Exact and Heuristic Algorithms
for Semi-Nonnegative Matrix Factorization

Nicolas Gillis
Department of Mathematics and Operational Research
Faculté Polytechnique, Université de Mons
Rue de Houdain 9, 7000 Mons, Belgium
nicolas.gillis@umons.ac.be

Abhishek Kumar
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
abhishk@us.ibm.com

Abstract

Given a matrix $M$ (not necessarily nonnegative) and a factorization rank $r$, semi-nonnegative matrix factorization (semi-NMF) looks for a matrix $U$ with $r$ columns and a nonnegative matrix $V$ with $r$ rows such that $UV$ is the best possible approximation of $M$ according to some metric. In this paper, we study the properties of semi-NMF from which we develop exact and heuristic algorithms. Our contribution is threefold. First, we prove that the error of a semi-NMF of rank $r$ has to be smaller than the best unconstrained approximation of rank $r - 1$. This leads us to a new initialization procedure based on the singular value decomposition (SVD) with a guarantee on the quality of the approximation. Second, we propose an exact algorithm (that is, an algorithm that finds an optimal solution), also based on the SVD, for a certain class of matrices (including nonnegative irreducible matrices) which we use as a heuristic for matrices not belonging to that class. Numerical experiments illustrate that this second heuristic performs extremely well, and allows us to compute optimal semi-NMF decompositions in many situations. Finally, we analyze the computational complexity of semi-NMF and prove that it is NP-hard, already in the rank-one case (that is, for $r = 1$).

Keywords. semi-nonnegative matrix factorization, semi-nonnegative rank, initialization, algorithms.

1 Introduction

Semi-nonnegative matrix factorization (semi-NMF) can be defined as follows: Given a matrix $M \in \mathbb{R}^{m \times n}$ and a factorization rank $r$, solve

$$
\min_{U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{r \times n}} \|M - UV\|_F^2 \quad \text{such that} \quad V \geq 0,
$$

where $\|.\|_F$ is the Frobenius norm and $V \geq 0$ means that $V$ is component-wise nonnegative. Note that any other suitable metric could be used but we focus in this paper on this particular objective function. Semi-NMF has been used in the context of data analysis and clustering [7]. In fact, letting each column
of the input matrix represent an element of a data set (there are \( n \) elements in dimension \( m \)), the semi-NMF decomposition can be equivalently written as

\[
M(\cdot, j) \approx \sum_{k=1}^{r} U(\cdot, k) V(k, j) \quad \text{for all } j,
\]

so that each column of \( M \) is a conic combination of the columns of \( U \) since \( V \geq 0 \). Each column of \( U \) can then be interpreted as a cluster centroid while the columns of \( V \) are the weights needed to reconstruct approximately each column of \( M \) using the columns of \( U \) and hence can be interpreted as cluster membership indicators; see the discussion in [7]. Semi-NMF has been used successfully for example for motion segmentation with missing data [15], image super-resolution [3] or hyperspectral unmixing [21].

Let us define the semi-nonnegative rank of matrix \( M \), denoted \( \text{rank}_s(M) \), as the smallest \( r \) such that there exists \( U \in \mathbb{R}^{m \times r} \) and \( V \in \mathbb{R}^{r \times n} \) with \( M = UV \) and \( V \geq 0 \). Let us also define exact semi-NMF, a problem closely related to semi-NMF, as follows: Given a matrix \( M \in \mathbb{R}^{m \times n} \), compute its semi-nonnegative rank \( r_s = \text{rank}_s(M) \) and a corresponding factorization \( U \in \mathbb{R}^{m \times r_s} \) and \( V \in \mathbb{R}^{r_s \times n} \) such that \( M = UV \) and \( V \geq 0 \). We also denote \( \text{rank}(M) \) the usual rank of a matrix \( M \), and \( \text{rank}_+(M) \) its nonnegative rank which is the smallest \( k \) such that there exists \( U \in \mathbb{R}^{m \times k_+} \) and \( V \in \mathbb{R}^{k_+ \times n} \) with \( M = UV \); see, e.g., [10] and the references therein. By definition, we have

\[
\text{rank}(M) \leq \text{rank}_s(M) \leq \text{rank}_+(M).
\]

The paper is organized as follows.

- In Section 2, we prove that the error of a semi-NMF of rank \( r \) has to be smaller than the best unconstrained approximation\(^1\) of rank \( r - 1 \) (Theorem 1), which implies \( \text{rank}_s(M) \leq \text{rank}(M) + 1 \). This leads us to a new initialization procedure for semi-NMF based on the singular value decomposition (SVD) with a guarantee on the quality of the approximation (Algorithm 2).

- In Section 3, we prove that solving exact semi-NMF can be done in polynomial time (Theorem 3). In particular, we show that \( \text{rank}_s(M) = \text{rank}(M) \) if and only if a positive vector belongs to the row space of \( M \) (after having removed its zero columns), otherwise \( \text{rank}_s(M) = \text{rank}(M) + 1 \) (Theorem 2). We propose an algorithm that solves semi-NMF\(^1\) for a certain class of matrices (which includes nonnegative matrices \( M \) for which \( M^T M \) is irreducible) and requires one SVD computation (Algorithm 3). We also generalize this algorithm for matrices not belonging to that class, and, in Section 5, we show that it performs extremely well, often leading to optimal solutions of semi-NMF\(^1\).

- In Section 4, we prove that semi-NMF is NP-hard already for \( r = 1 \) (Theorem 5). In light of the results above, this shows that computing (approximate) semi-NMF is much more difficult than computing exact semi-NMF (unless \( P = NP \)).

---

\(^1\) In the remainder of the paper, unless stated otherwise, we will refer to the best rank-\( r \) approximation \( X \) of \( M \) as an optimal solution of \( \min_{X \text{ such that } \text{rank}(X) \leq r} \| M - X \|_F^2 \). Note that the best rank-\( r \) unconstrained approximation of a given matrix is not necessarily unique. In fact, it is if and only if the \( r \)th and \( (r + 1) \)th singular values are distinct; see, e.g., [12]. However, we will use in this paper this abuse of language as the non-uniqueness issue does not play a role in our developments.
Remark 1. While finishing up this paper, we noticed the very recent paper [6] (available online 17 October 2014). It treats the exact semi-NMF problem, and studies Theorems 1 and 2 of this paper.

Our contribution goes further than proving Theorems 1 and 2, and is rather oriented towards algorithmic aspects: we propose (i) a polynomial-time algorithm for exact semi-NMF, and (ii) a very efficient way to initialize semi-NMF algorithms which is provably optimal for a subclass of matrices; see Algorithm 3. Moreover, we prove NP-hardness of semi-NMF; see Section 4.

