CONIC OPTIMIZATION-BASED ALGORITHMS FOR NONNEGATIVE MATRIX FACTORIZATION

Valentin Leplat, Yurii Nesterov, Nicolas Gillis, François Glineur

REPRINT | 3254

UCLouvain

CORE | LIDAM
Louvain Institute of Data Analysis and Modeling in economics and statistics

CORE

Voie du Roman Pays 34, L1.03.01

B-1348 Louvain-la-Neuve

Tel (32 10) 47 43 04

Email:lidam-library@uclouvain.be

https://uclouvain.be/en/research-institutes/lidam/core/core-reprints.html

Conic-Optimization Based Algorithms for Nonnegative Matrix Factorization

Valentin Leplat[a], Yurii Nesterov[b], Nicolas Gillis[c], François Glineur[b]

[a]Center for Artificial Intelligence Technology, Skoltech, Bolshoy Boulevard 30, bld. 1, Moscow, Russia 121205; [b]CORE and ICTEAM/Mathematical Engineering (INMA), UCLouvain, Avenue Georges Lemaître 4, B-1348 Louvain-la-Neuve, Belgium ; [c]Department of Mathematics and Operational Research, Faculté Polytechnique, Université de Mons, Rue de Houdain 9, 7000 Mons, Belgium

ARTICLE HISTORY
Compiled January 25, 2023

ABSTRACT
Nonnegative matrix factorization is the following problem: given a nonnegative input matrix V and a factorization rank K, compute two nonnegative matrices, W with K columns and H with K rows, such that WH approximates V as well as possible. In this paper, we propose two new approaches for computing high-quality NMF solutions using conic optimization. These approaches rely on the same two steps. First, we reformulate NMF as minimizing a concave function over a product of convex cones–one approach is based on the exponential cone, and the other on the second-order cone. Then, we solve these reformulations iteratively: at each step, we minimize exactly, over the feasible set, a majorization of the objective functions obtained via linearization at the current iterate. Hence these subproblems are convex conic programs and can be solved efficiently using dedicated algorithms. We prove that our approaches reach a stationary point with an accuracy decreasing as $\mathcal{O}(\frac{1}{i})$, where i denotes the iteration number. To the best of our knowledge, our analysis is the first to provide a convergence rate to stationary points for NMF. Furthermore, in the particular cases of rank-one factorizations (that is, $K = 1$), we show that one of our formulations can be expressed as a convex optimization problem implying that optimal rank-one approximations can be computed efficiently. Finally, we show on several numerical examples that our approaches are able to frequently compute exact NMFs (that is, with $V = WH$), and compete favorably with the state of the art.

KEYWORDS
nonnegative matrix factorization, nonnegative rank, exponential cone, second-order cone, concave minimization, conic optimization, Frank-Wolfe gap, convergence to stationary points

VL acknowledges the support by the European Research Council (ERC Advanced Grant no 788368) and the support by Ministry of Science and Higher Education grant No. 075-10-2021-068. Email: V.Leplat@skoltech.ru

NG acknowledges the support by the Fonds de la Recherche Scientifique - FNRS and the Fonds Wetenschappelijk Onderzoek - Vlanderen (FWO) under EOS Project no O005318F-RG47, by the European Research Council (ERC Starting Grant no 679515), and by the Francqui Foundation. Email: nicolas.gillis@umons.ac.be

YN and FG acknowledge support from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 788368). FG also acknowledges support from EOS project no O005318F-RG47. Emails: {yurii.nesterov,francois.glineur}@uclouvain.be.

1. Introduction

Nonnegative matrix factorization (NMF) is the problem of approximating a given nonnegative matrix, $V \in \mathbb{R}_+^{F\times N}$, as the product of two smaller nonnegative matrices, $W \in \mathbb{R}_+^{F\times K}$ and $H \in \mathbb{R}_+^{K\times N}$, where K is a given parameter known as the factorization rank. One aims at finding the best approximation, that is, the one that minimizes the discrepancy between V and the product WH, often measured by the Frobenius norm of their difference, $\|V - WH\|_F$. Despite the fact that NMF is NP-hard in general [30] (see also [27]), it has been used successfully in many domains such as probability, geoscience, medical imaging, computational geometry, combinatorial optimization, analytical chemistry, and machine learning; see [11, 13] and the references therein. Many local optimization schemes have been developed to compute NMFs. They aim to identify local minima or stationary points of optimization problems that minimize the discrepancy between V and the approximation WH. Most of these iterative algorithms rely on a two-block coordinate descent scheme that consists in (approximatively) optimizing alternatively over W with H fixed, and vice-versa; see [5, 13] and the references therein. In this paper, we are interested in computing high-quality local minima for the NMF optimization problems without relying on the block coordinate descent (BCD) framework. We will perform the optimization over W and H jointly. Moreover, our focus is on finding exact NMFs, that is, computing nonnegative factors W and H such that $V = WH$, although our approaches can be used to find approximate NMFs as well.

