SIAM J. MATRIX ANAL. APPL.
Vol. 36, No. 4, pp. 1404–1424

EXACT AND HEURISTIC ALGORITHMS FOR SEMI-NONNEGATIVE MATRIX FACTORIZATION*

NICOLAS GILLIS† AND ABHISHEK KUMAR‡

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) from which we derive an initialization for matrices not belonging to that class. Numerical experiments illustrate that this second approach performs extremely well, and allows us to compute optimal semi-NMF decompositions in many situations. Finally, we analyze the computational complexity of semi-NMF proving its NP-hardness, already in the rank-one case (that is, for $r = 1$), and we show that semi-NMF is sometimes ill-posed (that is, an optimal solution does not exist).

Key words. low-rank matrix approximation, semi-nonnegative matrix factorization, semi-nonnegative rank, initialization, algorithms

AMS subject classifications. 15A23, 65F30

DOI. 10.1137/140993272

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

$$\min_{U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{r \times n}} ||M - UV||_F^2 \quad \text{such that} \quad V \geq 0, \tag{1.1}$$

where $||.||_F$ is the Frobenius norm and $V \geq 0$ means that V is componentwise 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 [9]. 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

$$M(:, j) \approx \sum_{k=1}^{r} U(:, k)V(k, j) \qquad \text{for all } j,$$

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 [9]. Semi-NMF has been used successfully for example for motion segmentation with missing data [21], image superresolution [4], or hyperspectral unmixing [27].

*Received by the editors October 28, 2014; accepted for publication (in revised form) by C.-H. Guo July 31, 2015; published electronically October 15, 2015.
http://www.siam.org/journals/simax/36-4/99327.html
†Department of Mathematics and Operational Research, Faculté Polytechnique, Université de Mons, 7000 Mons, Belgium (nicolas.gillis@umons.ac.be).
‡IBM T.J. Watson Research Center, Yorktown Heights, NY 10598 (abhishk@us.ibm.com).

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., [12] 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 2.1), 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 3.2). 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 3.1). We propose an algorithm that solves semi-NMF (1.1) for a certain class of matrices (which includes nonnegative matrices M for which M^TM 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 (1.1).
- In section 4, we prove that semi-NMF is NP-hard already for $r = 1$ (Theorem 4.1). In light of the results above, this shows that computing (approximate) semi-NMF is much more difficult than computing exact semi-NMF (unless $P = NP$). Moreover, we also show that semi-NMF is sometimes ill-posed (that is, an optimal solution does not exist).

Remark 1. While finishing up this paper, we noticed the very recent paper [7]. It treats the exact semi-NMF problem, and studies Theorems 2.1 and 3.1 of this paper.[2]

Our contribution goes further than proving Theorems 2.1 and 3.1, 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 and ill-posedness 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-$2r$ semi-NMF

[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 $\min_{X,\text{rank}(X)\leq r} ||M - X||_F^2$. 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 rth and $(r+1)$th singular values are distinct; see, e.g., [14]. However, we will use in this paper this abuse of language as the nonuniqueness issue does not play a role in our developments.

[2] Note however that the case of matrices with zero columns is not properly treated in [7]; see Theorem 3.1.

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

$$\operatorname{rank}_s(M) \leq 2 \operatorname{rank}(M). \tag{2.1}$$

2.1. Tight upper bound based on the usual rank. A much better bound than (2.1) 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 2.1 (see also [7], 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 $\bar{a} = -\sum_i A(:, i) = -Ae$, where e is the vector of all ones of appropriate dimensions. Let also

$$U = [A\ \bar{a}] \qquad \text{and} \qquad V(:, j) = \begin{pmatrix} B(:, j) \\ 0 \end{pmatrix} + \max\left(0, \max_i(-B_{ij})\right) e \ \in \ \mathbb{R}^{k+1}_+$$

for all $1 \leq j \leq n$. We have for all j that

$$\begin{aligned} UV(:, j) &= [A\,\bar{a}] \left[\begin{pmatrix} B(:, j) \\ 0 \end{pmatrix} + \max\left(0, \max_i(-B_{ij})\right) e \right] \\ &= AB(:, j) + \max\left(0, \max_i(-B_{ij})\right) [A\,\bar{a}]e = AB(:, j), \end{aligned}$$

since $[A\,\bar{a}]e = 0$ by construction. ∎

Theorem 2.1 can be geometrically interpreted as follows: *any set of data points in an 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 [7, section 4] for more details.

COROLLARY 2.2. *For any matrix M, we have*

$$\operatorname{rank}(M) \leq \operatorname{rank}_s(M) \leq \operatorname{rank}(M) + 1.$$

Proof. This follows from Theorem 2.1. ∎

This implies that either $\operatorname{rank}_s(M) = \operatorname{rank}(M)$ or $\operatorname{rank}_s(M) = \operatorname{rank}(M) + 1$. Observe the following:

- The above bound is tight. For example, the matrix

$$M = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{pmatrix}$$

  satisfies $\operatorname{rank}_s(M) = \operatorname{rank}(M) + 1 = 3$ (because the cone spanned by the columns of M, namely, $\mathbb{R}^2$, cannot be represented as a cone spanned by two vectors).
- When n is large ($n \gg \operatorname{rank}(M) = r$), in general, $\operatorname{rank}_s(M) = r + 1$. In fact, it is not likely for a set of n points in an 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, if we generate these 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.

- 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 2.2). For example, the matrix

$$M = \begin{pmatrix} -1 & 0 & -1 \\ 0 & -1 & -1 \\ 1 & 1 & 2 \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

satisfies $\text{rank}_s(M) = 2 \neq \text{rank}_s(M^T) = 3$.

2.2. 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., [11] and the references therein.

Algorithm 1 implements this strategy and is guaranteed to converge to a stationary point of (1.1) because each block of variables is optimized exactly and achieves a unique global minimizer[3] [2, 3] (Prop. 2.7.1). (Note that a value of maxiter between 100 and 500 usually gives good results, although this depends on the initialization and the dimensions m, n, and r; see section 5 for some numerical experiments.)