2 Semi-NMF based on Unconstrained Low-Rank Approximations

Given any rank-\(r\) factorization \((A, B)\) of a matrix \(M = AB\), an exact rank-2\(r\) semi-NMF can be constructed since

\[ M = AB = A(B_+ - B_-) = [A, -A] \begin{bmatrix} B_+ \\ B_- \end{bmatrix}, \]

where \(B_+ = \max(B, 0) \geq 0\) and \(B_- = \max(-B, 0) \geq 0\) so that \(B = B_+ - B_-\). This implies

\[ \text{rank}_s(M) \leq 2 \text{rank}(M). \] (2)

2.1 Tight Upper Bound based on the Usual Rank

A much better bound than (2) based on the usual rank can be derived. In fact, we now show that any factorization of rank \(k\) can be transformed into a semi-NMF of rank \(k + 1\).

Theorem 1 (see also [6], Lem. 1). Let \(A \in \mathbb{R}^{m \times k}\) and \(B \in \mathbb{R}^{k \times n}\). Then, there exists \(U \in \mathbb{R}^{m \times (k+1)}\) and \(V \in \mathbb{R}^{(k+1) \times n}\) such that \(V \geq 0\) and \(UV = AB\).

Proof. Let us define \(\tilde{a} = -\sum_i A(:, i) = -Ae\) where \(e\) is the vector of all ones of appropriate dimensions. Let also

\[ U = [A \tilde{a}] \quad \text{and} \quad V(:, j) = \begin{bmatrix} B(:, j) \\ 0 \end{bmatrix} + \max_i \left( 0, \max_j (-B_{ij}) \right) e \in \mathbb{R}^{k+1} \quad \text{for all } 1 \leq j \leq n. \]

We have for all \(j\) that

\[ UV(:, j) = [A \tilde{a}] \begin{bmatrix} B(:, j) \\ 0 \end{bmatrix} + \max_i \left( 0, \max_j (-B_{ij}) \right) e = AB(:, j) + \max_i \left( 0, \max_j (-B_{ij}) \right) [A \tilde{a}] e = AB(:, j), \]

since \([A \tilde{a}] e = 0\) by construction.

Theorem 1 can be geometrically interpreted as follows: any set of data points in a \(r\)-dimensional space can be enclosed in the convex hull of \(r + 1\) vertices. For example, any set of points in a two-dimensional affine subspace is enclosed in a triangle (the triangle just needs to be big enough to contain all data points); see [6, Section 4] for more details.

Corollary 1. For any matrix \(M\), we have

\[ \text{rank}(M) \leq \text{rank}_s(M) \leq \text{rank}(M) + 1. \]

\[2\] Note however that the case of matrices with zero columns is not properly treated in [6]; see Theorem 2.
\textbf{Proof.} This follows from Theorem 1.

This implies that either \( \text{rank}_s(M) = \text{rank}(M) \) or \( \text{rank}_s(M) = \text{rank}(M) + 1 \). Observe that

- The above bound is tight. For example, the matrix
  \[ M = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{pmatrix} \]
  satisfies \( \text{rank}_s(M) = \text{rank}(M) + 1 = 3 \) (because the cone spanned by the columns of \( M - \mathbb{R}^2 \) cannot be represented as a cone spanned by two vectors).

- When \( n \) is large \( (n \gg \text{rank}(M) = r) \), in general, \( \text{rank}_s(M) = r + 1 \). In fact, it is not likely for a set of \( n \) points in a \( r \)-dimensional space to be spanned by a cone with \( r \) rays when \( n \gg r \). This would require these vectors to be contained in the same half space (see Section 3 for a complete characterization). For example, in two dimensions, if we generate vectors uniformly at random on the unit disk, the probability for these vectors to be in the same half space goes to zero extremely fast as \( n \) grows (faster than \( 2^{-n+1} \)).

- The function \( \text{rank}_s(.) \) is not invariant under transposition, that is, \( \text{rank}_s(M) \) is not necessarily equal to \( \text{rank}_s(M^T) \), although they cannot differ by more than one (see Corollary 1). For example, the matrix
  \[ M = \begin{pmatrix} -1 & 0 & -1 \\ 0 & -1 & -1 \\ 1 & 1 & 2 \end{pmatrix} = \begin{pmatrix} -1 & 0 & -1 \\ 0 & -1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \]
  satisfies \( \text{rank}_s(M) = 2 \neq \text{rank}_s(M^T) = 3 \).

\subsection{Algorithm for Semi-NMF}

A simple yet effective algorithm for semi-NMF is a block coordinate descent method that alternatively optimizes over \( U \) for \( V \) fixed and over \( V \) for \( U \) fixed:

- The problem in \( U \) is an unconstrained least squares and can be solved with dedicated solvers.

- The problem in \( V \) is a nonnegative least squares problem. To solve this problem, we propose to use a block coordinate descent method on the rows of \( V \) since the optimal solution for a given row (all other rows being fixed) has a closed-form solution; see, e.g., [9] and the references therein.

Algorithm 1 implements this strategy and is guaranteed to converge to a stationary point of \( (1) \) because each block of variables is optimized exactly and achieves a unique global minimize (Prop. 2.7.1).

\footnote{Given that \( U \) and \( V \) remain full rank.}
Algorithm 1 Coordinate Descent for Semi-NMF

Input: A matrix $M \in \mathbb{R}^{m \times n}$, an initialization $V \in \mathbb{R}_{+}^{r \times n}$, a maximum number of iterations maxiter.

Output: A rank-$r$ semi-NMF $(U, V)$ of $M \approx UV$ with $V \geq 0$.

1: for $i = 1 : \text{maxiter}$ do
2: $U \leftarrow \arg\min_{X \in \mathbb{R}^{m \times r}} ||M - XV||_F^2$ ($= M/V$ in Matlab)
3: % Coordinate descent on the rows of $V$
4: for $i = 1 : r$ do
5: $V(i,:)^T \leftarrow \arg\min_{x \in \mathbb{R}_+^n} \frac{||M - U(:,I)V(I,:) - U(:,i)x^T||_F^2}{||U(:,i)||_2^2}$
   where $I = \{1, 2, \ldots, r\}\{i\}$.
6: end for
7: end for

2.3 SVD-based Initialization

We can use the construction of Theorem 1 to initialize semi-NMF algorithms such as Algorithm 1; see Algorithm 2. Given a rank-$r$ approximation $(A, B)$ of $M \approx AB$ computed via the truncated SVD, we flip the sign of the rows of $B$ (and the columns of $A$ accordingly) so that the minimum on each row of $B$ is maximized. (Note that other sign permutations exist; see, e.g., [5].) The motivation behind this choice is to reduce the effect of the correction done at step 5 of Algorithm 2. We have the following result:

Corollary 2. Let $M \in \mathbb{R}^{m \times n}$, and let $M_{r-1}$ be its best rank-$(r - 1)$ approximation with respect to the norm $||.||$, then

$$\min_{U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}_{+}^{r \times n}} ||M - UV|| \leq ||M - M_{r-1}||.$$  \hfill (3)

The solution provided by Algorithm 1 initialized with Algorithm 2 satisfies this bound for the Frobenius norm.