The minimum factorization rank K for which an exact NMF exists is called the nonnegative rank of V and is denoted $\operatorname{rank}_+(V)$, we have

$$\operatorname{rank}_+(V) = \min\Big\{K \in \mathbb{N} \,\Big|\, \text{there exist } W \in \mathbb{R}_+^{F\times K} \text{ and } H \in \mathbb{R}_+^{K\times N} \text{ such that } V = WH\Big\}.$$

The computation of the nonnegative rank is NP-hard [30], and is a research topic on its own; see [13, Chapter 3] [8, 9] and the references therein for recent progress on this question.

1.1. Computational complexity

Solving exact NMF can be used to compute the nonnegative rank, by finding the smallest K such that an exact NMF exists. Cohen and Rothblum [7] give a super-exponential time algorithm for this problem. Vavasis [30] proved that checking whether $\operatorname{rank}(V) = \operatorname{rank}_+(V)$, where $\operatorname{rank}(V) = K$ is part of the input, is NP-hard. Since determining the nonnnegative rank is a generalization of exact NMF, the results in [30] imply that computing an exact NMF is also NP-hard. Similarly, the standard NMF problem using any norm is a generalization of exact NMF, and therefore any hardness result that applies to exact NMF also applies to most approximated NMF models [30]. Hence, unless **P**=**NP**, no algorithm can solve exact NMF using a number of arithmetic operations bounded by a polynomial in K and in the size of V; see also [27] that gives a different proof using algebraic arguments.

More recently, Arora et al. [2] showed that no algorithm to solve this problem can run in time $(FN)^{o(K)}$ unless[1] 3-SAT can be solved in time $2^{o(n)}$ on instances with n

[1] 3-SAT, or 3-satisfiability, is an instrumental problem in computational complexity to prove NP-completeness results. 3-SAT is the problem of deciding whether a set of disjunctions containing 3 Boolean variables or their negation can be satisfied.

variables. However, in practice, K is small and it makes senses to wonder what is the complexity if K is assumed to be a fixed constant. In that case, they showed that exact NMF can be solved in polynomial time in F and N, namely in time $\mathcal{O}((FN)^{c2^K K^2})$ for some constant c, which Moitra [24] later improved to $\mathcal{O}((FN)^{cK^2})$.

The argument is based on quantifier elimination theory, using the seminal result by Basu, Pollack and Roy [3]. Unfortunately, this approach cannot be used in practice, even for small size matrices, because of its high computational cost: although the term $\mathcal{O}((FN)^{cK^2})$ is a polynomial in F and N for K fixed, it grows extremely fast (and the hidden constants are usually very large). Let us illustrate with a 4-by-4 matrix with $K = 3$, we have a complexity of order $16^9 \approx 7\ 10^{10}$ and for a 5-by-5 matrix with $K = 4$, the complexity raises up to $25^{16} \approx 2\ 10^{22}$. Therefore developing an effective computational technique for exact NMF of small matrices is an important direction of research. Some heuristics have been recently developed that allow solving exact NMF for matrices up to a few dozen rows and columns [29].

1.2. Contribution and outline of the paper

In this paper, we introduce two formulations for computing an NMF using conic optimization. They rely on the same two steps. First, in Section 2, we reformulate NMF as minimizing a concave function over a product of convex cones; one approach is based on the exponential cone and leads to under-approximations, and the other on the second-order cone and leads to over-approximations. For the latter formulation, in the case of a rank-one factorization, we show that it can be cast as a convex optimization problem, leading to an efficient computation of the optimal rank-one over-approximation. Then, in Section 3, we solve these reformulations iteratively: at each step, we minimize exactly over the feasible set a majorization of the objective functions obtained via linearization at the current iterate. Hence these subproblems are convex conic programs and can be solved efficiently using dedicated algorithms. In Section 4, we show that our optimization scheme relying on successive linearizations is a special case of the Frank-Wolfe (FW) algorithm. By using an appropriate measure of stationarity, namely the FW gap, we show in Theorem 4.1 that the minimal FW gap generated by our algorithm converges as $\mathcal{O}(\frac{1}{i})$, where i is the iteration index. Finally, in Section 5, we use our approaches to compute exact NMFs, and show that they compete favorably with the state of the art when applied to several classes of nonnegative matrices; namely, randomly generated, infinitesimally rigid and slack matrices.

Remark 1 (Focus on exact NMF)**.** Our two NMF formulations can be used to compute approximate factorizations (namely, under- and over-approximations). However, in this paper, we focus on exact NMF for the numerical experiments. The reason is twofold:

(1) Exact NMF problems allow us to guarantee whether a globally optimal solution is reached, and hence compare algorithms in terms of global optimality.
(2) Our current algorithms rely on interior-point methods that do not scale well. Therefore, they are significantly slower, at this point, than state-of-the-art algorithms to compute approximate NMFs on large data sets (such as images or documents). Making our approach scalable is a topic of further research. Moreover, because of the under/over-approximations, the error obtained with the proposed algorithms would be larger, and the comparison would not be fair.