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: $\quad U \leftarrow \text{argmin}_{X \in \mathbb{R}^{m \times r}} ||M - XV||_F^2 \quad (= M/V \text{ in MATLAB})$
3: $\quad$ *% Coordinate descent on the rows of V*
4: $\quad$ **for** $i = 1 : r$ **do**
5:

$$V(i,:)^T \leftarrow \text{argmin}_{x \in \mathbb{R}_+^n} ||M - U(:,\mathcal{I})V(\mathcal{I},:) - U(:,i)x^T||_F^2$$
$$= \max\left(0, \frac{(M - U(:,\mathcal{I})V(\mathcal{I},:))^T U(:,i)}{||U(:,i)||_2^2}\right), \mathcal{I} = \{1, \ldots, r\} \backslash \{i\}.$$

6: $\quad$ **end for**
7: **end for**

Remark 2 (original semi-NMF algorithm). In the original paper introducing semi-NMF [9], the proposed algorithm is the following:

- The matrices U and V are initialized using k-means: the columns of U are taken as the cluster centroids of the columns of M, while V is the binary indicator matrix to which the constant 0.2 is added (for the multiplicative updates to be able to modify all entries of V; see below).

[3] Given that U and V remain full rank.

- V is updated using the following multiplicative updates: for all k, j,

$$V_{kj} \leftarrow V_{kj} \sqrt{\frac{\max(0,(U^TM)_{kj}) + \max(0,-(U^TUV)_{kj})}{\max(0,-(U^TM)_{kj}) + \max(0,(U^TUV)_{kj})}}.$$

  These updates are guaranteed to decrease the objective function.
- U is updated as in Algorithm 1, using the optimal solution for V fixed.

Hence this algorithm is rather similar to Algorithm 1, where V would be initialized with k-means and would be updated with the multiplicative updates. However, compared to Algorithm 1, the algorithm from [9] suffers from the following drawbacks:

- It is not guaranteed to converge to a stationary point (nonincreasningness is not a sufficient condition).
- It has a locking phenomenon: once an entry of matrix V is set to zero, it cannot be modified (because of the multiplicative nature).
- It sometimes runs into numerical problems, because the denominator in the update rules is equal to zero.
- Although it has almost exactly the same computational cost as Algorithm 1 (the update of V requires the matrix products U^TM and U^TUV in both cases), it converges significantly slower. The same observation was made by several works comparing coordinate descent approaches to multiplicative updates for optimizing U and V in NMF [8, 19, 17, 20, 11].

Moreover, in this paper, our goal is not to compare strategies to update matrices U and V but rather to compare initialization strategies. For these reasons, we do not use the algorithm from [9] in this paper, but we will compare their initialization based on k-means to our proposed approaches; see section 5.

2.3. SVD-based initialization. We can use the construction of Theorem 2.1 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., [6].) 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 2.3. *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*

$$\min_{U\in\mathbb{R}^{m\times r}, V\in\mathbb{R}^{r\times n}_+} ||M - UV|| \leq ||M - M_{r-1}||. \tag{2.2}$$

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

Proof. This follows directly from Theorem 2.1, 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 equal to the error of the best rank-$(r-1)$ approximation of M), it performs relatively poorly, in most cases worse than random initializations.

ALGORITHM 2. SVD-BASED INITIALIZATION FOR SEMI-NMF (SEE THEOREM 2.1).

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: $U = [A \quad -Ae]$;
5: $V(:,j) = \begin{pmatrix} B(:,j) \\ 0 \end{pmatrix} + \max\left(0, \max_i(-B_{ij})\right) e \qquad 1 \leq j \leq n.$

3. Exact algorithm for semi-NMF. In the previous section, we showed that for any matrix M, $\text{rank}_s(M)$ equals $\text{rank}(M)$ or $\text{rank}(M)+1$. In this section, we first completely characterize these two cases, and derive an algorithm to solve the exact semi-NMF problem.

THEOREM 3.1. *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 nonzero 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 nonzero 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 $z \neq 0$ or, equivalently, there exists a positive vector in the rows space of M after its zero columns have been removed.*

(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 nonzero columns of B belong to the interior of a half-space $\mathcal{P}_y$ for some $y \neq 0$.*

Remark 3. This result is very similar to [7, Thm. 3]. However, the case of matrices with zero columns is not treated properly in [7, Thm. 3]. For example, according to [7, Thm. 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 $\text{rank}(B) = r$.

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

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

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 3).

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 ith column of the identity matrix. In other terms,

$$V = B + \alpha x^T = B + \alpha y^T B = \left(I + \alpha y^T\right) B \geq 0.$$

We can take $U = A(I + \alpha y^T)^{-1}$ so that $M = AB = UV$ with $V \geq 0$. Using the Sherman–Morrison formula, we have that

$$\left(I + \alpha y^T\right)^{-1} = I - \frac{\alpha y^T}{1 + y^T \alpha},$$

hence U can be computed given that $y^T \alpha \neq -1$. It remains to show that $y^T \alpha \neq -1$. We have

$$\begin{aligned} y^T \alpha = \sum_{i=1}^r y_i \alpha_i &= \sum_{i=1}^r y_i \max\left(0, \max_j \frac{-B(i,j)}{x_j}\right) \\ &> \sum_{i=1}^r y_i \left(\frac{1}{n} \sum_{j=1}^n \frac{-B(i,j)}{x_j}\right) \\ &= \frac{-1}{n} \sum_{j=1}^n \left(\frac{\sum_{i=1}^r B(i,j) y_i}{\sum_{k=1}^r B(k,j) y_k}\right) = -1. \end{aligned}$$

The strict inequality follows from the fact that $\max_j B(i,j) > 0$ for all i.

Let us show $(i) \Rightarrow (ii)$. Let $r = \text{rank}(M) = \text{rank}_s(M)$, 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

$$M^T \left(U^{\dagger T} e\right) = V^T U^T U^{\dagger T} e = V^T (U^\dagger U)^T e = V^T e > 0. \qquad \square$$

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

Remark 4. Note that if $\text{rank}(M) = n$, then $M \in \mathbb{R}^{m \times n}$ necessarily contains a positive vector in its row space (since it spans $\mathbb{R}^n$), hence, $\text{rank}_s(M) = \text{rank}(M) = n$. We also have $\text{rank}_s(M) \leq n$ using the trivial decomposition $M = MI_n$, where I_n is the n-by-n identity matrix; see also [7, Lemma 1].

