldsymbol{D}(\boldsymbol{x}_t) = \text{diag}\{d_1(\boldsymbol{x}_t), \ldots, d_J(\boldsymbol{x}_t)\}$, $\boldsymbol{x}_t \geq 0$, and

$$d_j(\boldsymbol{x}_t) = \begin{cases} x_{jt} & \text{if} \quad g_j(\boldsymbol{x}_t) \geq 0, \\ 1 & \text{otherwise} \end{cases}$$

Applying the Newton method to (21), we have in the k-th iterative step:

$$(\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{A}^T\boldsymbol{A} + \boldsymbol{E}_k(\boldsymbol{x}_t))\boldsymbol{p}_k = -\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{g}_k(\boldsymbol{x}_t), \tag{22}$$

where

$$\boldsymbol{E}_k(\boldsymbol{x}_t) = \text{diag}\{e_1(\boldsymbol{x}_t), \ldots, e_J(\boldsymbol{x}_t)\}. \tag{23}$$

In [54], the entries of the matrix $\boldsymbol{E}_k(\boldsymbol{x}_t)$ are defined by

$$e_j(\boldsymbol{x}_t) = \begin{cases} g_j(\boldsymbol{x}_t) & \text{if} \quad 0 \leq g_j(\boldsymbol{x}_t) < x_{jt}^{\gamma}, \quad \text{or} \quad (g_j(\boldsymbol{x}_t))^{\gamma} > x_{jt}, \\ 1 & \text{otherwise} \end{cases} \tag{24}$$

for $1 < \gamma \leq 2$.

If the solution is degenerate, i.e. $t = 1, \ldots, T$, $\exists j : x_{jt}^* = 0$, and $g_{jt} = 0$, the matrix $\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{A}^T\boldsymbol{A} + \boldsymbol{E}_k(\boldsymbol{x}_t)$ may be singular. To avoid such a case, the system of equations has been re-scaled to the following form:

$$\boldsymbol{W}_k(\boldsymbol{x}_t)\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{M}_k(\boldsymbol{x}_t)\boldsymbol{p}_k = -\boldsymbol{W}_k(\boldsymbol{x}_t)\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{g}_k(\boldsymbol{x}_t), \tag{25}$$

with

$$\boldsymbol{M}_k(\boldsymbol{x}_t) = \boldsymbol{A}^T\boldsymbol{A} + \boldsymbol{D}_k(\boldsymbol{x}_t)^{-1}\boldsymbol{E}_k(\boldsymbol{x}_t), \tag{26}$$

$$\boldsymbol{W}_k(\boldsymbol{x}_t) = \text{diag}\{w_1(\boldsymbol{x}_t), \ldots, w_J(\boldsymbol{x}_t)\}, \quad w_j(\boldsymbol{x}_t) = (d_j(\boldsymbol{x}_t) + e_j(\boldsymbol{x}_t))^{-1}, \tag{27}$$

for $\boldsymbol{x}_t > 0$. In [54], the system (25) is solved by the inexact Newton method, which leads to the following updates:

$$\boldsymbol{W}_k(\boldsymbol{x}_t)\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{M}_k(\boldsymbol{x}_t)\boldsymbol{p}_k = -\boldsymbol{W}_k(\boldsymbol{x}_t)\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{g}_k(\boldsymbol{x}_t) + \boldsymbol{r}_k(\boldsymbol{x}_t), \tag{28}$$

$$\hat{\boldsymbol{p}}_k = \max\{\sigma, 1 - ||P_{\Omega}[\boldsymbol{x}_t^{(k)} + \boldsymbol{p}_k] - \boldsymbol{x}_t^{(k)}||_2\} \left(P_{\Omega}[\boldsymbol{x}_t^{(k)} + \boldsymbol{p}_k] - \boldsymbol{x}_t^{(k)}\right), \tag{29}$$

$$\boldsymbol{x}_t^{(k+1)} = \boldsymbol{x}_t^{(k)} + \hat{\boldsymbol{p}}_k, \tag{30}$$

where $\sigma \in (0,1)$, $\boldsymbol{r}_k(\boldsymbol{x}_t) = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x}_t - \boldsymbol{y}_t)$, and $P_\Omega[\cdot]$ is a projection onto a feasible set Ω.

The transformation of the normal matrix $\boldsymbol{A}^T\boldsymbol{A}$ by the matrix $\boldsymbol{W}_k(\boldsymbol{x}_t)\boldsymbol{D}_k(\boldsymbol{x}_t)$ in (25) makes the system matrix $\boldsymbol{W}_k(\boldsymbol{x}_t)\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{M}_k(\boldsymbol{x}_t)$ is no longer symmetric and positive-definite. There are many methods for handling such systems of linear equations, like QMR [55], BiCG [56,57], BiCGSTAB [58], or GMRES-like methods [59], however, they are more complicated and computationally demanding than, e.g. the basic CG algorithm [60]. To apply the CG algorithm the system matrix in (25) must be converted to a positive-definite symmetric matrix, which can be easily done with normal equations. The methods like CGLS [61] or LSQR [62] are therefore suitable for such tasks. The transformed system has the form:

$$\boldsymbol{Z}_k(\boldsymbol{x}_t)\tilde{\boldsymbol{p}}_k = -\boldsymbol{S}_k(\boldsymbol{x}_t)\boldsymbol{g}_k(\boldsymbol{x}_t) + \tilde{\boldsymbol{r}}_k(\boldsymbol{x}_t), \tag{31}$$

$$\boldsymbol{S}_k(\boldsymbol{x}_t) = \sqrt{\boldsymbol{W}_k(\boldsymbol{x}_t)\boldsymbol{D}_k(\boldsymbol{x}_t)}, \tag{32}$$

$$\boldsymbol{Z}_k(\boldsymbol{x}_t) = \boldsymbol{S}_k(\boldsymbol{x}_t)\boldsymbol{M}_k(\boldsymbol{x}_t)\boldsymbol{S}_k(\boldsymbol{x}_t) = \boldsymbol{S}_k(\boldsymbol{x}_t)\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{S}_k(\boldsymbol{x}_t) + \boldsymbol{W}_k(\boldsymbol{x}_t)\boldsymbol{E}_k(\boldsymbol{x}_t), \tag{33}$$

with $\tilde{\boldsymbol{p}}_k = \boldsymbol{S}_k^{-1}(\boldsymbol{x}_t)\boldsymbol{p}_k$ and $\tilde{\boldsymbol{r}}_k = \boldsymbol{S}_k^{-1}(\boldsymbol{x}_t)\boldsymbol{r}_k(\boldsymbol{x}_t)$.

Since our cost function is quadratic, its minimization in a single step is performed with combining the projected Newton step with the constrained scaled Cauchy step that is given in the form:

$$\boldsymbol{p}_k^{(C)} = -\tau_k\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{g}_k(\boldsymbol{x}_t), \quad \tau_k > 0 \tag{34}$$

Assuming $\boldsymbol{x}_t^{(k)} + \boldsymbol{p}_k^{(C)} > 0$, τ_k is chosen as being either the unconstrained minimizer of the quadratic function $\psi_k(-\tau_k\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{g}_k(\boldsymbol{x}_t))$ or a scalar proportional to the distance to the boundary along $-\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{g}_k(\boldsymbol{x}_t)$, where

$$\begin{aligned}\psi_k(\boldsymbol{p}) &= \frac{1}{2}\boldsymbol{p}^T\boldsymbol{M}_k(\boldsymbol{x}_t)\boldsymbol{p} + \boldsymbol{p}^T\boldsymbol{g}_k(\boldsymbol{x}_t) \\ &= \frac{1}{2}\boldsymbol{p}^T(\boldsymbol{A}^T\boldsymbol{A} + \boldsymbol{D}_k^{-1}(\boldsymbol{x}_t)\boldsymbol{E}_k(\boldsymbol{x}_t))\boldsymbol{p} + \boldsymbol{p}^T\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x}_t^{(k)} - \boldsymbol{y}_t).\end{aligned} \tag{35}$$

Thus

$$\tau_k = \begin{cases} \tau_1 = \arg\min_\tau \psi_k(-\tau_k\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{g}_k(\boldsymbol{x}_t)) \;\text{ if }\; \boldsymbol{x}_t^{(k)} - \tau_1\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{g}_k(\boldsymbol{x}_t) > 0, \\ \tau_2 = \theta \min_j \left\{ \dfrac{x_{jt}^{(k)}}{\left(\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{g}_k(\boldsymbol{x}_t)\right)_j} : \left(\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{g}_k(\boldsymbol{x}_t)\right)_j > 0 \right\} \;\text{ otherwise} \end{cases} \tag{36}$$

where $\psi_k(-\tau_k\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{g}_k(\boldsymbol{x}_t)) = \dfrac{\left(\boldsymbol{g}_k(\boldsymbol{x}_t)\right)^T\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{g}_k(\boldsymbol{x}_t)}{\left(\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{g}_k(\boldsymbol{x}_t)\right)^T\boldsymbol{M}_k(\boldsymbol{x}_t)\boldsymbol{D}_k(\boldsymbol{x}_t)\boldsymbol{g}_k(\boldsymbol{x}_t)}$ with $\theta \in (0,1)$. For $\psi_k(\boldsymbol{p}_k^{(C)}) < 0$, the global convergence is achieved if $\mathrm{red}(\boldsymbol{x}_t^{(k+1)} - \boldsymbol{x}_t^{(k)}) \geq 0$, $\beta \in (0,1)$, with

$$\mathrm{red}(\boldsymbol{p}) = \frac{\psi_k(\boldsymbol{p})}{\boldsymbol{p}_k^{(C)}}. \tag{37}$$

The usage of the constrained scaled Cauchy step leads to the following updates:

$$\boldsymbol{s}_t^{(k)} = t(\boldsymbol{p}_k^{(C)} - \hat{\boldsymbol{p}}_k) + \hat{\boldsymbol{p}}_k, \tag{38}$$

$$\boldsymbol{x}_t^{(k+1)} = \boldsymbol{x}_t^{(k)} + \boldsymbol{s}_t^{(k)}, \tag{39}$$

with $t \in [0,1)$, $\hat{\boldsymbol{p}}_k$ and $\boldsymbol{p}_k^{(C)}$ given by (29) and (34), respectively, and t being the smaller square root (laying in $(0,1)$) of the quadratic equation:

$$\pi(t) = \psi_k\left(t(\boldsymbol{p}_k^{(C)} - \hat{\boldsymbol{p}}_k) + \hat{\boldsymbol{p}}_k\right) - \beta\psi_k(\boldsymbol{p}_k^{(C)}) = 0. \tag{40}$$