Here is a simple example, let

$$V = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, V_{\text{NMF}} = \begin{pmatrix} 0.45 & 0.72 \\ 0.72 & 1.17 \end{pmatrix}, V_{\text{U}} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}, V_{\text{O}} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

The best rank-one NMF of V is V_{NMF} (with two digits of accuracy) with Frobenius error $\|V - V_{\text{NMF}}\|_F = 0.62$, while a best rank-one under-approximation (resp. over-approximation) of V is V_{U} (resp. V_{O}) with Frobenius error 1. (Note however that under-/over-approximations can have some useful properties in practice [16, 28].)

2. NMF formulations based on conic optimization

In this section we propose two new formulations for NMF, where the feasible set is represented using the exponential cone (Section 2.1) and the second-order cone (Section 2.2).

2.1. NMF formulation via exponential cones

Given a non-negative matrix $V \in \mathbb{R}_+^{F \times N}$ and a positive integer $K \ll \min(F, N)$, we want to compute an NMF. Our first proposed formulation is the following:

$$\begin{array}{ll} \max\limits_{W \in \mathbb{R}^{F \times K}, H \in \mathbb{R}^{K \times N}} & \sum\limits_{f=1}^{F} \sum\limits_{n=1}^{N} \left(\sum\limits_{k=1}^{K} W_{fk} H_{kn} \right) \\ \text{subject to} & \sum\limits_{k=1}^{K} W_{fk} H_{kn} \leqslant V_{fn} \text{ for } f \in \mathcal{F}, n \in \mathcal{N}, \\ & W_{fk} \geqslant 0, H_{kn} \geqslant 0 \text{ for } f \in \mathcal{F}, k \in \mathcal{K}, n \in \mathcal{N}. \end{array} \tag{1}$$

where $\mathcal{F} = \{1, ..., F\}$, $\mathcal{N} = \{1, ..., N\}$ and $\mathcal{K} = \{1, ..., K\}$. Any feasible solution (W, H) of (1) provides an under-approximation of V, because of the elementwise constraint $WH \leqslant V$. The objective function of (1) maximizes the sum of the entries of WH. Therefore, if V admits an exact NMF of size K, that is, $\text{rank}_+(V) \leqslant K$, any optimal solution (W^*, H^*) of (1) must satisfy $W^*H^* = V$, and hence will provide an exact NMF of V. Note that this problem is nonconvex because of the bilinear terms appearing in the objective and the constraint $WH \leqslant V$.

Let us now reformulate (1) using exponential cones. In order to deal with nonnegativity constraints on the entries of W and H, we use the following change of variables: $W_{fk} = G(U_{fk}) = e^{U_{fk}}$ and $H_{kn} = G(T_{kn}) = e^{T_{kn}}$, where $U \in \mathbb{R}^{F \times K}$ and $T \in \mathbb{R}^{K \times N}$, with $f = 1, \ldots, F$, $n = 1, \ldots, N$ and $k = 1, \ldots, K$ and $G(t) = e^t$. By applying a logarithm on top of this change of variables to the objective function, and on both sides of the inequality constraints $WH \leqslant V$, (1) can be nearly equivalently rewritten

as follows, the difference being that zero elements in W and H are now excluded:

$$\begin{aligned} \max_{U\in\mathbb{R}^{F\times K},T\in\mathbb{R}^{K\times N}} \quad & \log\left(\sum_{f,n,k} e^{U_{fk}+T_{kn}}\right) \\ \text{subject to} \quad & \log\left(\sum_{k=1}^{K} e^{U_{fk}+T_{kn}}\right) \leqslant \log\left(V_{fn}\right) \text{ for } f\in\mathcal{F}, n\in\mathcal{N}, \end{aligned} \tag{2}$$

which corresponds to the maximization of a convex function (logarithm of the sums of exponentials) over a convex set, each constraint being convex for the same reason.

We rewrite the convex feasible set of (2) with explicit conic constraints as follows:

$$\begin{aligned} &\sum_{k=1}^{K} t_{fkn} \leqslant V_{fn} \text{ for } f\in\mathcal{F}, n\in\mathcal{N}, \\ &(t_{fkn}, 1, U_{fk}+T_{kn}) \in K_{exp} \text{ for } f\in\mathcal{F}, k\in\mathcal{K}, n\in\mathcal{N}, \end{aligned} \tag{3}$$

where $K_{exp}\subset\mathbb{R}^3$ denotes the (primal) exponential cone defined as:

$$K_{exp} = \left\{(x_1,x_2,x_3)\in\mathbb{R}^3 | x_1\geqslant x_2 e^{\frac{x_3}{x_2}}, x_2>0\right\} \cup \left\{(x_1,0,x_3)\,|x_1\geqslant 0, x_3\leqslant 0\right\}. \tag{4}$$