Proof. This follows directly from Theorem 1 and the fact that Algorithm 1 generates a sequence of iterates monotonically decreasing the objective function.

In Section 5, we will compare Algorithm 2 with several other initialization strategies. It turns out that, although it is an appealing solution from a theoretical point of view (as it guarantees a solution with error smaller than the best rank-$(r - 1)$ approximation of $M$), it performs relatively poorly, in most cases worse than random initializations.

3 Exact Algorithm for Semi-NMF

In the previous section, we showed that for any matrix $M$, rank$_r(M)$ equals rank($M$) or rank($M$) + 1. In this section, we first completely characterize these two cases, and derive an algorithm to solve the exact semi-NMF problem.
Algorithm 2 SVD-based Initialization for Semi-NMF (see Theorem 1)

**Input:** A matrix $M \in \mathbb{R}^{m \times n}$ and a factorization rank $r$.

**Output:** A rank-$r$ semi-NMF $(U, V)$ of $M \approx UV$ with $V \geq 0$ achieving the same error than the best rank-$(r-1)$ approximation of $M$.

1. $[A, S, B^T] = \text{svds}(M, r-1)$; % See the Matlab function svds
2. $A = AS$;
3. For each $1 \leq i \leq r-1$: multiply $B(i, :)$ and $A(:, i)$ by -1 if $\min_j B(i, j) \leq \min_j (-B(i, j))$;
4. $U = \left[ A \ x_{-1}^T \right] e$;
5. $V(:, j) = \begin{pmatrix} B(:, j) \\ 0 \end{pmatrix} + \max(0, \max_i(-B_{ij})) e \quad 1 \leq j \leq n.$

Theorem 2 (\cite{6}). Let $M \in \mathbb{R}^{m \times n}$. The following statements are equivalent

(i) $\text{rank}(M) = \text{rank}_s(M)$.

(ii) There exists a non-zero vector $z \in \mathbb{R}^m$ such that $M(:, j)^T z > 0$ for all $j$ such that $M(:, j) \neq 0$. In other terms, all non-zero columns of $M$ belong to the interior of a half space $\mathcal{P}_z = \{x \in \mathbb{R}^m \mid x^T z \geq 0\}$ for some $z \neq 0$.

(iii) Given any factorization $(A, B) \in \mathbb{R}^{m \times r} \times \mathbb{R}^{r \times n}$ of $M = AB$ with $r = \text{rank}(M)$, all non-zero columns of $B$ belong to the interior of a half space $\mathcal{P}_y$ for some $y \neq 0$.

Proof. We assume without loss of generality (w.l.o.g.) that $M$ does not contain a zero column (otherwise discard it, which does not influence the conditions above).

The equivalence (ii) $\iff$ (iii) follows from simple linear algebra. Since the columns of the matrix $M$ belong to the interior of $\mathcal{P}_z$, we have $M^T z > 0$. Without loss of generality, we can take $z = Aw$ for some $w$. In fact, let us denote $A^\perp$ the orthogonal complement of $A$ so that, for any $z$, there exists $w$ and $w^\perp$ with $z = Aw + A^\perp w^\perp$ for which we have

$$0 < M^T z = B^T A^T (Aw + A^\perp w^\perp) = B^T A^T (Aw) = M^T (Aw).$$

Hence, replacing $z$ with $Aw$ does not modify $M^T z > 0$. Moreover, the derivation above shows that the columns of $B$ belong to the half space $\mathcal{P}_y$ with $y = A^T Aw = A^T z$ which proves (ii) $\Rightarrow$ (iii). Proving the direction (iii) $\Rightarrow$ (ii) is similar: the left inverse $A^\dagger \in \mathbb{R}^{r \times m}$ of $A$ exists since $\text{rank}(A)$ must be equal to $r$ ($A^\dagger A = I_r$) and taking $z = A^\dagger^T y$, we have

$$0 < B^T y = B^T \left( A^\dagger A \right)^T y = B^T A^T A^\dagger^T y = M^T z.$$

Let us show (iii) $\Rightarrow$ (i). We have $y \in \mathbb{R}^r$ such that $x = B^T y > 0$. Note that the rows of $B$ are different from zero since $B \in \mathbb{R}^{r \times n}$ and $\text{rank}(B) = r$. Hence we can flip the sign of the rows of $B$ along with the corresponding entries of $y$ (keeping $B^T y$ unchanged) so that the maximum entry on each row of $B$ is positive, that is, $\max_j B(i, j) > 0$ for all $i$ (see for example step 2 of Algorithm \cite{3}).

---

\footnote{This result is very similar to \cite{6} Th.3. However, the case of matrices with zero columns is not treated properly in \cite{6} Th.3. For example, according to \cite{6} Th.3, $\text{rank}(0) \neq \text{rank}_s(0)$ which is incorrect. In fact, the authors claim that: ‘As a consequence, $B$ contains a zero column which contradicts the fact that $\text{rank}(B) = r$’. This is not true: if $n > r$ (which is usually the case since $r = \text{rank}(M) \leq n$), $B$ can contain zero columns while $\text{rank}(B) = r$.}
Let 
\[ V(i,:) = B(i,:) + \alpha_i x^T = B(i,:) + \alpha_i y^T B = (e_i + \alpha_i y)^T B, \]
where \( \alpha_i = \max (0, \max_j \frac{-B(i,j)}{x_j}) \) for all \( 1 \leq i \leq r \) so that \( V(i,:) \geq 0 \), and \( e_i \) is the \( i \)th column of the identity matrix. In other terms,
\[ V = B + \alpha x^T = B + \alpha y^T B = (I + \alpha y^T) B \geq 0. \]
We can take \( U = A(I + \alpha x^T)^{-1} \) so that \( M = AB = UV \) with \( V \geq 0 \). Using Sherman-Morrison formula, we have that
\[ (I + \alpha y^T)^{-1} = I - \frac{\alpha y^T}{1 + y^T \alpha}, \]
hence \( U \) can be computed given that \( y^T \alpha \neq -1 \). It remains to show that \( y^T \alpha \neq -1 \). We have
\[
y^T \alpha = \sum_{i=1}^r y_i \alpha_i = \sum_{i=1}^r y_i \max \left(0, \max_j \frac{-B(i,j)}{x_j}\right) \\
> \sum_{i=1}^r y_i \left( \frac{1}{n} \sum_{j=1}^n \frac{-B(i,j)}{x_j}\right) \\
= -\frac{1}{n} \sum_{j=1}^n \left( \frac{\sum_{i=1}^r B(i,j)y_i}{\sum_{k=1}^r B(k,j)y_j} \right) = -1.
\]
The strict inequality follows from the fact that \( \max_j B(i,j) > 0 \) for all \( i \).