3.5. Sequential coordinate-wise algorithm

The NNLS problem (1) can be expressed in terms of the following Quadratic Problem (QP):

$$\min_{\boldsymbol{x}_t \geq 0} \Psi(\boldsymbol{x}_t), \quad t = 1, \ldots, T \tag{41}$$

where

$$\Psi(\boldsymbol{x}_t) = \frac{1}{2}\boldsymbol{x}_t^T \boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{c}_t^T \boldsymbol{x}_t, \tag{42}$$

with $\boldsymbol{H} = \boldsymbol{A}^T\boldsymbol{A}$ and $\boldsymbol{c}_t = -\boldsymbol{A}^T\boldsymbol{y}_t$. The function $\Psi(\boldsymbol{x}_t)$ can be equivalently rewritten as:

$$\begin{aligned}
\Psi(\mathbf{x_t}) &= \frac{1}{2}\sum_{p\in\mathcal{I}}\sum_{r\in\mathcal{I}} x_{pt}x_{rt}(\boldsymbol{A}^T\boldsymbol{A})_{pr} + \sum_{p\in\mathcal{I}} x_{pt}(\boldsymbol{A}^T\boldsymbol{y}_t)_{pt} \\
&= \frac{1}{2}x_{jt}^2(\boldsymbol{A}^T\boldsymbol{A})_{jj} + x_{jt}(\boldsymbol{A}^T\boldsymbol{y}_t)_{jt} + x_{jt}\sum_{p\in\mathcal{I}\setminus\{j\}} x_{pt}(\boldsymbol{A}^T\boldsymbol{A})_{pj} + \sum_{p\in\mathcal{I}\setminus\{j\}} x_{pt}(\boldsymbol{A}^T\boldsymbol{y}_t)_{pt} \\
&+ \frac{1}{2}\sum_{p\in\mathcal{I}\setminus\{j\}}\sum_{r\in\mathcal{I}\setminus\{j\}} x_{pt}x_{rt}(\boldsymbol{A}^T\boldsymbol{A})_{pr} = \frac{1}{2}x_{jt}^2 h_{jj} + x_{jt}\beta_{jt} + \gamma_{jt},
\end{aligned} \tag{43}$$

where $\mathcal{I} = \{1, \ldots, J\}$, and

$$h_{jj} = (\boldsymbol{A}^T\boldsymbol{A})_{jj}, \tag{44}$$

$$\beta_{jt} = (\boldsymbol{A}^T\boldsymbol{y}_t)_{jt} + \sum_{p\in\mathcal{I}\setminus\{j\}} x_{pt}(\boldsymbol{A}^T\boldsymbol{A})_{pj} = [\boldsymbol{A}^\mathbf{T}\boldsymbol{A}\boldsymbol{x_t} + \boldsymbol{A}^\mathbf{T}\boldsymbol{y_t}]_{jt} - (\boldsymbol{A}^T\boldsymbol{A})_{jj}x_{jt}, \tag{45}$$

$$\gamma_{jt} = \sum_{p\in\mathcal{I}\setminus\{j\}} x_{pt}(\boldsymbol{A}^T\boldsymbol{y}_t)_{pt} + \frac{1}{2}\sum_{p\in\mathcal{I}\setminus\{j\}}\sum_{r\in\mathcal{I}\setminus\{j\}} x_{pt}x_{rt}(\boldsymbol{A}^T\boldsymbol{A})_{pr}. \tag{46}$$

$$\tag{47}$$

Considering the optimization of $\Psi(\mathbf{x_t})$ with respect to the selected variable x_{jt}, the following analytical solution can be derived:

$$\begin{aligned}
x_{jt}^* &= \arg\min \Psi\left([x_{1t}, \ldots, x_{jt}, \ldots, x_{Jt}]^T\right) \\
&= \arg\min \frac{1}{2}x_{jt}^2 h_{jj} + x_{jt}\beta_{jt} + \gamma_{jt} \\
&= \max\left(0, -\frac{\beta_{jt}}{h_{jj}}\right) \\
&= \max\left(0, x_{jt} - \frac{[\boldsymbol{A}^\mathbf{T}\boldsymbol{A}\boldsymbol{x_t}]_{jt} + [\boldsymbol{A}^T\boldsymbol{y}_t]_{jt}}{(\boldsymbol{A}^T\boldsymbol{A})_{jj}}\right).
\end{aligned} \tag{48}$$

The Sequential Coordinate-Wise Algorithm (SCWA) [63] updates only single variable x_{jt} in one iterative step. Thus

$$x_{pt}^{(k+1)} = x_{pt}^{(k)}, \quad \forall p \in \mathcal{I}\backslash\{j\} \quad \text{and} \quad x_{jt}^{(k+1)} \neq x_{jt}^{(k)}. \tag{49}$$