Note that the exponential cone is closed and includes the subset $\{(x_1,0,x_3)\,|x_1\geqslant 0, x_3\leqslant 0\}$, therefore the scenarios for which the entries V_{fn} are equal to zero can be handled by exponential conic constraints, which was not possible with formulation (2) since the log function is not defined at zero. Hence the optimization problem (1) can be written completely equivalently as

$$\begin{aligned} \max_{U\in\mathbb{R}^{F\times K},T\in\mathbb{R}^{K\times N},t\in\mathbb{R}^{F\times K\times N}} \quad & \log\left(\sum_{f,n,k} e^{U_{fk}+T_{kn}}\right) \\ \text{subject to} \quad & \sum_{k=1}^{K} t_{fkn} \leqslant V_{fn} \text{ for } f\in\mathcal{F}, n\in\mathcal{N}, \\ & (t_{fkn}, 1, U_{fk}+T_{kn}) \in K_{exp} \text{ for } f\in\mathcal{F}, k\in\mathcal{K}, n\in\mathcal{N}. \end{aligned} \tag{5}$$

This leads to $F\times N$ inequality constraints and the introduction of $F\times K\times N$ exponential cones. In Section 3, we propose an algorithm to tackle (5) using successive linearizations of the objective function.

2.2. NMF formulation via rotated second-order cones

Our second proposed NMF formulation is the following:

$$\begin{aligned} \min_{W\in\mathbb{R}^{F\times K},H\in\mathbb{R}^{K\times N}} \quad & \sum_{f=1}^{F}\sum_{n=1}^{N}\left(\sum_{k=1}^{K} W_{fk}H_{kn}\right) \\ \text{subject to} \quad & \sum_{k=1}^{K} W_{fk}H_{kn} \geqslant V_{fn} \text{ for } f\in\mathcal{F}, n\in\mathcal{N}, \\ & W_{fk}, H_{kn} \geqslant 0 \text{ for } f\in\mathcal{F}, k\in\mathcal{K}, n\in\mathcal{N}. \end{aligned} \tag{6}$$

Any feasible solution (W, H) of (6) provides an over-approximation of V, because of the constraint $WH \geqslant V$. The objective function of (1) minimizes the sum of the entries of WH. Therefore, if $\text{rank}_+(V) \leqslant K$, any optimal solution (W^*, H^*) of (1) must satisfy $W^*H^* = V$, and hence will provide an exact NMF of V. Again the problem is nonconvex due to the bilinear terms.

Let us use the following change of variables: we let $W_{fk} = G(U_{fk}) = \sqrt{U_{fk}}$ and $H_{kn} = G(T_{kn}) = \sqrt{T_{kn}}$ where $U \in \mathbb{R}_+^{F\times K}$ and $T \in \mathbb{R}_+^{K\times N}$, with $f = 1,\ldots,F$, $n = 1,\ldots,N$ and $k = 1,\ldots,K$, this time with $G(t) = \sqrt{t}$. Thus the optimization problem (6) can be equivalently rewritten as:

$$\begin{aligned} \min_{U\in\mathbb{R}_+^{F\times K},T\in\mathbb{R}_+^{K\times N}} \quad & \sum_{f=1}^{F}\sum_{n=1}^{N}\left(\sum_{k=1}^{K} \sqrt{U_{fk}}\sqrt{T_{kn}}\right) \\ \text{subject to} \quad & \sum_{k=1}^{K} \sqrt{U_{fk}}\sqrt{T_{kn}} \geqslant V_{fn} \text{ for } f\in\mathcal{F}, n\in\mathcal{N}, \end{aligned} \tag{7}$$

which minimizes a concave function over a convex set. Indeed, the function $\sqrt{xy}$ is concave.

This set can be written with conic constraints as follows:

$$\begin{aligned} & \sum_{k=1}^{K} t_{fkn} \geqslant V_{fn}, \text{ for } f\in\mathcal{F}, n\in\mathcal{N}, \\ & \left(U_{fk}, \frac{1}{2}T_{kn}, t_{fkn}\right) \in \mathcal{Q}_r^3 \text{ for } f\in\mathcal{F}, k\in\mathcal{K}, n\in\mathcal{N}, \end{aligned} \tag{8}$$

where $\mathcal{Q}_r^3$ denotes the 3-dimensional rotated second-order cone defined as:

$$\mathcal{Q}_r^3 = \left\{(x_1, x_2, x_3) \in \mathbb{R}^3 \mid 2x_1x_2 \geqslant x_3^2, x_1 \geqslant 0, x_2 \geqslant 0\right\}.$$

Thus, the optimization problem (7) becomes

$$\begin{array}{ll} \min\limits_{U\in\mathbb{R}_+^{F\times K},T\in\mathbb{R}_+^{K\times N},t\in\mathbb{R}^{F\times K\times N}} & \sum\limits_{f=1}^{F}\sum\limits_{n=1}^{N}\left(\sum\limits_{k=1}^{K}\sqrt{U_{fk}}\sqrt{T_{kn}}\right) \\ \text{subject to} & \sum\limits_{k=1}^{K} t_{fkn} \geqslant V_{fn} \text{ for } f\in\mathcal{F}, n\in\mathcal{N}, \\ & \left(U_{fk}, \frac{1}{2}T_{kn}, t_{fkn}\right) \in \mathcal{Q}_r^3 \text{ for } f\in\mathcal{F}, k\in\mathcal{K}, n\in\mathcal{N}, \end{array} \tag{9}$$

which leads to $F \times N$ inequality constraints and the introduction of of $F \times K \times N$ rotated quadratic cones. In section 3, we present an algorithm to tackle (5) and (9).