THEOREM 3.2. *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 3.1, 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:

$$M(:,j)^T z \geq 1 \text{ for all } j \text{ such that } M(:,j) \neq 0. \tag{3.1}$$

If the system is feasible (which can be checked in polynomial time, both in the Turing machine model and the real model of computation [5]), $\text{rank}_s(M) = \text{rank}(M)$, otherwise $\text{rank}_s(M) = \text{rank}(M) + 1$. 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 [10]. The factorization can be transformated into an exact semi-NMF using the construction of Theorem 2.1 in the case $\text{rank}_s(M) = \text{rank}(M)+1$ and of Theorem 3.1 in the case $\text{rank}_s(M) = \text{rank}(M)$. ∎

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

We have just shown 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 3.3. *Let $M \in \mathbb{R}^{m\times n}$. If the rank-$r$ truncated SVD of M is semi-nonnegative, then semi-NMF* (1.1) *can be solved in polynomial time in m, n, and $\mathcal{O}(\log(1/\epsilon))$, where ϵ is the precision of the truncated SVD decomposition. Algorithm* 3 *is such a polynomial-time algorithm.*

Proof. This follows from Theorem 3.1 and the fact that the rank-r truncated SVD provides an optimal rank-r approximation and can be computed up to any precision ϵ in time polynomial in m, n, and $\mathcal{O}(\log(1/\epsilon))$; see, e.g., [23, 25] and the references therein. ∎

If no positive vector belongs to the row space of the second factor B of a rank-r factorization AB, then, by Theorem 3.1, there does not exist a rank-r semi-NMF U and $V \geq 0$ such that $AB = UV$ and the linear system (3.1) is not feasible. In that case, we propose to use the following heuristic: solve

(3.2)
$$\min_{y\in\mathbb{R}^r,\epsilon\in\mathbb{R}_+} \epsilon \quad \text{such that} \quad (B(:,j) + \epsilon\, e)^T y \geq 1 \text{ for all } j \text{ such that } B(:,j) + \epsilon\, e \neq 0.$$

Although problem (3.2) is not convex, a solution can be obtained using a bisection method on the variable ϵ. In fact, the optimal solution ϵ^* will belong to the interval $[0, \epsilon_+]$, where $\epsilon_+ = \max_{i,k} \max(-B_{ik}, 0)$ since $B + \epsilon_+ \geq 0$; hence, the problem is feasible (e.g., $y = e$). 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. In our implementation, we used a relative precision of 10^{-3}, that is, we stop the algorithm as soon as $\epsilon_f - \epsilon_i \leq 10^{-3}\epsilon_+$, where ϵ_f is the smallest feasible ϵ found so far (initialized at ϵ_+), and ϵ_i is the largest infeasible ϵ found so far (initialized at 0) so that our bisection procedure has to solve at most ten linear systems (since $0.001 > 2^{-10}$). The reason we choose a relatively low precision is that high precision is not necessary because, when the optimal $\epsilon^* \neq 0$, the algorithm will be used as an initialization procedure for Algorithm 1 that will refine the semi-NMF solution locally.

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 5. It is interesting to note that the value of ϵ^* tells us how far B is from being semi-nonnegative (hence AB; see Theorem 3.1). 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_{ϵ^*}. If B is close to being semi-nonnegative, ϵ^* will be small and Algorithm 3 will perform well; see section 5 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.

ALGORITHM 3. HEURISTIC FOR SEMI-NMF (SEE THEOREM 3.1 AND COROLLARY 3.3).

Input: A matrix $M \in \mathbb{R}^{m \times n}$, a factorization rank r.
Output: A rank-r semi-NMF (U, V) of $M \approx UV$ with $V \geq 0$.

1: $[A, S, B^T] = \text{svds}(M, r)$; % *See the MATLAB function* `svds`
2: For each $1 \leq i \leq r$: multiply $B(i,:)$ by -1 if $\min_j B(i,j) \leq \min_j(-B(i,j))$;
3: Let (y^*, ϵ^*) be the optimal solution of the following optimization problem

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

% If $\epsilon^* = 0$ ($\iff$ B is semi-nonnegative), then the heuristic is optimal.
4: $x = (B + \epsilon^* 1_{r \times n})^T y^* \geq 1$; % $1_{r \times n}$ *is the r-by-n matrix of all ones.*
5: $\alpha_i = \max\left(0, \max_j \frac{-B(i,j)}{x(j)}\right)$ for all $1 \leq i \leq r$;
6: $V = B + \alpha x^T$;
7: $U \leftarrow \text{argmin}_{X \in \mathbb{R}^{m \times r}} ||M - XV||_F^2$ $(= M/V$ in MATLAB).

3.1. Nonnegative matrices. Theorem 3.1 implies the following.

COROLLARY 3.4. *Let $M \in \mathbb{R}^{m \times n}_+$, then* $\text{rank}_s(M) = \text{rank}(M)$.

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 3.3. This will be, in general, the case for nonnegative matrices. In fact, the Perron–Frobenius theorem guarantees that this will be the case when $M^T M$ is a irreducible nonnegative matrix (since its first eigenvector can be chosen positive). Recall that a matrix A is irreducible if the graph induced by A is strongly connected (every vertex is reachable from every other vertex).

COROLLARY 3.5. *Let $M \in \mathbb{R}^{m \times n}_+$. If $M^T M$ is irreducible, then semi-NMF* (1.1) *can be solved via the truncated SVD for any rank r.*

Corollaries 3.3 and 3.5 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 3.3 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 [9, 24]. Our results show 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 is interesting to check whether the solutions obtained with these heuristics (always) generate optimal solutions under the above conditions. We will see

in section 5 that Algorithm 1 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 3.1 usually generates a matrix V with a single zero per row.) In fact, if V is required to have a single nonzero entry per column equal to one, semi-NMF reduces to k-means [9].

Remark 6. 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 3.1, 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 3.6. 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 3.3, 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., [14].