Any optimal solution to the QP (41) satisfies the KKT conditions given by (5) and the stationarity condition of the following Lagrange function:

$$\mathcal{L}(\boldsymbol{x}_t, \boldsymbol{\lambda}_t) = \frac{1}{2}\boldsymbol{x}_t^T \boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{c}_t^T \boldsymbol{x}_t - \boldsymbol{\lambda}_t^T \boldsymbol{x}_t, \tag{50}$$

where $\boldsymbol{\lambda}_t \in \mathbb{R}^J$ is a vector of Lagrange multipliers (or dual variables) corresponding to the vector $\boldsymbol{x}_t$. Thus $\nabla_{\boldsymbol{x}_t}\mathcal{L}(\boldsymbol{x}_t, \boldsymbol{\lambda}_t) = \boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{c}_t - \boldsymbol{\lambda}_t = \boldsymbol{0}$. In the SCWA, the Lagrange multipliers are updated in each iteration according to the formula:

$$\boldsymbol{\lambda}_t{}^{(k+1)} = \boldsymbol{\lambda}_t{}^{(k)} + \left(x_{jt}^{(k+1)} - x_{jt}^{(k)}\right)\boldsymbol{h}_j, \tag{51}$$

where $\boldsymbol{h}_j$ is the j-th column of $\boldsymbol{H}$, and $\boldsymbol{\lambda}_t^{(0)} = \boldsymbol{c}_t$.

Finally, the SCWA has the following updates:

$$x_{jt}^{(k+1)} = \max\left(0, x_{jt}^{(k)} - \frac{\lambda_j^{(k)}}{(\boldsymbol{A}^T\boldsymbol{A})_{jj}}\right) \quad \text{and} \quad x_{pt}^{(k+1)} = x_{pt}^{(k)}, \quad \forall p \in \mathcal{I}\backslash\{j\} \tag{52}$$

$$\boldsymbol{\lambda}_t{}^{(k+1)} = \boldsymbol{\lambda}_t{}^{(k)} + \left(x_{jt}^{(k+1)} - x_{jt}^{(k)}\right)\boldsymbol{h}_j, \tag{53}$$

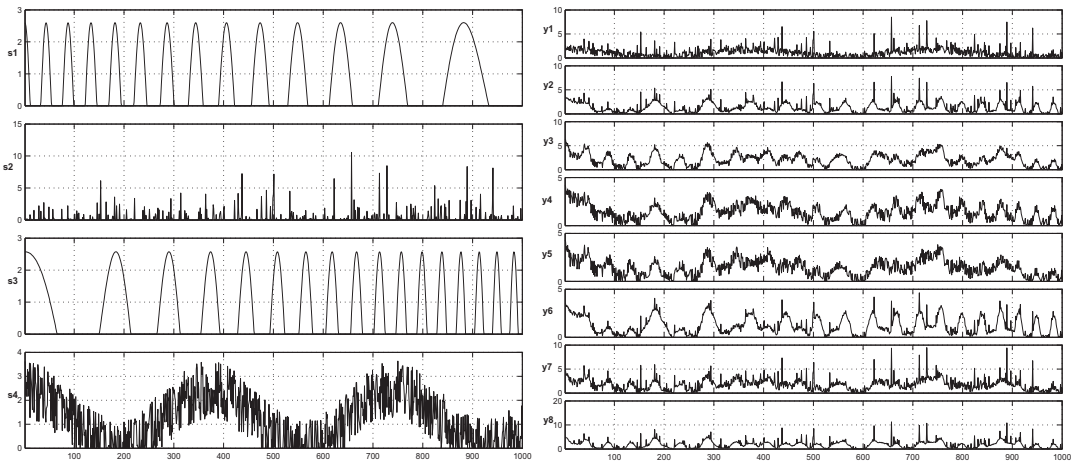
4. Simulations

The analyzed algorithms are evaluated with numerical tests related to typical BSS problems. We used the synthetic benchmark of 4 partially dependent nonnegative signals that are illustrated in Fig. 1 (a). The signals are mixed by random, uniformly distributed nonnegative matrix $\boldsymbol{A} \in \mathbb{R}^{8\times 4}$ with cond$\{\boldsymbol{A}\} = 4.11$. The matrix $\boldsymbol{A}$ is displayed in (54):

$$\boldsymbol{A} = \begin{bmatrix} 0.0631 & 0.7666 & 0.0174 & 0.6596 \\ 0.2642 & 0.6661 & 0.8194 & 0.2141 \\ 0.9995 & 0.1309 & 0.6211 & 0.6021 \\ 0.2120 & 0.0954 & 0.5602 & 0.6049 \\ 0.4984 & 0.0149 & 0.2440 & 0.6595 \\ 0.2905 & 0.2882 & 0.8220 & 0.1834 \\ 0.6728 & 0.8167 & 0.2632 & 0.6365 \\ 0.9580 & 0.9855 & 0.7536 & 0.1703 \end{bmatrix} \tag{54}$$

The mixing signals are shown in Fig. 1(b).