2.2.1. Rank-one Nonnegative Matrix Over-approximation

In this section, we show that our over-approximation formulation (6) can be expressed as a convex optimization problem in the case of a rank-one factorization (that is, $K = 1$). Hence we will be able to compute an optimal rank-one nonnegative matrix over-approximation (NMO). For $K = 1$, (6) becomes:

$$\begin{array}{ll} \min\limits_{w\in\mathbb{R}^F,h\in\mathbb{R}^N} & \sum\limits_{f=1}^{F}\sum\limits_{n=1}^{N} w_f h_n \\ \text{subject to} & w_f h_n \geqslant V_{fn} \text{ for } f\in\mathcal{F}, n\in\mathcal{N}, \\ & w_f, h_n \geqslant 0 \text{ for } f\in\mathcal{F}, n\in\mathcal{N}. \end{array} \tag{10}$$

Any feasible solution (w, h) of (10) provides a rank-one NMO of V, because of the constraints. The objective function of (10) minimizes the sum of the entries of $wh^\top$, which is equal to $\langle wh^\top, ee^\top\rangle = \langle w, e\rangle \langle h, e\rangle$, where e denotes the all-one factor of appropriate dimension. Since any solution $wh^\top$ can be rescaled as $(\lambda w)(h^\top/\lambda)$ for any $\lambda > 0$, we can assume without loss of generality (w.l.o.g.) that $\langle w, e\rangle = 1$, and hence (10) can be equivalently written as follows:

$$\begin{array}{ll} \min\limits_{w\in\mathbb{R}^F,h\in\mathbb{R}^N} & \langle h, e\rangle \\ \text{subject to} & \langle w, e\rangle = 1, \\ & w_f h_n \geqslant V_{fn} \text{ for } f\in\mathcal{F}, n\in\mathcal{N}, \\ & w_f, h_n \geqslant 0 \text{ for } f\in\mathcal{F}, n\in\mathcal{N}. \end{array} \tag{11}$$

Then, by letting each h_n take the minimal value allowed by the constraints, that is, $h_n = \max_{f\in\mathcal{F}} V_{fn}/w_f$ for each $n \in \mathcal{N}$, and replacing w_f by its inverse, $u_f = 1/w_f$ for each $f \in \mathcal{F}$, (11) becomes:

$$\begin{array}{ll} \min\limits_{u\in\mathbb{R}^F} & \sum\limits_{n} \max_f \left(u_f V_{fn}\right) \\ \text{subject to} & \sum\limits_{f} 1/u_f \leqslant 1,\ u_f \geqslant 0 \text{ for } f\in\mathcal{F}. \end{array} \tag{12}$$

The feasible set of (12) can be formulated by using various conic constraints:

- a semi-definite programming formulation: introduce variables y_f such that $u_f y_f \geqslant 1, \sum_f y_f \leqslant 1$ to obtain

$$\begin{pmatrix} u_f & 1 \\ 1 & y_f \end{pmatrix} \in \mathbb{S}_+^2 \text{ for } f \in \mathcal{F}, \sum_f y_f \leqslant 1,$$

  where $\mathbb{S}_+^2$ denotes the set of positive semi-definite matrices of dimension 2.
- a power-cone formulation: for $p < 0$ the function $g(x) = x^p$ is convex for $x > 0$ and the inequality $z \geqslant x^p$ is equivalent to $z^{1/(1-p)} x^{-p/(1-p)} \geqslant 1 \iff (z, x, 1) \in P^{1/(1-p)}$ where $P^\alpha = \{(x, z, a) \mid z^\alpha x^{1-\alpha} \geqslant a\}$ is a power cone. In our case, by introducing $y_f \geqslant u_f^{-1}$, we obtain

$$(y_f, u_f, 1) \in P^{1/2} \text{ for } f \in \mathcal{F}, \sum_f y_f \leqslant 1.$$

- a (rotated) quadratic formulation: introducing variables y_f such that $y_f \geqslant 1/u_f$ for $u_f \geqslant 0$, this can be formulated as follows: $(u_f, y_f, \sqrt{2}) \in Q_r^3$ where Q_r^3 denotes the set of rotated quadratic cones of dimension 3. We then have :

$$(u_f, y_f, \sqrt{2}) \in Q_r^3 \text{ for } f \in \mathcal{F}, \sum_f y_f \leqslant 1.$$

In this paper, we consider the (rotated) quadratic formulation which is the easiest to implement in the MOSEK software [25].