THEOREM 3.6. *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 an approximation of M such that*

$$||M(:,j) - X(:,j)||_2 \quad < \quad M(:,j)^T z \quad \text{for all } j \text{ such that } M(:,j) \neq 0,$$

and $X(:,j) = 0$ whenever $M(:,j) = 0$ (which is optimal and does not influence the rank of X). 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 $M(:,j) = 0$ implies $X(:,j) = 0$, hence, the result follows from Theorem 3.1. ∎

Note that, for X being the best rank-k approximation of M, we have for all j that $\|M(:,j) - X(:,j)\|_2 \leq \sigma_{k+1}$, where σ_{k+1} is the $(k+1)$th singular value of M. Hence the smaller σ_{k+1} is, the more likely it is for X to be semi-nonnegative. This also means that the larger k 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 and ill-posedness 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

$$\min_{u \in \mathbb{R}^m, v \in \mathbb{R}^n} ||M - uv^T||_F^2 \quad \text{such that } v \geq 0. \tag{4.1}$$

THEOREM 4.1. *Rank-one semi-NMF* (4.1) *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,

$$\begin{aligned} ||M - uv||_F^2 &= ||M||_F^2 - 2u^T Mv + ||uv^T||_F^2 \\ &= ||M||_F^2 - 2v^T M^T Mv + ||Mv||_2^2 ||v||_2^2 \\ &= ||M||_F^2 - 2||Mv||_2^2 + ||Mv||_2^2 \\ &= ||M||_F^2 - ||Mv||_2^2 = ||M||_F^2 - v^T (M^T M)v, \end{aligned}$$

hence, (4.1) is equivalent to

$$\max_{v \in \mathbb{R}^n} v^T (M^T M)v \quad \text{such that} \quad v \geq 0 \text{ and } ||v||_2 = 1. \tag{4.2}$$

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 N. D. Stein on Mathoverflow.net,[4] the problem (4.2) is equivalent to

$$\max_{v \in \mathbb{R}^n} v^T Bv \quad \text{such that} \quad v \geq 0 \text{ and } ||v||_2 = 1, \tag{4.3}$$

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

$$v^T \left(B - \lambda_{\min}(B) I_n\right) v = v^T Bv - \lambda_{\min}(B).$$

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

$$\min_{v \geq 0, ||v||_2 = 1} v^T Cv$$

is nonnegative, is co-NP-complete [22] (a decision problem is co-NP-complete if it is a member of co-NP–, its complement is in NP–, and any problem in co-NP can be reduced to it in polynomial time; see [1] for more details). Since this problem can be solved using (4.3) with $B = -C$, this implies that rank-one semi-NMF (4.1) is NP-hard. ∎

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 [26]. 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 [13, 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.

[4] See http://mathoverflow.net/questions/48843/non-negative-quadratic-maximization.

Let us now show that semi-NMF is not always a well-posed problem (this question was raised by M. Hanafi in a personal communication), that is, that an optimal solution of (1.1) does not always exist. Here is a simple example:

$$M = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The columns of M belong to the same two-dimensional half-space $\{x \in \mathbb{R}^2 \mid x_2 \geq 0\}$. However, the first two columns are on the boundary of that half-space. Therefore, they are not contained in its interior, hence, $\text{rank}_s(M) = 3$ (Theorem 3.1). However, the infimum of (1.1) for $r = 2$ is equal to zero, taking

$$U = \begin{pmatrix} 1 & -1 \\ \delta & \delta \end{pmatrix} \text{ and } V = \begin{pmatrix} 1 & -1 & (2\delta)^{-1} \\ 0 & 0 & (2\delta)^{-1} \end{pmatrix}, \text{ with } UV = \begin{pmatrix} 1 & -1 & 0 \\ \delta & \delta & 1 \end{pmatrix}$$

and making δ tend to zero.

Note that it is not likely for semi-NMF problems to be ill-posed; this only happens when the best cone approximating the columns of M does not exist as it should be a half-space.

5. Numerical experiments. In this section, we compare four strategies to initialize Algorithm 1 (which only requires the matrix V as an input):

1. *Random initialization (RD)*: each entry of V is generated following the uniform distribution in the interval $[0, 1]$, that is, `V = rand(r,n)` in MATLAB notation (which we will reuse in the following).
2. *K-means (KM)*: V is taken as the binary cluster indicator matrix generated by k-means ($V_{kj} = 1$ if and only if the jth column of M belongs to the kth cluster) to which is added the constant[5] 0.2. This is the initialization from [9], although we do not use their algorithm to update U and V because Algorithm 1 is numerically more stable and has much better convergence properties; see Remark 2.
3. *Algorithm* 2 (A2): we use Algorithm 2 to initialize V. This initialization guarantees the error to be same as the error of the best rank-$(r-1)$ approximation.
4. *Algorithm* 3 (A3): 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^TM irreducible).

In the following two 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 four different initializations and consider the error obtained after 10 and 100 iterations. We use the notation RD/10 (resp., RD/100) to refer to the algorithm that performs 10 (resp., 100) iterations of Algorithm 1 using RD as an initialization. We use the same notation for the three other initializations KM, A2 and A3, namely, KM/10 and KM/100, A2/10 and A2/100, and A3/10 and A3/100. We also test the different initialization strategies on real data in section 5.3.

[5] A priori, because we use Algorithm 1 and not the multiplicative updates of [9], we do not have to add a constant to V. However, we observed that it allows Algorithm 1 to converge faster, and to better stationary points. In fact, taking V as the binary cluster indicator matrix seems to induce some kind of locking phenomenon as Algorithm 1 has difficulties getting away from this initial point.

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,

$$\text{quality}(U,V) \quad = \quad 100\left(\frac{||M-UV||_F}{||M-X||_F}-1\right) \quad \geq \quad 0, \tag{5.1}$$

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/. All tests are preformed using MATLAB on a laptop Intel CORE i5-3210M CPU @2.5 GHz 6GB RAM. We use the function `linprog` of MATLAB to solve the linear systems within the bisection method implemented for problem (3.2) (we have also implemented a version using CVX [16, 15] for users' convenience—note that the solution of (3.2) in y is nonunique 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 A3 computes optimal solutions for nonnegative matrices (given that M^TM is irreducible; see Corollary 3.5), and for many semi-nonnegative matrices (under a certain condition; see Theorem 3.6), we generate matrices as follows:

1. *Nonnegative matrices.* We generate each entry of M with the uniform distribution in the interval [0,1], that is, we use `M = 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 = randn(m,k)*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 (5.1) (we perform a single initialization for each generated matrix). These results confirm that A3 performs perfectly for these types of matrices. Note that (i) A2 performs relatively poorly and leads to solutions worse than RD/100 and KM/100, and (ii) RD/100 (resp., KM/100) do not always generate solutions close to optimality, in particular, for semi-nonnegative matrices when $r = 80$ for which the average quality is 5.5% (resp., 1.9%).