Let us show \((i) \Rightarrow (ii)\). Let \( r = \text{rank}(M) = \text{rank}_a(M) \), and \( M = UV \) with \( U \in \mathbb{R}^{m \times r} \) and \( V \in \mathbb{R}^{r \times n} \) where \( \text{rank}(U) = \text{rank}(V) = r \). Since no column of \( M \) is equal to zero, no column of \( V \) is hence \( V^T e > 0 \). Since \( U \) is full rank, its left inverse \( U^\dagger \in \mathbb{R}^{r \times m} \) exists \((U^\dagger U = I_r)\). This implies that
\[
M^T \left(U^\dagger^T e\right) = V^T U^T U^\dagger^T e = V^T (U^\dagger U)^T e = V^T e > 0.
\]
\[ \square \]

In the remainder of the paper, we say that a matrix \( M \) is semi-nonnegative if and only if \( \text{rank}(M) = \text{rank}_a(M) \) if and only if the non-zero columns of \( M \) are contained in the interior of a half space.

**Remark 2.** Note that if \( \text{rank}(M) = n \), then \( M \) necessarily contains a positive vector in its row space (since it spans \( \mathbb{R}^n \)) hence \( \text{rank}_a(M) = \text{rank}(M) = n \). In other words, \( \text{rank}_a(M) \leq n \) as we have the trivial decomposition \( M = MI_n \) where \( I_n \) is the \( n \)-by-\( n \) identity matrix; see also [6, Lemma 1].

**Theorem 3.** Given a matrix \( M \), solving exact semi-NMF can be done in polynomial time (both in the Turing machine model and the real model of computation).

**Proof.** By Theorem 2 it suffices to check whether a positive vector belongs to the row space of \( M \) (after having discarded the zero columns). For example, one can check whether the following linear system of inequalities has a solution:
\[
M(:,j)^T z \geq 1 \text{ for all } j \text{ such that } M(:,j) \neq 0.
\] (4)
If the system is feasible (which can be checked in polynomial time, both in the Turing machine model and the real model of computation \[4\]), rank\(_s(M) = \text{rank}(M)\), otherwise rank\(_s(M) = \text{rank}(M) + 1\).

The rank of a matrix and a corresponding low-rank factorization can be computed in polynomial time as well, e.g., using row-echelon form \[8\]. The factorization can be transformed into an exact semi-NMF using the construction of Theorem 1 in the case rank\(_s(M) = \text{rank}(M)\) + 1 and of Theorem 2 in the case rank\(_s(M) = \text{rank}(M)\).

In practice, it is better to compute a factorization of \(M\) using the singular value decomposition, which is implemented in our algorithms (Algorithms 2 and 3).

We have just showed how to compute an exact NMF. The same result can actually be used to compute approximate semi-NMF, given that the columns of the best rank-\(r\) approximation of \(M\) are contained in the same half-space.

Corollary 3. Let \(M \in \mathbb{R}^{m \times n}\). If the rank-\(r\) truncated SVD of \(M\) is semi-nonnegative, then semi-NMF \((1)\) can be solved in polynomial time in \(m\), \(n\) and \(O(\log(1/\epsilon))\) where \(\epsilon\) is the precision of the truncated SVD decomposition. Algorithm 3 is such a polynomial-time algorithm.

Proof. This follows from Theorem 2 and the fact that the rank-\(r\) truncated SVD provides an optimal rank-\(r\) approximation and can be computed up to any precision \(\epsilon\) in time polynomial in \(m\), \(n\) and \(O(\log(1/\epsilon))\); see, e.g., \[17, 20\] and the references therein. \(\square\)

If no positive vector belongs to the row space of the second factor \(B\) of a rank-\(r\) factorization \(AB\), then, by Theorem 2, there does not exist a rank-\(r\) semi-NMF \(U\) and \(V \geq 0\) such that \(AB = UV\) and the linear system \((4)\) is not feasible. In that case, we propose to use the following heuristic: solve

\[
\min_{y \in \mathbb{R}^r, \epsilon \in \mathbb{R}_+} \epsilon \quad \text{such that} \quad (B(:, j) + \epsilon e)^T y \geq 1 \quad \text{for all} \quad j \quad \text{such that} \quad B(:, j) + \epsilon e \neq 0. \tag{5}
\]

Although Problem \((5)\) is not convex, a solution can be obtained using a bisection method on the variable \(\epsilon\). In fact, the optimal solution \(\epsilon^*\) will belong to the interval \([0, \epsilon_+]\) where \(\epsilon_+ = \max_{i,k} \max(-B_{ik}, 0)\) since \(B + \epsilon_+ I \geq 0\) hence the problem is feasible (e.g., \(y = \epsilon\)). Note that the bisection method first checks whether \(\epsilon = 0\) is feasible in which case it terminates in one step and returns an optimal semi-NMF. Algorithm 3 implements this strategy and will be used in Section 5 to initialize Algorithm 1 and will be shown to perform extremely well.

Remark 3. It is interesting to note that the value of \(\epsilon^*\) tells us how far \(B\) is from being semi-nonnegative (hence \(AB\); see Theorem 2). In fact, by construction, the matrix \(B_{\epsilon^*} = B + \epsilon^* 1_{r \times n}\) is semi-nonnegative. The idea behind Algorithm 3 is to replace \(B\) with its semi-nonnegative approximation \(B_{\epsilon^*}\). If \(B\) is close to being semi-nonnegative, \(\epsilon^*\) will be small and Algorithm 3 will perform well; see Section 2 for the numerical experiments. Note that other strategies for finding a semi-nonnegative matrix close to \(B\) are possible and it would be interesting to compare them with Algorithm 3: this is a direction for further research.