Further, the objective function of (12) is a sum of convex piece-wise linear functions. Hence by posing $t_n \geqslant \max_f (u_f V_{fn})$, (12) can equivalently be formulated as follows:

$$\begin{aligned} \min_{t \in \mathbb{R}^N, u, y \in \mathbb{R}^F} \quad & \sum_n t_n \\ \text{subject to} \quad & \sum_f y_f \leqslant 1, \\ & (u_f, y_f, \sqrt{2}) \in Q_r^3 \text{ for } f \in \mathcal{F}, \\ & t_n \geqslant u_f V_{fn} \text{ for } f \in \mathcal{F}, n \in \mathcal{N}, \end{aligned} \tag{13}$$

which involves $2F + N$ variables and $F(1 + N) + 1$ constraints. This problem can be solved to optimality and efficiently with an interior-point method (IPM), as available for example in MOSEK [25].

For the formulation based on exponential cones, [12] showed that the rank-one underapproximation for positive input matrices can be expressed as the dual of an optimal transportation problem, and hence can also be solved optimally and efficiently with polynomial-time methods [26].

3. A successive linearization algorithm

In this section, we present an iterative algorithm to tackle problems (5) and (9). Both problems can be written as the minimization of a concave function Φ over a convex set denoted by Q. Note that Q designates either the feasible set of (5) or the feasible set of (9). We perform this minimization by solving a sequence of simpler problems in which the objective function is replaced by its linearization constructed at the current solution (U, T). Let us denote $Z^{(i)} = (U^{(i)}, T^{(i)})$ the ith iterate of our algorithm. At each iteration i, we update Z as follows:

$$\begin{aligned} Z^{(i)} &\in \underset{Z \in Q}{\operatorname{argmin}}\ \Phi(Z^{(i-1)}) + \left\langle \nabla\Phi(Z^{(i-1)}), Z - Z^{(i-1)} \right\rangle \\ &\in \underset{Z \in Q}{\operatorname{argmin}}\ \left\langle \nabla\Phi(Z^{(i-1)}), Z \right\rangle, \end{aligned} \tag{14}$$

where Φ is the objective function of (5) or (9). Since the objective of (14) is linear in Z, the subproblems become convex. Moreover they are particular structured conic optimization problems. In this paper, we use the MOSEK software [25] to solve each successive problem (14) with an IPM. Algorithm 1 summarizes our proposed method to tackle (5) and (9).

To initialize U and T, we chose to randomly initialize W and H (using the uniform distribution in the interval $[0, 1]$ for each entry of W and H) and apply the two changes of variables, $G(.)$, to compute the initializations for U and T.

In this paper, we use a tolerance for the relative error equal to 10^{-6}, that is, we assume that an exact NMF (W, H) is found for an input matrix V as soon as $\frac{\|V-WH\|_F}{\|V\|_F} \leqslant 10^{-6}$, as done in [29].

The main algorithm integrates a procedure that automatically updates the optimization problems in the case subsets of entries of the solution tend to zero. Indeed, due to numerical limitations of the solver, the required level of accuracy cannot be reached in some numerical tests even if the solution is close to convergence. This procedure is referred to as Sparsity Patterns Integration (SPI) and is detailed in Appendix B. Note that the update of the optimization problem is computationally costly, in particular the update of the matrix of coefficients defining the constraints. Hence, SPI is triggered twice; at 80% and 95% of the maximum number of iterations. This practical choice has been motivated by numerical experiments that showed that two activations in the final iterations are sufficient to reach the tolerance error when the current solution is close enough to a high-accuracy local optimum. However, for exponential cones, in some numerical tests, the relative error can be stuck in the interval $[10^{-4}, 10^{-5}]$. In this context only, a final refinement step further improves the output of the main algorithm using the state-of-the-art accelerated HALS algorithm, an exact BCD method for NMF, from [14] to go below the 10^{-6} tolerance.

In Section 4 , we discuss the convergence guarantees for Algorithm 1.

4. Convergence results

Let us focus on the following optimization problem

$$\min_{Z \in Q} \quad \Phi(Z), \tag{15}$$

Algorithm 1 Successive Conic Convex Approximation for Exact NMF

Require: Input matrix $V \in \mathbb{R}_+^{F \times N}$, the factorization rank K, number of iterations *maxiter*, choose the formulation (5) (exponential cones) or (9) (second-order cones).

Ensure: $(W, H) \geqslant 0$ such that $V \approx WH$, and $V \leqslant WH$ for (5) (under-approximation) or $V \geqslant WH$ for (9) (over-approximation).