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 = randn(m,r)*rand(r,n)`, similarly as in the previous subsection. Then we compute the average of the absolute values 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 $N = \delta\, x_M$ `randn(m,n)`, where δ 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 = randn(m,n)`. Figure 2 displays the box plots of the measure defined in (5.1) for different values of δ. 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:

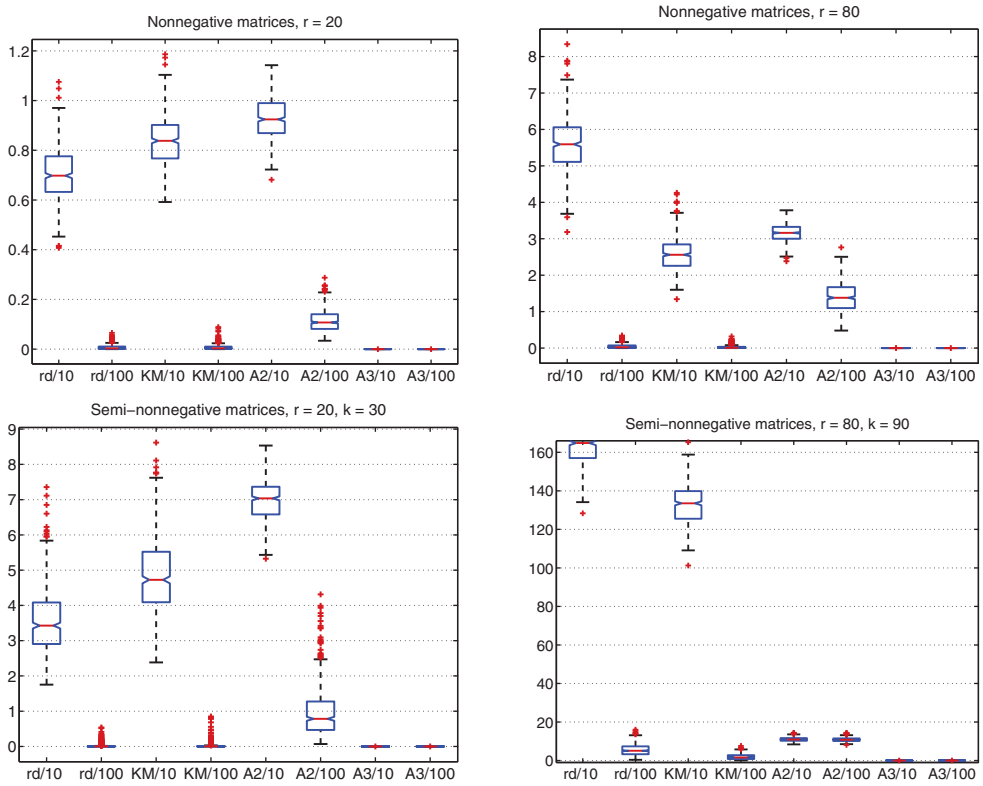

FIG. 1. *Box plots of the error defined in* (5.1) *for nonnegative and semi-nonnegative matrices.*

- A2 performs quite poorly: it is always dominated by RD/100 and KM/100. Hence, although A2 is appealing from a theoretical point of view, it does not seem to have much practical use.
- KM performs on average slightly better than RD, hence KM initialization seems beneficial in some cases (this will not be the case for some real data sets tested in section 5.3).
- Even for a relatively large noise level (in particular, $\delta = 5$), A3 performs perfectly (all semi-NMF have quality (5.1) smaller than 10^{-2}) because the best rank-r approximation of M is close to being semi-nonnegative. The reason why A3 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 3.6 is not (always) met and some solutions generated by A3 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 3.6 is never met and A3 fails most of the time to extract a semi-NMF whose error is close to the error of the best rank-r approximation.

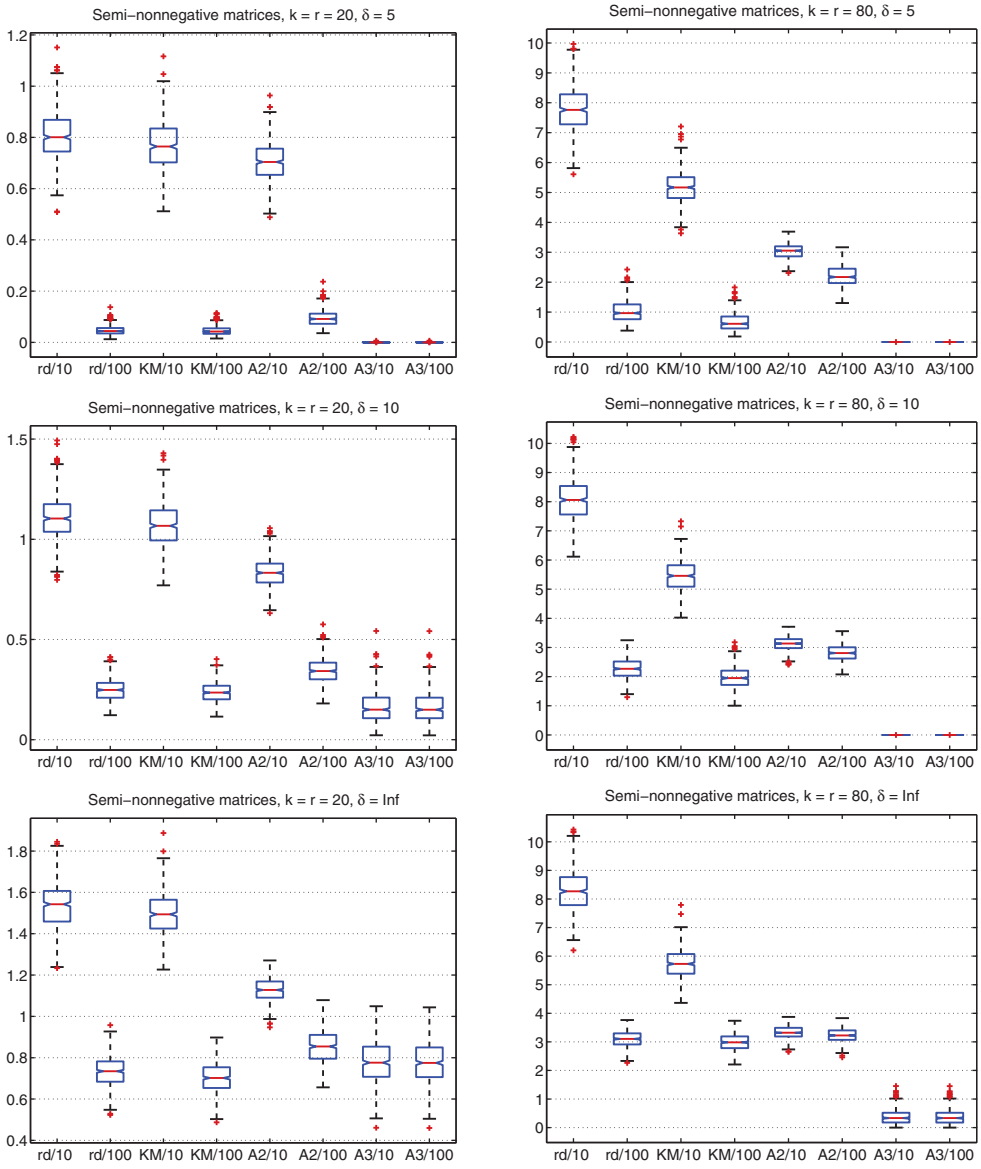

FIG. 2. *Box plots of the error from* (5.1)*: semi-nonnegative matrices with noise* ($\delta = 5, 10, +\infty$).