Because the number of variables in $\boldsymbol{X}$ is much greater than in $\boldsymbol{A}$, i.e. $I \times J = 32$ and $J \times T = 4000$, we test the projected gradient algorithms only for updating $\boldsymbol{A}$. The



(a) (b)

Figure 1. Dataset: (a) original 4 source signals, (b) observed 8 mixed signals.

variables in $\boldsymbol{X}$ are updated with the standard projected Fixed Point Alternating Least Squares (FP-ALS) algorithm that is extensively analyzed in [64].

In general, the FP-ALS algorithm solves the least-squares problem

$$\boldsymbol{X}^* = \arg\min_{\mathbf{X}} \left\{ \frac{1}{2} ||\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}||_F^2 \right\} \tag{55}$$

with the Moore-Penrose pseudo-inverse of a system matrix, i.e. in our case, the matrix $\boldsymbol{A}$. Since in NMF usually $I \geq J$, we formulate normal equations as $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X} = \boldsymbol{A}^T\boldsymbol{Y}$, and the least-squares solution of minimal l_2-norm to the normal equations is $\boldsymbol{X}_{LS} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{Y} = \boldsymbol{A}^+\boldsymbol{Y}$, where $\boldsymbol{A}^+$ is the Moore-Penrose pseudo-inverse of $\boldsymbol{A}$. The projected FP-ALS algorithm is obtained with a simple "half-rectified" projection, i.e.

$$\boldsymbol{X} = \left[\boldsymbol{A}^+\boldsymbol{Y}\right]_+, \tag{56}$$

where $[\xi]_+ = \max\{\epsilon, \xi\}$ with small constant $\epsilon \approx 10^{-16}$ to avoid numerical instabilities.

The alternating minimization is non-convex in spite of the cost function being convex with respect to one set of variables. Thus, many NMF algorithms may get stuck in local minima, and hence, initialization plays a predominate role. In the tests, we apply the multi-start initialization described in [47] with the following parameters: $N = 10$ (number of restarts), $K_i = 30$ (number of initial alternating steps), and $K_f = 1000$ (number of final alternating steps). Each initial sample of $\boldsymbol{A}$ and $\boldsymbol{X}$ has been randomly generated from a uniform distribution. [Each algorithm has been tested for two cases of inner iterations, i.e. with $k = 1$ and $k = 5$. The inner iterations means a number of iterative steps that are performed to update only $\boldsymbol{A}$ (before going to the update of $\boldsymbol{X}$). Additionally, the multilayer technique [46,47] with 3 layers ($L = 3$) is applied.

The algorithms have been evaluated with the Signal-to-Interference Ratio (SIR) measures, calculated separately for each source signal and each column in the mixing matrix.

Let $\boldsymbol{x}_j$ and $\hat{\boldsymbol{x}}_j$ be the j-th source and estimated signal, respectively. Analogically, let $\boldsymbol{a}_j$ and $\hat{\boldsymbol{a}}_j$ be the j-th column of the true and estimated mixing matrix, respectively. Thus the SIRs for the sources are given by:

$$SIR_j^{(X)} = 20 \log \left\{ \frac{||\hat{\boldsymbol{x}}_j - \boldsymbol{x}_j||_2}{||\boldsymbol{x}_j||_2} \right\}, \quad j = 1, \ldots, J, \quad \text{[dB]} \tag{57}$$

and similarly for each column in $\boldsymbol{A}$ we have:

$$SIR_j^{(A)} = 20 \log \left\{ \frac{||\hat{\boldsymbol{a}}_j - \boldsymbol{a}_j||_2}{||\boldsymbol{a}_j||_2} \right\}, \quad j = 1, \ldots, J, \quad \text{[dB]}. \tag{58}$$

We test the algorithms with the Monte Carlo (MC) analysis, running each algorithm 100 times. Each run has been initialized with the multi-start procedure. The algorithms have been evaluated with the mean-SIR values that are calculated as follows:

$$\overline{SIR}_X = \frac{1}{J} \sum_{j=1}^{J} SIR_j^{(X)}, \tag{59}$$

$$\overline{SIR}_A = \frac{1}{J} \sum_{j=1}^{J} SIR_j^{(A)}, \tag{60}$$

for each MC sample. The mean-SIRs for the worst (with the lowest mean-SIR values) and best (with the highest mean-SIR values) samples are given in Table 1. The number k means the number of inner iterations for updating $\boldsymbol{A}$, and L denotes the number of layers in the multilayer technique [46,47]. The notation $L = 1$ means that the multilayer technique is not used. The elapsed time [in seconds] is measured in Matlab, and it informs us about a degree of complexity of the algorithm.

For comparison, Table 1 contains also the results obtained for the standard multiplicative NMF algorithm (denoted as M-NMF) that minimizes the squared Euclidean distance. Additionally, the results of testing the PG algorithms which were proposed in [47] have been also included. The acronyms Lin-PG, IPG, RMRNSD refer to the following algorithms: Projected Gradient proposed by Lin [45]), Interior-Point Gradient, and Regularized Minimal Residual Norm Steepest Descent (the algorithm proposed by Nagy and Strakos [65]). The algorithms have been tested in [47] in the context of their usefulness to BSS with NMF.

5. Conclusions

The performance of the analyzed algorithms can be inferred from the results given in Table 1. In particular, the results show how the algorithms are sensitive to initialization, or in other words, how easily they fall in local minima. Also the complexity of the algorithms can be estimated from the information on the elapsed time that is measured in Matlab.