1: *% Block 1: Initialization*
2: $(W^{(0)}, H^{(0)}) \longleftarrow$ positive random initialization(F, K, N).
3: $(U^{(0)}, T^{(0)}) \longleftarrow G^{-1}(W^{(0)}, H^{(0)})$ where G is the change of variables
4: $Z^{(0)} \longleftarrow (U^{(0)}, T^{(0)})$
5: *% Block 2: iterative update of Z*
6: **for** $i = 1, 2, \ldots,$ maxiter **do**
7: $\quad Z^{(i)} \longleftarrow \underset{Z \in Q}{\operatorname{argmin}} \langle \nabla \Phi(Z^{(i-1)}), Z \rangle$ with IPMs available in MOSEK [25]
8: **end for**
9: $(W, H) \longleftarrow G(Z^{(i)})$

where Φ is a concave continuously differentiable function over the domain Q which is assumed to be convex and compact. Let us first describe the convergence of the sequence of objective function values $\{\Phi(Z^{(i)})\}$ obtained with Algorithm 1. Since $\Phi(Z)$ is concave, its linearization around the current iterate $Z^{(i)}$ provides an upper approximation, that is,

$$\Phi(Z) \leqslant \Phi(Z^{(i)}) + \langle \nabla \Phi(Z^{(i)}), Z - Z^{(i)} \rangle \text{ for all } Z \in Q. \tag{16}$$

This upper bound is tight at the current iterate and is exactly minimized over the feasible set Q at each iteration. Hence Algorithm 1 is a majorization-minimization algorithm. This implies that $\Phi(Z)$ is nonincreasing under the updates of Algorithm 1 and since $\Phi(Z)$ is bounded below on Q, by construction, the sequence of objective function values $\{\Phi(Z^{(i)})\}$ converges.

We now focus on the convergence analysis of the sequence of iterates $\{Z^{(i)}\}$ generated by Algorithm 1, in particular convergence to a stationary point. To achieve this goal, we first recall some basics about the Frank-Wolfe (FW) algorithm. The FW algorithm [10] is a popular first-order method to solve (15) that relies on the ability to compute efficiently the so-called Linear Minimization Oracle (LMO), that is, $LMO(D) := \underset{Z \in Q}{\operatorname{argmin}} \langle D, Z \rangle$ where D denotes some search direction. The FW algorithm with adaptive step size is given in Algorithm 2. A step of this algorithm can be briefly summarized as follows: at a current iterate $Z^{(i)}$, the algorithm considers the first-order model of the objective function (its linearization), and moves towards a minimizer of this linear function, computed on the same domain Q.

Algorithm 2 Frank-Wolfe algorithm

1: $Z^{(0)} \in Q$, number of iterations I.
2: **for** $i = 1, 2, \ldots,$ I **do**
3: Compute $V^{(i)} := \underset{Z \in Q}{\operatorname{argmin}} \langle \nabla \Phi(Z^{(i-1)}), Z \rangle$
4: Choose $0 < \tau^{(i)} \leqslant 1$. (A standard choice in the literature is $\tau^{(i)} := \frac{2}{i+1}$.)
5: Update $Z^{(i)} := (1 - \tau^{(i)}) Z^{(i-1)} + \tau^{(i)} V^{(i)}$
6: **end for**

Algorithm 1 is a particular case of Algorithm 2 for which $\tau^{(i)} = 1$ for all i. In the last decade, FW-based methods have regained interest in many fields, mainly driven by their good scalability and the crucial property that Algorithm 2 maintains its iterates as a convex combination of a few extreme points. This results in the generation of sparse and low-rank solutions since at most one additional extreme point of the set Q is added to the convex combination at each iteration. More details and insights about the later observations can be found in [6, 17]. FW algorithms have been recently studied in terms of convergence guarantees for the minimization of various classes of functions over convex sets, such as convex functions with Lipschitz continous gradient [18], and non-convex differentiable functions [22]. However, we are not aware of any convergence rates proven for Algorithm 2 when solving (15) assuming only concavity of the objective. To derive rates in the concave setting, we need to define a measure of stationarity for our iterates. In this paper, we consider the so-called FW gap of Φ at $Z^{(i)}$ defined as follows:

$$\mu^{(i)} := \max_{Z \in Q} \langle \nabla \Phi(Z^{(i)}), Z^{(i)} - Z \rangle. \tag{17}$$

This quantity is standard in the analysis of FW algorithms, see [18, 22] and the references therein. A point $Z^{(i)}$ is a stationary point for the constrained optimization problem (15) if and only if $\mu^{(i)} = 0$. Moreover, the FW gap

- provides a lower bound on the accuracy: $0 \leqslant \mu^{(i)} \leqslant \Phi(Z^{(i)}) - \Phi^*$ for all i, where $\Phi^* := \min_{Z \in Q} \Phi(Z)$,
- is affine invariant, that is, it is invariant with respect to an affine transformation of the domain Q in problem (15) [18], and
- is not tied to any specific choice of norms, unlike criteria such as $\|\nabla \Phi(Z^{(i)})\|$.

Let us provide a convergence rate for the FW algorithm (Algorithm 2).

Theorem 4.1. *Consider the problem (15) where Φ is a continuously differentiable concave function over the compact convex domain Q. Let us denote $Z^{(i)}$ the sequence of iterates generated by the FW algorithm (Algorithm 2) applied on (15). Assume there exists a constant $\tilde{\tau} > 0$ such that $\tilde{\tau} \leqslant \tau^{(i)} \leqslant 1$ for all i. Then the minimal FW gap, defined as $\tilde{\mu}^{(i)} := \min_{0 \leqslant j \leqslant i} \mu^{(j)}$, satisfies, for all $i = 1, 2, \ldots,$*

$$\tilde{\mu}^{(i)} \leqslant \frac{1}{\tilde{\tau}} \frac{\Phi(Z^{(0)}) - \Phi^*}{i+1}, \tag{18}$$

where $\Phi(Z^{(0)}) - \Phi^*$ is the (global) accuracy of the initial iterate.