This is not surprising as these matrices are not likely to have semi-nonnegative best rank-r approximations.

However, when r is large ($r = 80$), A3 outperforms RD, KM, and A2; 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, A3 provides a good initial approximation.

On the contrary, when r is small ($r = 20$), A3 performs worse than RD/100 and KM/100 although the gap between both approaches is not significant as

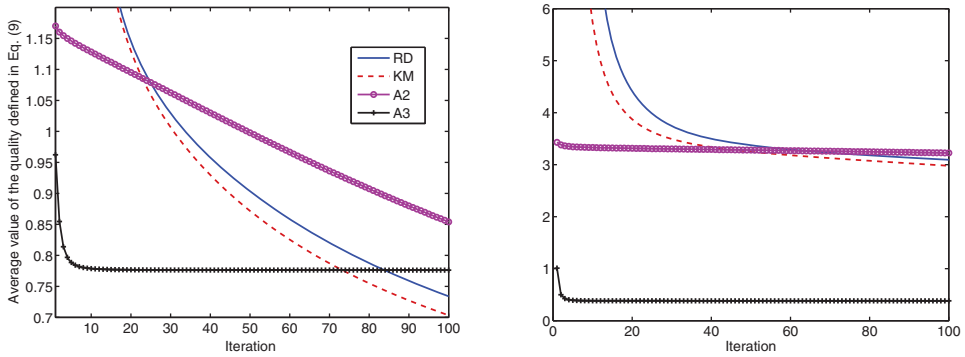

FIG. 3. *Average value of the quality* (5.1) *for each iteration of Algorithm* 1 *on the* 500 *Gaussian matrices* ($\delta = +\infty$) *for* $r = 20$ *(left) and* $r = 80$ *(right) with the different initialization strategies.*

shown in Figure 2 (bottom left); see also Figure 3. (Note that, with $r = 10$, A3 performs even worse compared to RD/100 and KM/100; see also section 5.3 for an example of this situation on real data.)

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 A3 to perform well. (At least, the row space cannot get further away from the nonnegative orthant as it is expanded as r increases.) This will be confirmed on real data in section 5.3.

- It seems that A3 generates matrices close to stationary points of (1.1) as the difference between the box plots of A3/10 and A3/100 is small on all these examples. This is another advantage of A3: 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 (5.1) for each iteration on the 500 Gaussian matrices. This observation will be confirmed on real data in section 5.3.

Finally, the two main recommendations that we can give based on these experiments are the following:

1. A3 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. RD and KM sometimes work better than A3 (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 (in fact, if the best rank-r approximation would be semi-nonnegative, A3 would perform perfectly). They should be tried on matrices for which A3 was not able to compute a semi-NMF whose error is close to the error of the best rank-r approximation.

Remark 7. 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 3.6). For example, running exactly the same experiment with $m = 200, n = 100, r = 20$, and $\delta = 10$, 86% of the solutions

TABLE 1
Numerical results on real-world data sets: quality (5.1) *of the different initialization approaches for semi-NMF.*

	CBCL	Ion 3	Ion 5	Ion 10	Wave 3	Wave 5	Wave 10
RD/10 (average)	17.59	0.59	1.90	3.12	0.19	2.43	3.44
RD/10 (best)	16.73	0.47	1.49	2.63	0.15	1.96	2.90
KM/10 (average)	27.44	0.54	2.11	4.22	0.23	5.40	14.87
KM/10 (best)	26.13	**0.41**	1.99	2.68	0.23	5.24	13.88
A2/10	1.18	0.63	3.04	3.65	0.56	2.09	3.50
A3/10	**0**	0.67	**0.38**	**0**	**0**	**0**	**0**
RD/100 (average)	1.59	0.16	0.44	0.37	0.01	0.03	0.07
RD/100 (best)	1.34	**0.15**	**0.29**	0.33	0.01	0.01	0.03
KM/100 (average)	6.00	0.16	0.98	0.44	0.01	0.09	0.13
KM/100 (best)	5.61	**0.15**	0.31	0.39	0.01	0.05	0.06
A2/100	1.18	0.16	0.99	1.57	0.02	0.14	0.25
A3/100	**0**	0.67	0.38	**0**	**0**	**0**	**0**

generated by A3 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).

5.3. Real data. In this section, we compare the different approaches on three data sets:

- CBCL face data set: it is arguably the most popular data set for NMF as it was used in the foundational paper of Lee and Seung [18] with $r = 49$; see http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html. It consists in 2429 facial images, 19-by-19 pixels each. The corresponding matrix M therefore has size 361 by 2429 and is nonnegative. This will illustrate the optimality of A3 on nonnegative data.
- Ionosphere and Waveform UCI data sets: these are the two data sets that contain both positive and negative entries used in [9]. Ionosphere corresponds to a 34-by-351 matrix with values in the interval $[-1, 1]$, Waveform to a 22-by-5000 matrix with values in the interval $[-4.2, 9.06]$ and we use $r = 3, 5, 10$ for both data sets; see https://archive.ics.uci.edu/ml/datasets.html for all the details.

Our goal is to illustrate, on real data, the observations made on synthetic data sets. Again we compare the four semi-NMF initializations (RD, KM, A2, and A3) combined with Algorithm 1 in terms of the quality measure defined in (5.1). For RD and KM, we use ten initializations (KM generates, in most cases, different solutions for different runs) and report both the average quality and the best quality obtained by the different runs. Table 1 reports the error after 10 and 100 iterations of the four approaches (as before).

Figure 4 shows the evolution of the quality for the CBCL data set, Figure 5 for the Ionosphere data set, and Figure 6 for the Waveform data set.