3.1 Nonnegative Matrices

Theorem 2 implies that

Corollary 4. Let \(M \in \mathbb{R}_+^{m \times n}\), then rank\(_s(M) = \text{rank}(M)\).
Algorithm 3 Heuristic for Semi-NMF (see Theorem\textsuperscript{2} and Corollary\textsuperscript{3})

\textbf{Input:} A matrix $M \in \mathbb{R}^{m \times n}$, a factorization rank $r$.

\textbf{Output:} A rank-$r$ semi-NMF $(U, V)$ of $M \approx UV$ with $V \succeq 0$.

1: $[A, S, B^T] = \text{svds}(M, r)$ ; \textit{\% See the Matlab function \texttt{svds}}
2: For each $1 \leq i \leq r$: multiply $B(i,:) \times -1$ if $\min_j B(i,j) \leq \min_j (-B(i,j))$ ;
3: Let $(y^*, \epsilon^*)$ be the optimal solution of the following optimization problem
   \[
   \min_{y \in \mathbb{R}^r, \epsilon \in \mathbb{R}^+} \epsilon \text{ such that } (B(:,j) + \epsilon \epsilon^T) y \geq 1 \text{ for all } j \text{ such that } B(:,j) + \epsilon \epsilon^T \neq 0. 
   \]
   \% If $\epsilon^* = 0$ (\iff $B$ is semi-nonnegative), then the heuristic is optimal.
4: $x = (B + \epsilon^* 1_{r \times n}) y^* \geq 1$ ; \textit{\% $1_{r \times n}$ is the $r$-by-$n$ matrix of all ones.}
5: $\alpha_i = \max \left( 0, \max_j \frac{B(i,j)}{x^T} \right)$ for all $1 \leq i \leq r$ ;
6: $V = B + \alpha x^T$ ;
7: $U \leftarrow \arg\min_{X \in \mathbb{R}^{m \times r}} \| M - XV \|^2_F$ \textit{(= $M/V$ in Matlab)}.

\textbf{Proof.} In fact, any nonnegative vector different from zero belongs to the interior of the half space \( \mathcal{P}_e = \{ x \in \mathbb{R}^m | \sum_{i=1}^m x_i \geq 0 \} \).

We have seen that if the best rank-$r$ approximation of a matrix contains a positive vector in its row space, then an optimal semi-NMF of the corresponding matrix can be computed; see Corollary\textsuperscript{3}. This will be in general the case for nonnegative matrices. In particular, the Perron-Frobenius theorem guarantees that this will be the case when $M^TM$ is an irreducible nonnegative matrix (since its first eigenvector is positive). Recall that a symmetric matrix $A$ is irreducible if the graph induced by $A$ is strongly connected (every vertex is reachable from every vertex).

Corollary 5. Let $M \in \mathbb{R}^{m \times n}_+$. If $M^TM$ is irreducible, then semi-NMF \textsuperscript{11} can be solved via the truncated SVD for any rank $r$.

Corollaries\textsuperscript{3} and\textsuperscript{5} suggest that in almost all cases semi-NMF of nonnegative matrices can be computed using a simple transformation of an unconstrained approximation (such as the truncated SVD). This observation challenges the meaning of semi-NMF of nonnegative matrices: does semi-NMF of nonnegative matrices really make sense? In fact, most nonnegative matrices encountered in practice are irreducible, and, even if they are not, it is likely for Corollary\textsuperscript{3} to hold since the columns of the best rank-$r$ approximation of a nonnegative matrix are likely to be close to the nonnegative orthant hence belong to a half space (in particular $\mathcal{P}_e$). (Note that, by the Perron-Frobenius theorem, for $M \in \mathbb{R}^{m \times n}_+$ and $r = 1$, there always exists a nonnegative best rank-1 approximation.) In these cases, semi-NMF can be solved by a simple transformation of the SVD and it is not clear what semi-NMF brings to the table.

Note that several authors have proposed semi-NMF algorithms and applied them to nonnegative matrices, e.g., multiplicative updates where proposed in \textsuperscript{7, 18}. Our results shows that, from a theoretical point of view, this does not really make sense (since local optimization techniques such as the multiplicative updates usually converge relatively slowly and are not guaranteed to converge to an optimal solution). However, it would interesting to check whether the solutions obtained with these heuristics (always) generate optimal solutions under the above conditions. For example, we will see in Section\textsuperscript{5} that Algorithm\textsuperscript{11} does not always converge to an optimal solution for (semi-)nonnegative matrices for all initializations (in particular, when $r$ is large).
A direction for further research that would make sense for semi-NMF of nonnegative matrices is to add structure to the factors \( U \) and/or \( V \). For example, imposing \( V \) to be sparse would enhance the clustering property of semi-NMF. (Note that the construction of Theorem 2 usually generates a matrix \( V \) with a single zero per row.) In fact, if \( V \) is required to have a single non-zero entry per column equal to one, semi-NMF reduces to \( k \)-means [7].

**Remark 4.** The results of this section also apply to nonpositive matrices since \( M \) is nonpositive if and only if \(-M\) is nonnegative. Hence if we have a semi-NMF of \(-M = UV\), we have a semi-NMF for \( M = (-U)V\).

### 3.2 Semi-Nonnegative Matrices

If one performs a semi-NMF of a semi-nonnegative matrix \( M \) with factorization rank \( r = \text{rank}(M) = \text{rank}_s(M) \), then, by Theorem 2, the solution computed by Algorithm 3 will be optimal. However, if \( r < \text{rank}(M) \), it is not guaranteed to be the case. In this section, we provide a sufficient condition for Algorithm 3 to be optimal for semi-nonnegative matrices \( M \) when \( r < \text{rank}_s(M) \); see Theorem 4.

Intuitively, the idea is the following: the columns of the best rank-\( r \) approximation \( X \) of \( M \) should be relatively close to the columns of \( M \) hence it is likely that they also belong to a half space. In that case, by Corollary 3 Algorithm 3 is optimal. Note that if the best rank-\( k \) approximation of \( M \) contains a positive vector in its row space, then the best rank-\( r \) approximation of \( M \) for all \( r \geq k \) does as well since optimal low-rank approximations can be computed one rank-one factor at a time; see, e.g., [12].

**Theorem 4.** Let \( M \) be a semi-nonnegative matrix so that there exist \( z \) with \( M(:,j)^T z > 0 \) for all \( j \) such that \( M(:,j) \neq 0 \) and \( \|z\|_2 = 1 \). Let \( X \) be the best rank-\( r \) approximation of \( M \). If for all \( j \) such that \( M(:,j) \neq 0 \)

\[
\|M(:,j) - X(:,j)\|_2 < M(:,j)^T z,
\]

then \( X \) is semi-nonnegative, that is, there exists a rank-\( r \) semi-NMF \((U,V)\) such that \( X = UV \).