It is easy to notice that the PSESOP algorithm gives the best estimation (the sample which has the highest mean-SIR value), however, it does not give the best estimation

Table 1
Mean-SIRs [dB] obtained with 100 samples of Monte Carlo analysis for estimation of sources and columns of mixing matrix from noise-free mixtures of signals in Fig. 1. Sources $\boldsymbol{X}$ are estimated with the projected pseudo-inverse. The number of inner iterations for updating $\boldsymbol{A}$ is denoted by k, and the number of layers (in the multilayer technique) by L. The notation *best* or *worst* in parenthesis that follows the algorithm's name means the mean-SIR value is calculated for the best or worst sample from Monte Carlo analysis, respectively. In the last column, the elapsed time [in seconds] is given for each algorithm with $k = 1$ and $L = 1$.

Algorithm:	Mean-SIR_A [dB]				Mean-SIR_X [dB]				Time
	$L = 1$		$L = 3$		$L = 1$		$L = 3$		
	$k = 1$	$k = 5$	$k = 1$	$k = 5$	$k = 1$	$k = 5$	$k = 1$	$k = 5$	
M-NMF (best)	21	22.1	42.6	37.3	26.6	27.3	44.7	40.7	1.9
M-NMF (worst)	5.5	5.7	5.3	6.3	5.8	6.5	5	5.5	
PL(best)	22.9	25.3	46.5	42	23.9	23.5	55.8	51	1.9
PL(worst)	4.8	4.8	4.8	5.0	4.6	4.6	4.6	4.8	
Lin-PG(best)	36.3	23.6	78.6	103.7	34.2	33.3	78.5	92.8	8.8
Lin-PG(worst)	14.4	13.1	17.5	40.1	13.9	13.8	18.1	34.4	
GPSR-BB(best)	18.2	22.7	7.3	113.8	22.8	54.3	9.4	108.1	2.4
GPSR-BB(worst)	7.4	17.3	7.3	24.9	4.6	14.7	2	23	
PSESOP(best)	21.2	22.6	71.1	132.2	23.4	55.5	56.5	137.2	5.4
PSESOP(worst)	8.3	15.8	6.9	28.7	8.2	16.6	7.2	30.9	
IPG(best)	20.6	22.2	52.1	84.3	35.7	28.6	54.2	81.4	2.7
IPG(worst)	10.5	13.4	9.4	21.2	10.2	13.5	8.9	15.5	
IPN(best)	20.8	22.6	59.9	65.8	53.5	52.4	68.6	67.2	14.2
IPN(worst)	11.7	15.2	7.5	7.1	5.7	2	1.5	2	
RMRNSD(best)	24.7	19.2	22.2	57.9	30.2	43.5	25.5	62.4	3.8
RMRNSD(worst)	5.5	15.9	3.6	8.4	4.7	13.8	1	3.9	
SCWA(best)	12.1	20.4	10.6	24.5	6.3	25.6	11.9	34.4	2.5
SCWA(worst)	7.3	11.4	6.9	12.8	3.8	10	3.3	10.8	

for the worst case. The sample, which has the highest value of SIR for the worst case, is obtained by the Lin-PG algorithm. Considering the elapsed time, the PL, GPSR-BB, SCWA, and IPG belong to the fastest algorithms, but the Lin-PG and IPN algorithms are the slowest.

The multilayer technique generally improves the performance of all the tested algorithms. The highest improvement can be observed for the PSESOP algorithm, especially when the number of inner iterations is greater than one.

In summary, no algorithm strictly dominates the other. The selection of the algorithm depends on a size of the problem to be solved. Nevertheless, the projected gradient algorithms seem to be much better in our tests than the multiplicative algorithms, provided that we can use the squared Euclidean cost function which is suitable for data with a Gaussian noise.

REFERENCES

1. A. Cichocki, R. Zdunek, and S. Amari. New algorithms for non-negative matrix factorization in applications to blind source separation. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP2006*, volume 5, pages 621–624, Toulouse, France, May 14–19 2006.
2. J. Piper, V. P. Pauca, R. J. Plemmons, and M. Giffin. Object characterization from spectral data using nonnegative matrix factorization and information theory. In *Proc. AMOS Tech. Conf.*, Maui, HI, September 2004.
3. M. N. Schmidt and M. Mørup. Nonnegative matrix factor 2-D deconvolution for blind single channel source separation. *Springer LNCS*, 3889:700–707, 2006.
4. A. Ziehe, P. Laskov, K. Pawelzik, and K.-R. Mueller. Non-negative sparse coding for general blind source separation. In *NIPS 2003*, Vancouver, Canada, 2003.
5. W. Wang, Y. Luo, J.A. Chambers, and S. Sanei. Non-negative matrix factorization for note onset detection of audio signals. In *Proc. IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2006)*, pages 447–452, Maynooth, Ireland, September 6–8 2006.
6. W. Wang. Squared euclidean distance based convolutive non-negative matrix factorization with multiplicative learning rules for audio pattern separation. In *Proc. 7-th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT 2007)*, Cairo, Egypt, December 15–18 2007.
7. H. Li, T. Adali, and D. Emge W. Wang. Non-negative matrix factorization with orthogonality constraints for chemical agent detection in raman spectra. In *IEEE Workshop on Machine Learning for Signal Processing*. Mystic, USA, 2005.
8. P. Sajda, S. Du, T. Brown, L. Parra, and R. Stoyanova. Recovery of constituent spectra in 3D chemical shift imaging using nonnegative matrix factorization. In *4th International Symposium on Independent Component Analysis and Blind Signal Separation*, pages 71–76, Nara, Japan, April 2003.
9. P. Sajda, S. Du, T. R. Brown, R. Stoyanova D. C. Shungu, Xiangling Mao, and L. C. Parra. Nonnegative matrix factorization for rapid recovery of constituent spectra in magnetic resonance chemical shift imaging of the brain. *IEEE Trans. Medical Imaging*, 23(12):1453–1465, 2004.