Interestingly, these results confirm the observations on synthetic data sets:

- For the CBCL data set (a nonnegative matrix), A3 identifies an optimal solution while the other approaches are not able to (although performing more iterations of Algorithm 1 would improve their solutions; see Figure 4).
- For the Ionosphere data set, when r is small ($r = 3$), A3 is not able to identify a good initial point and performs the worse (as for Gaussian matrices with $r = 20$; see Figure 3). When r is large ($r = 10$), A3 is again the only approach that leads to an optimal solution matching the error of the best

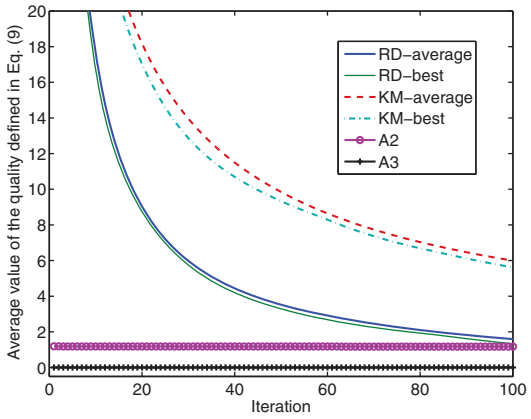

FIG. 4. *Quality* (5.1) *for each iteration of Algorithm* 1 *with the different initialization strategies on the CBCL data set* ($r = 49$).

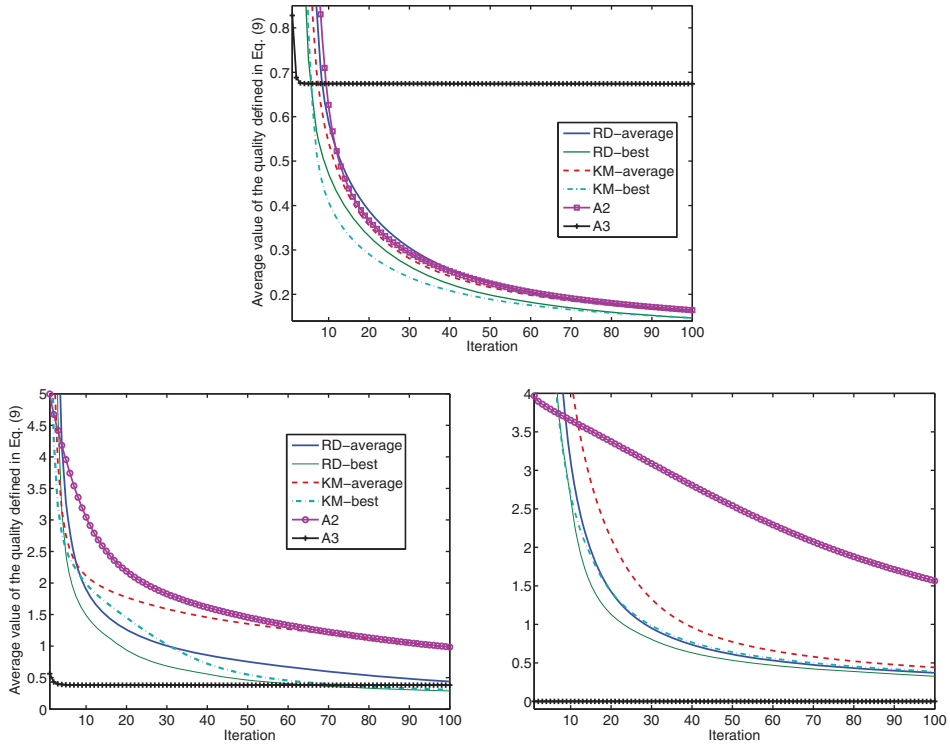

FIG. 5. *Quality* (5.1) *for each iteration of Algorithm* 1 *with the different initialization strategies for the Ionosphere data set:* $r = 3$ *(top),* $r = 5$ *(bottom left), and* $r = 10$ *(bottom right).*

rank-r approximation. For $r = 5$, it does not perform best, but allows us to obtain a rather good initial point (A3/10 performs best).

- For the Wave data set, A3 is always able to identify an optimal solution, because its best rank-r approximation is semi-nonnegative (recall that if it is semi-nonnegative for some r, it is for all $r' \geq r$; see section 3.2).

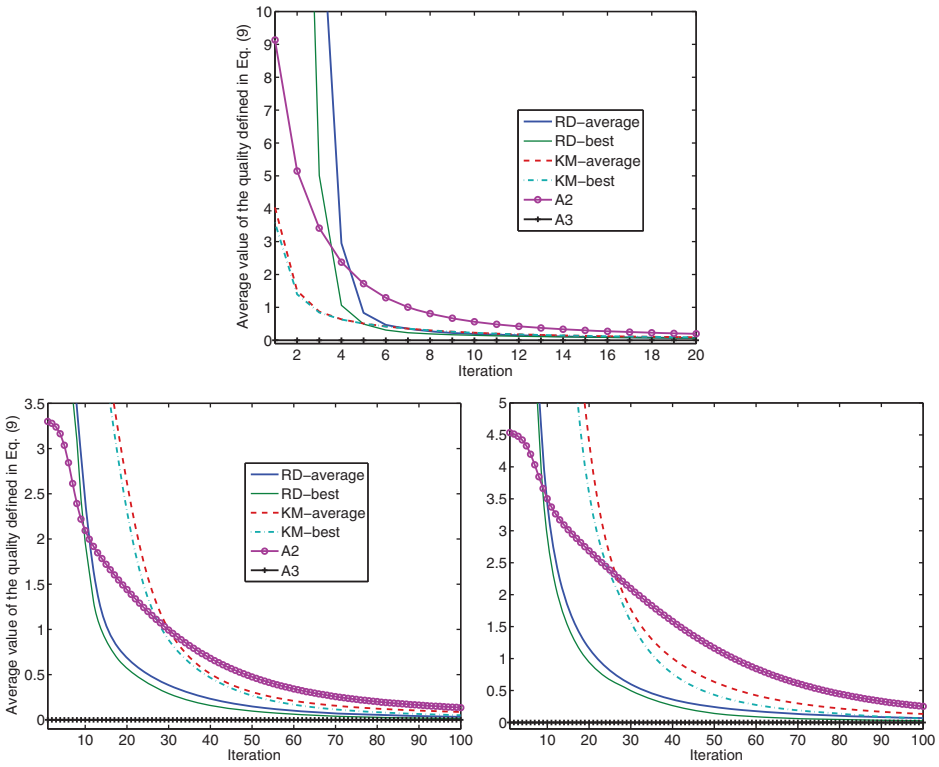

FIG. 6. *Quality* (5.1) *for each iteration of Algorithm* 1 *with the different initialization strategies for the Waveform data set:* $r = 3$ *(top),* $r = 5$ *(bottom left), and* $r = 10$ *(bottom right).*

- A3 allows Algorithm 1 to converge very quickly, in all cases in less than 10 iterations.