**Proof.** Let us denote the residual of the approximation \( E = M - X \). For all \( j \) such that \( M(:,j) \neq 0 \) we have

\[
X(:,j)^T z = (M(:,j) - E(:,j))^T z = M(:,j)^T z - E(:,j)^T z \geq M(:,j)^T z - \|E(:,j)\|_2 > 0,
\]

while \( M(:,j) = 0 \) \( \Rightarrow \) \( X(:,j) = 0 \), hence the result follows from Theorem 2.

Note that \( \|M(:,j) - X(:,j)\|_2 \leq \sigma_{r+1} \) for all \( j \) where \( \sigma_{r+1} \) is the \((r+1)\)th singular value of \( M \) hence the smaller \( \sigma_{r+1} \) is, the more likely it is for \( X \) to be semi-nonnegative. This also means that the larger \( r \) is, the more likely it is for Algorithm 3 to perform well (in particular, to return an optimal semi-NMF). This will be illustrated in Section 5 with some numerical experiments.

### 4 Computational Complexity of Semi-NMF

Despite the positive results described in the previous sections, the semi-NMF problem in the general case (that is, when the input matrix is not close to being semi-nonnegative) seems more difficult. The rank-one semi-NMF problem is the following: given \( M \in \mathbb{R}^{m \times n} \), solve

\[
\min_{u \in \mathbb{R}^m, v \in \mathbb{R}^n} \|M - uv^T\|_F^2 \quad \text{such that } v \geq 0.
\]
Theorem 5. Rank-one semi-NMF (6) is NP-hard.

Proof. Assume w.l.o.g. that $||v||_2 = 1$. Then the optimal solution for $u$ is given by $u^* = Mv$. Therefore, at optimality,

\[
||M - uv||_F^2 = ||M||_F^2 - 2u^T M v + ||uv^T||_F^2
\]

\[
= ||M||_F^2 - 2v^T M^T M v + ||Mv||_2^2
\]

\[
= ||M||_F^2 - 2||Mv||_2^2 + ||Mv||_2^2
\]

\[
= ||M||_F^2 - ||Mv||_2^2 = ||M||_F^2 - v^T (M^T M)v,
\]

hence (6) is equivalent to

\[
\max_{v \in \mathbb{R}^n} v^T (M^T M)v \quad \text{such that} \quad v \geq 0 \text{ and } ||v||_2 = 1.
\]

(7)

Since $M$ is arbitrary, $M^T M$ can represent any semidefinite positive matrix hence rank-one semi-NMF is equivalent to maximizing a convex quadratic over the unit ball in the nonnegative orthant. As explained by Noah D. Stein on Mathoverflow.net\(^5\)\(^\text{a}\), the problem (7) is equivalent to

\[
\max_{v \in \mathbb{R}^n} v^T B v \quad \text{such that} \quad v \geq 0 \text{ and } ||v||_2 = 1,
\]

(8)

where $B$ is any symmetric matrix (not necessarily semidefinite positive). In fact, if $B$ is not semidefinite positive, one can consider the problem with $B - \lambda_{\min}(B)I_n \succeq 0$ which only changes the objective function by a constant value since, for $||v||_2 = 1$,

\[
v^T (B - \lambda_{\min}(B)I_n) v = v^T B v - \lambda_{\min}(B).
\]

Let us use the following result: Checking copositivity of a symmetric matrix $C$, that is, checking whether the optimal value of

\[
\min_{v \geq 0, ||v||_2 = 1} v^T C v
\]

is nonnegative, is co-NP-complete \[^{16}\]. Since this problem can be solved using (8) with $B = -C$, this implies that rank-one semi-NMF (3) is NP-hard. \(\square\)

In comparison to NMF, this is a bit surprising: In fact, rank-one NMF can be solved in polynomial time (this follows from the Perron-Frobenius and Eckart-Young theorems) although it is NP-hard in general \[^{19}\]. The reason behind this difference is that semi-NMF allows both negative and positive elements in the input matrix (clearly, rank-one semi-NMF of nonnegative matrices can also be solved in polynomial time; see Section 3.1). In fact, rank-one NMF is also NP-hard if the input matrix is allowed to have both positive and negative signs \[^{11} \text{Cor.1}\] , that is, given a matrix $M \in \mathbb{R}^{m \times n}$, the problem

\[
\min_{u \in \mathbb{R}^m, v \in \mathbb{R}^n} ||M - uv^T||_F^2 \quad \text{such that} \quad u \geq 0 \text{ and } v \geq 0.
\]

is NP-hard.

\(^5\)See [http://mathoverflow.net/questions/48843/non-negative-quadratic-maximization](http://mathoverflow.net/questions/48843/non-negative-quadratic-maximization)
5 Numerical Experiments

In this section, we compare three strategies to initialize Algorithm 1 (which only requires the matrix \( V \) as an input):

1. **Random initialization (RAND):** each entry of \( V \) is generated following the uniform distribution in the interval \([0, 1]\), that is, \( V = \text{rand}(r, n) \) in Matlab notations (which we will reuse in the following).

2. **Algorithm 2 (AL2):** we use Algorithm 2 to initialize \( V \). This initialization guarantees the error to be smaller than the error of the best rank-\((r - 1)\) approximation.

3. **Algorithm 3 (AL3):** we use Algorithm 3 to initialize \( V \). This initialization generates an optimal solution for matrices whose best rank-\( r \) approximation contains a positive vector in its row space (which will be the case for example when \( M \) is nonnegative and \( M^T M \) irreducible).

In the following subsections, we generate several synthetic data sets where the dimensions of the input matrix \( M \) are \( m = 100 \) and \( n = 200 \), and the factorization rank is \( r = 20 \) and \( r = 80 \). For each generated synthetic matrix, we run Algorithm 1 with the three different initializations and consider the error obtained after 10 and 100 iterations. We use the notation RAND(10) (resp. RAND(100)) to refer to the algorithm that performs 10 (resp. 100) iterations of Algorithm 1 using RAND as an initialization. We use the same notation for the two other initializations AL1 and AL2, namely AL2(10) and AL2(100), and AL3(10) and AL3(100).