10. P. Sajda, S. Du, and L. Parra. Recovery of constituent spectra using non-negative matrix factorization. In *Proceedings of SPIE*, volume 5207, pages 321–331. Wavelets: Applications in Signal and Image Processing, 2003.
11. D. D. Lee and H. S. Seung. Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401:788–791, October 1999.
12. W. Liu and N. Zheng. Non-negative matrix factorization based methods for object recognition. *Pattern Recognition Letters*, 25(8):893–897, 2004.
13. M. W. Spratling. Learning image components for object recognition. *Journal of Machine Learning Research*, 7:793–815, 2006.
14. Y. Wang, Y. Jia, C. Hu, and M. Turk. Non-negative matrix factorization framework for face recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(4):495–511, 2005.
15. P. Smaragdis. Non-negative matrix factor deconvolution; Extraction of multiple sound sources from monophonic inputs. *Lecture Notes in Computer Science*, 3195:494–499, 2004.
16. P. Smaragdis. Convolutive speech bases and their application to supervised speech separation. *IEEE Transactions on Audio, Speech and Language Processing*, 15(1):1–12, January 2007.
17. J-H. Ahn, S. Kim, J-H. Oh, and S. Choi. Multiple nonnegative-matrix factorization of dynamic PET images. In *ACCV*, page 5, 2004.
18. P. Carmona-Saez, R. D. Pascual-Marqui, F. Tirado, J. M. Carazo, and A. Pascual-Montano. Biclustering of gene expression data by non-smooth non-negative matrix factorization. *BMC Bioinformatics*, 7(78), 2006.
19. D. Guillamet, B. Schiele, and J. Vitrià. Analyzing non-negative matrix factorization for image classification. In *16th International Conference on Pattern Recognition (ICPR'02)*, volume 2, pages 116–119, Quebec City, Canada, August 2002.
20. D. Guillamet and J. Vitrià. Classifying faces with nonnegative matrix factorization. In *Proc. 5th Catalan Conference for Artificial Intelligence*, Castello de la Plana, Spain, 2002.
21. D. Guillamet, J. Vitrià, and B. Schiele. Introducing a weighted nonnegative matrix factorization for image classification. *Pattern Recognition Letters*, 24(14):2447–2454, 2003.
22. O. G. Okun. Non-negative matrix factorization and classifiers: experimental study. pages 550–555, Marbella, Spain, 2004.
23. O. G. Okun and H. Priisalu. Fast nonnegative matrix factorization and its application for protein fold recognition. *EURASIP Journal on Applied Signal Processing*, pages Article ID 71817, 8 pages, 2006.
24. A. Pascual-Montano, J. M. Carazo, K. Kochi, D. Lehmean, and R. Pacual-Marqui. Nonsmooth nonnegative matrix factorization (nsNMF). *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(3):403–415, 2006.
25. V. P. Pauca, F. Shahnaz, M. W. Berry, and R. J. Plemmons. Text mining using non-negative matrix factorizations. In *Proc. SIAM Inter. Conf. on Data Mining*, Orlando, FL, April 2004.
26. F. Shahnaz, M. Berry, P. Pauca, and R. Plemmons. Document clustering using non-negative matrix factorization. *Journal on Information Processing and Management*,