It is interesting to note that, for these experiments, RD performs better than KM although the difference is not significant (except for the CBCL face data set).

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 threefold. We showed the following:

- 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 equal to the error of the best rank-$(r-1)$ approximation; see Theorem 2.1 and 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 3.2), 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 already NP-hard, in general, in the rank-one case (Theorem 4.1). Moreover, we showed that some semi-NMF instances are ill-posed (that is, an optimal solution does not exist).

Further research on semi-NMF includes the design of other initialization strategies (in particular, in the case r is small and the best rank-r approximation of M is far from being semi-nonnegative; see also Remark 5), 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 3.1 and in [9] for more details. In particular, it would be interesting to see how Algorithm 3 performs as an initialization strategy in that case.

Acknowledgment. The authors would like to thank the reviewers and the editor for their insightful comments which helped improve the paper.

REFERENCES

[1] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, Cambridge, 2009.
[2] D.P. Bertsekas, *Nonlinear Programming:* 2*nd ed.*, Athena Scientific, Belmont, MA, 1999.
[3] D.P. Bertsekas, *Corrections for the Book Nonlinear Programming (*2*nd ed.)*, http://www.athenasc.com/nlperrata.pdf (2014).
[4] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L.A. Morel, *Neighbor embedding based single-image super-resolution using semi-nonnegative matrix factorization*, in Proceedings of the IEEE International Conferemce on Acoustics, Speech and Signal Processing (ICASSP), IEEE, Piscataway, NJ, 2012, pp. 1289–1292.
[5] L. Blum, *Computing over the reals: Where Turing meets Newton*, Notices Amer. Math. Soc., 51 (2004), pp. 1024–1034.
[6] R. Bro, E. Acar, and T.G. Kolda, *Resolving the sign ambiguity in the singular value decomposition*, J. Chemometrics, 22 (2008), pp. 135–140.
[7] M. Chouh, , M. Hanafi, and K. Boukhetala, *Semi-nonnegative rank for real matrices and its connection to the usual rank*, Linear Algebra Appl., 466 (2015), pp. 27–37.
[8] A. Cichocki and A.H. Phan, *Fast local algorithms for large scale nonnegative matrix and tensor factorizations*, IEICE Trans. Fundam. Electron., Vol. E92-A, (2009), pp. 708–721.
[9] C. Ding, T. Li, and M.I. Jordan, *Convex and semi-nonnegative matrix factorizations*, IEEE Trans. Pattern Anal. Mach. Intell., 32 (2010), pp. 45–55.
[10] J. Edmonds, *Systems of distinct representatives and linear algebra*, J. Res. Nat. Bur. Standards Sect. B, 71 (1967), pp. 241–245.
[11] N. Gillis and F. Glineur, *Accelerated multiplicative updates and hierarchical* ALS *algorithms for nonnegative matrix factorization*, Neural Comput., 24 (2012), pp. 1085–1105.
[12] N. Gillis and F. Glineur, *On the geometric interpretation of the nonnegative rank*, Linear Algebra Appl., 437 (2012), pp. 2685–2712.
[13] N. Gillis and F. Glineur, *A continuous characterization of the maximum-edge biclique problem*, J. Global Optim., 58 (2014), pp. 439–464.
[14] G.H. Golub and C.F. Van Loan, *Matrix Computation*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.
[15] M. Grant and S. Boyd, *Graph implementations for nonsmooth convex programs*, in Recent Advances in Learning and Control (a Tribute to M. Vidyasagar), V. Blondel, S. Boyd, and H. Kimura, eds., Lecture Notes in Control and Inform. Sci., Springer, London, 2008, pp. 95–110.
[16] M. Grant and S. Boyd, *CVX: MATLAB Software for Disciplined Convex Programming, Version* 1.21, http://cvxr.com/cvx/ (2011).
[17] C.-J. Hsieh and I.S. Dhillon, *Fast coordinate descent methods with variable selection for nonnegative matrix factorization*, in Proceedings of the 17th ACM International Conference on Knowledge Discovery and Data Mining, ACM, New York, 2011, pp. 1064–1072.
[18] D.D. Lee and H.S. Seung, *Learning the parts of objects by nonnegative matrix factorization*, Nature, 401 (1999), pp. 788–791.
[19] L. Li and Y.-J. Zhang, *FastNMF: Highly efficient monotonic fixed-point nonnegative matrix factorization algorithm with good applicability*, J. Electron. Imaging, 18 (2009), 033004.
[20] J. Liu, J. Liu, P. Wonka, and J. Ye, *Sparse non-negative tensor factorization using columnwise coordinate descent*, Pattern Recognit., 45 (2012), pp. 649–656.

[21] Q. Mo and B.A. Draper, *Semi-nonnegative matrix factorization for motion segmentation with missing data*, in Computer Vision–ECCV 2012, Springer, Berlin, 2012, pp. 402–415.
[22] K.G. Murty and S.N. Kabadi, *Some NP-complete problems in quadratic and nonlinear programming*, Math. Program., 39 (1987), pp. 117–129.
[23] L.N. Trefethen and D. Bau, III, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.
[24] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B.W. Schuller, *A deep semi-NMF model for learning hidden representations*, in International Conference on Machine Learning (ICML '14), 2014, pp. 1692–1700.
[25] S.A Vavasis, *Nonlinear Optimization: Complexity Issues*, Oxford University Press, New York, 1991.
[26] S.A. Vavasis, *On the complexity of nonnegative matrix factorization*, SIAM J. Optim., 20 (2010), pp. 1364–1377.
[27] N. Yokoya, J. Chanussot, and A. Iwasaki, *Generalized bilinear model based nonlinear unmixing using semi-nonnegative matrix factorization*, in IEEE International Geoscience and Remote Sensing Symposium (IGARSS), IEEE, Piscataway, NJ, 2012, pp. 1365–1368.