In order to compare meaningfully the error of solutions obtained for different input matrices, we use the following measure: given a semi-NMF \((U, V)\) of \( M \),

\[
\text{quality}(U, V) = 100 \left( \frac{||M - UV||_F}{||M - X||_F} - 1 \right) \geq 0,
\]

where \( X \) is the best rank-\( r \) unconstrained approximation of \( M \). It tell us how far away, in percent, the semi-NMF \( UV \) is from the best unconstrained solution \( X \). Note that \( \text{quality}(U, V) = 0 \) if and only if \( UV \) matches the error of the best rank-\( r \) approximation of \( M \) if and only if the best rank-\( r \) approximation of \( M \) is semi-nonnegative.

The Matlab code is available at [https://sites.google.com/site/nicolasgillis/](https://sites.google.com/site/nicolasgillis/). All tests are preformed using Matlab on a laptop Intel CORE i5-3210M CPU @2.5GHz 2.5GHz 6Go RAM. We use the function \text{linprog} of Matlab to solve the linear systems within the bisection method implemented for Problem [5] (we have also implemented a version using CVX [14, 13] for users’ convenience –note that the solution of [5] in \( y \) is non-unique and hence the solutions generated by different solvers are usually different).

### 5.1 Nonnegative and Semi-Nonnegative Matrices

In order to confirm our theoretical findings from Section 3, namely that AL3 computes optimal solutions for nonnegative matrices (given that \( M^T M \) is irreducible; see Corollary 5), and for many semi-nonnegative matrices (under a certain condition; see Theorem 4), we generate matrices as follows:

1. **Nonnegative matrices.** We generate each entry of \( X \) with the uniform distribution in the interval \([0,1]\), that is, we use \( M = \text{rand}(m, n) \). Note that, with probability one, \( M > 0 \) hence \( MM^T \) is irreducible.
2. Semi-nonnegative matrices of rank higher than \( r \). We generate matrices for which \( k = \text{rank}(M) = \text{rank}_s(M) = r + 10 \): we take \( M = UV \) where each entry of \( U \) is generated with the normal distribution (mean 0, variance 1) and each entry of \( V \) with the uniform distribution in the interval \([0,1] \), that is, we use \( M = \text{randn}(m,k) \ast \text{rand}(k,n) \).

For each value of \( r \) (20, 80), we generate 500 such matrices and Figure 1 displays the box plots of the measure defined in Equation (9) (we perform a single initialization for each generated matrix). These results confirm that AL3 performs perfectly for these types of matrices. Note that (i) AL2 performs relatively poorly and lead to solutions worse than RAND(100), and (ii) RAND(100) performs poorly for semi-nonnegative matrices when \( r = 80 \).

![Box plots](image)

Figure 1: Box plots of the error defined in Eq. (9) for nonnegative and semi-nonnegative matrices.

5.2 Semi-Nonnegative Matrices plus Noise

In this subsection, we generate semi-nonnegative matrices with \( \text{rank}(M) = \text{rank}_s(M) = r \) to which we add Gaussian noise. First we compute \( M = UV \) where each entry of \( U \) is generated with the normal
distribution (mean 0, variance 1) and each entry of \( V \) with the uniform distribution in the interval \([0, 1]\), that is, we use \( M = \text{randn}(m, r) \cdot \text{rand}(r, n) \), similarly as in the previous subsection. Then we compute the average of the entries of \( M \): 

\[
x_M = \frac{1}{mn} \sum_{i,j} |M(i,j)|
\]

and add Gaussian noise proportional to \( x_M \): we generate \( N = \delta x_M \text{randn}(m, n) \) where \( \delta \) is the noise level and then update \( M \leftarrow M + N \). For \( \delta = +\infty \), each entry of \( M \) is generated using the normal distribution, that is, \( M = \text{randn}(m, n) \).

Figure 2 displays the box plots of the measure defined in Eq. (9) for different values of \( \delta \). As in the previous subsection, for each experiment, we generate 500 matrices and report the quality obtained for a single initialization for each algorithm. We observe that:

- **AL2** performs quite poorly: it is always dominated by RAND(100). Hence, although AL2 is appealing from a theoretical point of view, it does not seem to have much practical use.

- Even for a relatively large noise level (in particular, \( \delta = 5 \)), **AL3** performs perfectly (all semi-NMF have quality (9) smaller than \( 10^{-2} \)) because the best rank-\( r \) approximation of \( M \) is close to being semi-nonnegative. The reason why **AL3** works perfectly even for very large noise levels can be explained with the way the matrix \( V \) was generated: using the uniform distribution for each entry. This makes the columns of matrix \( UV \) be far from the boundary of a well-chosen half space, that is, there exists \( z \) such that \((UV)^T z/\|z\|_2 \gg 0\). In particular, taking \( z = (U^\dagger)^T e \), the expected value of \((UV)^T z = V^T e\) is equal to \( \frac{r}{2} e \).

- When the noise level increases and the rank is not sufficiently large (the first example is for \( \delta = 10 \) and \( r = 20 \)), the condition of Theorem 4 is not (always) met and some solutions generated by **AL3** do not match the error of the best rank-\( r \) approximation.

- When only noise is present and the matrix is Gaussian (\( \delta = +\infty \)), the condition of Theorem 4 is never met and **AL3** fails most of the time to extract a semi-NMF whose error is close to the error of the best rank-\( r \) approximation. This is not surprising as these matrices are not likely to have semi-nonnegative best rank-\( r \) approximations.

However, when \( r \) is large (\( r = 80 \), **AL3** outperforms RAND and AL2; see Figure 2 (bottom right) and Figure 3 (right). We believe the reason is that, although the best rank-\( r \) approximations are not semi-nonnegative, they have a vector in their row space close enough to the nonnegative orthant hence **AL3** provides a good initial approximation.

On the contrary, when \( r \) small (\( r = 20 \), **AL3** performs in average slightly worse than RAND(100) although the gap between both approaches is not significant as shown in Figure 2 (bottom left); see also Figure 3.

Note that we have observed this behavior for other values of \( m, n \) and \( r \). As explained above, the reason is that, it becomes more likely as \( r \) increases for the row space of the best rank-\( r \) approximation of \( M \) to contain a vector close to the nonnegative orthant hence for **AL3** to perform well.

- It seems that **AL3** generates matrices close to stationary points of (1) as the difference between the boxplots of **AL3**(10) and **AL3**(100) is small on all these examples. This is another advantage of **AL3**: in all the experiments we have performed, it always allowed Algorithm 1 to converge extremely quickly (essentially within 10 iterations); see Figure 3 displaying the average value of the quality (9) for each iteration of the 500 Gaussian matrices.

Finally, the two main recommendations that we can give based on these experiments are the following:
Figure 2: Box plots of the error from Eq. (9): semi-nonnegative matrices with noise ($\delta = 5, 10, +\infty$).
1. AL3 works in general very well, being optimal for matrices whose best rank-\(r\) approximation is semi-nonnegative. Hence we would always recommend to try it on your favorite matrices.