42:373–386, 2006.
27. T. Li and Ch. Ding. The relationships among various nonnegative matrix factorization methods for clustering. In *Proc. 6th International Conference on Data Mining (ICDM06)*, pages 362–371, Washington, DC, USA, 2006. IEEE Computer Society.
28. C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix tri-factorizations for clustering. In *KDD06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 126–135, New York, NY, USA, 2006. ACM Press.
29. R. Zass and A. Shashua. A unifying approach to hard and probabilistic clustering. In *International Conference on Computer Vision (ICCV)*, Beijing, China, Oct. 2005.
30. A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with Bregman divergences. In *SIAM International Conf. on Data Mining*, Lake Buena Vista, Florida, April 2004. SIAM.
31. H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum sum squared residue based co-clustering of gene expression data. In *Proc. 4th SIAM International Conference on Data Mining (SDM)*, pages 114–125, Florida, 2004.
32. S. Wild. *Seeding non-negative matrix factorization with the spherical k-means clustering*. M.Sc. Thesis, University of Colorado, 2000.
33. M. Berry, M. Browne, A. Langville, P. Pauca, and R. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics and Data Analysis*, 52(1):155–173, 2007.
34. Y. C. Cho and S. Choi. Nonnegative features of spectro-temporal sounds for classification. *Pattern Recognition Letters*, 26:1327–1336, 2005.
35. J. P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov. Metagenes and molecular pattern discovery using matrix factorization. volume 101, pages 4164–4169. PNAS, 2000.
36. N. Rao, S. J. Shepherd, and D. Yao. Extracting characteristic patterns from genome – wide expression data by non-negative matrix factorization. In *Proc. of the 2004 IEEE Computational Systems Bioinformatics Conference (CSB 2004)*, Stanford, CA, August 2004.
37. A. Cichocki, R. Zdunek, and S. Amari. Csiszar's divergences for non-negative matrix factorization: Family of new algorithms. *Springer LNCS*, 3889:32–39, 2006.
38. A. Cichocki, S. Amari, R. Zdunek, R. Kompass, G. Hori, and Z. He. Extended SMART algorithms for non-negative matrix factorization. *Springer LNAI*, 4029:548–562, 2006.
39. R. Zdunek and A. Cichocki. Non-negative matrix factorization with quasi-Newton optimization. *Springer LNAI*, 4029:870–879, 2006.
40. P. Paatero. Least-squares formulation of robust nonnegative factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 37:23–35, 1997.
41. P. Paatero and U. Tapper. Positive matrix factorization: A nonnegative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 1994.
42. D. D. Lee and H. S. Seung. Algorithms for nonnegative matrix factorization. In *NIPS*, pages 556–562, 2000.
43. M. T. Chu, F. Diele, R. Plemmons, and S. Ragni. Optimality, computation, and interpretation of nonnegative matrix factorizations. Technical report, Departments of

Mathematics and Computer Science, Wake Forest University, U.S.A., 2004.
44. P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.
45. C.-J. Lin. Projected gradient methods for non-negative matrix factorization. *Neural Computation*, 19(10):2756–2779, October 2007.
46. A. Cichocki and R. Zdunek. Multilayer nonnegative matrix factorization. *Electronics Letters*, 42(16):947–948, 2006.
47. A. Cichocki and R. Zdunek. Multilayer nonnegative matrix factorization using projected gradient approaches. *International Journal of Neural Systems*, ~~to appear.~~
48. R. Zdunek and A. Cichocki. Nonnegative matrix factorization with constrained second-order optimization. *Signal Processing*, 87:1904–1916, 2007.
49. A. Cichocki and R. Zdunek. NMFLAB for Signal and Image Processing. Technical report, Laboratory for Advanced Brain Signal Processing, BSI, RIKEN, Saitama, Japan, 2006.
50. J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.
51. G. Narkiss and M. Zibulevsky. Sequential subspace optimization method for large-scale unconstrained problems. *Optimization Online*, page 26, October 2005.
52. M. Elad, B. Matalon, and M. Zibulevsky. Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization. *Journal on Applied and Computational Harmonic Analysis.*, 2007. in print.
53. A. Nemirovski. Orth-method for smooth convex optimization (in russian). *Izvestia AN SSSR, Ser. Tekhnicheskaya Kibernetika (the journal is translated to English as Engineering Cybernetics. Soviet J. Computer and Systems Sci.)*, 2, 1982.
54. B. Morini S. Bellavia, M. Macconi. An interior point Newton-like method for non-negative least-squares problems with degenerate solution. *Numer. Linear Algebra Appl.*, 13:825–846, 2006.
55. R. W. Freund and N. M. Nachtigal. QMR: A quasi-minimal residual method for non-Hermitian linear systems. *Numerische Mathematik*, 60(1):315–339, 1991.
56. R. Fletcher. Conjugate gradient methods for indefinite systems. In *Proc. of the Dundee Biennial Conference on Numerical Analysis*, pages 73–89, Castello de la Plana, Spain, 1975. Springer-Verlag.
57. C. Lanczos. Solution of systems of linear equations by minimized iterations. *Journal of Research of the National Bureau of Standards*, 49(1):33–53, 1952.
58. H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13(2):631–644, 1992.
59. Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, 1986.
60. M. R. Hestenes and E. Stiefel. Method of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Standards*, 49:409–436, 1952.
61. P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems*. SIAM, Philadelphia, 1998.
62. C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and

sparse least squares. *ACM Trans. Math. Software*, 8:43–71, 1982.
63. V. Franc, V. Hlaváč, and M. Navara. Sequential coordinate-wise algorithm for the non-negative least squares problem. In André Gagalowicz and Wilfried Philips, editors, *CAIP 2005: Computer Analysis of Images and Patterns*, volume 1 of *LNCS*, pages 407–414, Berlin, Germany, September 2005. Springer-Verlag.
64. A. Cichocki and R. Zdunek. Regularized alternating least squares algorithms for non-negative matrix/tensor factorizations. *Springer LNCS*, 4493:793–802, June 3–7 2007.
65. J. G. Nagy and Z. Strakos. Enforcing Nonnegativity in Image Reconstruction Algorithms. volume 4121 of *Mathematical Modeling, Estimation, and Imaging*, pages 182–190, 2000.