2. RAND sometimes works better than AL3 (after sufficiently many iterations of a semi-NMF algorithm), in particular when the factorization rank \(r\) is small and the best rank-\(r\) approximation of the input matrix is far from being semi-nonnegative. It should be tried on matrices for which AL3 was not able to compute a semi-NMF whose error is close to the error of the best rank-\(r\) approximation.

Remark 5. In these experiments, we intentionally took \(n \geq m\) because an \(m\)-by-\(n\) matrix is less likely to be semi-nonnegative when \(n \geq m\). In particular, an \(m\)-by-\(n\) Gaussian matrix with \(n \leq m\) is semi-nonnegative with probability one since \(\text{rank}(M) = \text{rank}_s(M) = n\) with probability one (although this does not imply that its best rank-\(r\) approximation is; see Theorem 4). For example, running exactly the same experiment with \(m = 200, n = 100, r = 20\) and \(\delta = 10\), 86% of the solutions generated by AL3 match the best rank-\(r\) approximation up to 0.01% (while only 0.4% do for \(m = 100, n = 200, r = 20\); see Figure 2).

6 Conclusion

In this paper, we have addressed theoretical questions related to semi-NMF that led us to the design of exact and heuristic algorithms. Our contribution is three-fold. We showed that

- The approximation error of semi-NMF of rank \(r\) has to be smaller than the approximation error of its unconstrained counterpart of rank \(r - 1\). This result allowed us to design a new initialization procedure for semi-NMF that guarantees the error to be smaller than the error of the best rank-(\(r - 1\)) approximation; see Theorem 1 Algorithm 2. However, it seems that this initialization procedure does not work very well in practice.
• Exact semi-NMF can be solved in polynomial time (Theorem 3), and semi-NMF of a matrix $M$ can be solved in polynomial time up to any given precision with Algorithm 3 given that the best rank-$r$ approximation of $M$ is semi-nonnegative. Algorithm 3 can also handle cases when the aforementioned condition is not met, and we illustrated its effectiveness on several synthetic data sets.

• Semi-NMF is NP-hard in general, already in the rank-one case (Theorem 5).

Further research on semi-NMF includes the design of other initialization strategies (in particular in the case $r$ is small and $M$ is far from being a rank-$r$ semi-nonnegative matrix; see also Remark 3), and the analysis of constrained variants of semi-NMF such as sparse semi-NMF where $V$ is required to be sparse. In fact, sparse semi-NMF would in general make more sense as it has better clustering properties; see the discussions in Section 5.1 and in [7] for more details. In particular, it would be interesting to see how Algorithm 3 performs as an initialization strategy in that case.

References

[1] Bertsekas, D.: Nonlinear Programming: Second Edition. Athena Scientific, Massachusetts (1999)

[2] Bertsekas, D.: Corrections for the book nonlinear programming (2nd ed.) (2014). Available online at [http://www.athenasc.com/nlperrata.pdf](http://www.athenasc.com/nlperrata.pdf)

[3] Bevilacqua, M., Roumy, A., Guillemot, C., Morel, M.L.: Neighbor embedding based single-image super-resolution using semi-nonnegative matrix factorization. In: IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), pp. 1289–1292 (2012)

[4] Blum, L.: Computing over the reals: Where Turing meets Newton. Notices of the American Mathematical Society 51, 1024–1034 (2004)

[5] Bro, R., Acar, E., Kolda, T.: Resolving the sign ambiguity in the singular value decomposition. Journal of Chemometrics 22(2), 135–140 (2008)

[6] Chouh, M., Hanafi, M., Boukhetala, K.: Semi-nonnegative rank for real matrices and its connection to the usual rank. Linear Algebra and its Applications 466, 27–37 (2015)

[7] Ding, C., Li, T., Jordan, M.: Convex and semi-nonnegative matrix factorizations. IEEE Trans. on Pattern Analysis and Machine Intelligence 32(1), 45–55 (2010)

[8] Edmonds, J.: Systems of distinct representatives and linear algebra. J. Res. Nat. Bur. Standards, Sect. B 71(4), 241–245 (1967)

[9] Gillis, N., Glineur, F.: Accelerated multiplicative updates and hierarchical ALS algorithms for nonnegative matrix factorization. Neural Computation 24(4), 1085–1105 (2012)

[10] Gillis, N., Glineur, F.: On the geometric interpretation of the nonnegative rank. Linear Algebra and its Applications 437(11), 2685–2712 (2012)

[11] Gillis, N., Glineur, F.: A continuous characterization of the maximum-edge biclique problem. J. of Global Optimization 58(3), 439–464 (2014)
[12] Golub, G., Van Loan, C.: Matrix Computation, 3rd Edition. The Johns Hopkins University Press (1996)

[13] Grant, M., Boyd, S.: Graph implementations for nonsmooth convex programs. Recent Advances in Learning and Control (a tribute to M. Vidyasagar), V. Blondel, S. Boyd, and H. Kimura, editors, pages 95-110, Lecture Notes in Control and Information Sciences, Springer (2008)

[14] Grant, M., Boyd, S.: CVX: Matlab software for disciplined convex programming, version 1.21. http://cvxr.com/cvx/ (April 2011)

[15] Mo, Q., Draper, B.: Semi-nonnegative matrix factorization for motion segmentation with missing data. In: Computer Vision–ECCV 2012, pp. 402–415 (2012)

[16] Murty, K., Kabadi, S.: Some NP-complete problems in quadratic and nonlinear programming. Mathematical programming 39(2), 117–129 (1987)

[17] Trefethen, L., Bau III, D.: Numerical Linear Algebra. SIAM (1997)

[18] Trigeorgis, G., Bousmalis, K., Zafeiriou, S., Schuller, B.: A Deep Semi-NMF Model for Learning Hidden Representations. In: International Conference on Machine Learning (ICML ’14), vol. 32, pp. 1692–1700 (2014)

[19] Vavasis, S.: On the complexity of nonnegative matrix factorization. SIAM Journal on Optimization 20(3), 1364–1377 (2009)

[20] Vavasis, S.A.: Nonlinear optimization: complexity issues. Oxford University Press, Inc. (1991)

[21] Yokoya, N., Chanussot, J., Iwasaki, A.: Generalized bilinear model based nonlinear unmixing using semi-nonnegative matrix factorization. In: IEEE Int. Geoscience and Remote Sensing Symposium (IGARSS), pp. 1365–1368 (2012)