Proof. Using (16), any points $Z^{(i)}$ and $Z^{(i+1)}$ generated by Algorithm 2 satisfy

$$\Phi(Z^{(i+1)}) \leqslant \Phi(Z^{(i)}) + \langle \nabla\Phi(Z^{(i)}), Z^{(i+1)} - Z^{(i)} \rangle. \tag{19}$$

Let us substitute $Z^{(i+1)}$ by its construction in Algorithm 2 (line 5) in (19) to obtain

$$\begin{aligned} \Phi(Z^{(i+1)}) &\leqslant \Phi(Z^{(i)}) + \langle \nabla\Phi(Z^{(i)}), (1-\tau^{(i)})Z^{(i)} + \tau^{(i)}V^{(i)} - Z^{(i)} \rangle \\ &= \Phi(Z^{(i)}) - \tau^{(i)} \langle \nabla\Phi(Z^{(i)}), Z^{(i)} - V^{(i)} \rangle. \end{aligned} \tag{20}$$

Let us show that $\langle \nabla\Phi(Z^{(i)}), Z^{(i)} - V^{(i)} \rangle$ is equal to $\mu^{(i)}$ defined in Equation (17):

$$\begin{aligned} \mu^{(i)} &= \max_{Z \in Q} \langle \nabla\Phi(Z^{(i)}), Z^{(i)} - Z \rangle \\ &= \langle \nabla\Phi(Z^{(i)}), Z^{(i)} \rangle + \max_{Z \in Q} \langle \nabla\Phi(Z^{(i)}), -Z \rangle \\ &= \langle \nabla\Phi(Z^{(i)}), Z^{(i)} \rangle - \min_{Z \in Q} \langle \nabla\Phi(Z^{(i)}), Z \rangle \\ &= \langle \nabla\Phi(Z^{(i)}), Z^{(i)} \rangle - \langle \nabla\Phi(Z^{(i)}), V^{(i)} \rangle = \langle \nabla\Phi(Z^{(i)}), Z^{(i)} - V^{(i)} \rangle, \end{aligned} \tag{21}$$

where the fourth equality holds be definition of $V^{(i)}$ from Algorithm 2 (line 3). Equation (20) becomes:

$$\Phi(Z^{(i+1)}) \leqslant \Phi(Z^{(i)}) - \tau^{(i)}\mu^{(i)}. \tag{22}$$

By recursively applying (22) for the iterates generated by Algorithm 2, we obtain

$$\Phi(Z^{(i+1)}) \leqslant \Phi(Z^{(0)}) - \sum_{j=0}^{i} \tau^{(j)}\mu^{(j)}. \tag{23}$$

Let define the quantities $\tilde{\mu}^{(i)} := \min\limits_{0 \leqslant j \leqslant i} \mu^{(j)}$, the minimal FW gap encountered along iterates, and $\tilde{\tau}$ such that $\tilde{\tau} \leqslant \tau^{(i)} \leqslant 1$ for all $i \geqslant 0$, so that inequality (23) implies

$$\Phi(Z^{(i+1)}) \leqslant \Phi(Z^{(0)}) - (i+1)\tilde{\tau}\tilde{\mu}^{(i)} \iff \tilde{\mu}^{(i)} \leqslant \frac{1}{\tilde{\tau}} \frac{\Phi(Z^{(0)}) - \Phi(Z^{(i+1)})}{i+1}. \tag{24}$$

Finally, using the fact that $\Phi(Z^{(0)}) - \Phi(Z^{(i+1)}) \leqslant \Phi(Z^{(0)}) - \Phi^*$ where $\Phi^* := \min\limits_{Z \in Q} \Phi(Z)$, inequality (24) becomes

$$\tilde{\mu}^{(i)} \leqslant \frac{1}{\tilde{\tau}} \frac{\Phi(Z^{(0)}) - \Phi^*}{i+1}, \tag{25}$$

which concludes the proof. □

Theorem 4.1 shows that it takes at most $\mathcal{O}(\frac{1}{\epsilon})$ iterations to find an approximate stationary point with gap smaller than ϵ. Note that Theorem 4.1 requires $\min_i \tau^{(i)} > 0$ and, for example, the standard choice $\tau^{(i)} = \frac{2}{i+1}$ does not satisfy this assumption.

4.1. Compactness assumption and convergence of Algorithm 1

By looking at the convergence rate given by (18), it is tempting to take $\tilde{\tau}$ as large as possible. However, since the set Q is convex, the maximum allowed value for $\tilde{\tau}$ is 1 to ensure that the iterates $Z^{(i)}$ remain feasible. This setting leads to a convergence rate of $\mathcal{O}(1/i)$, given the assumptions of Theorem 4.1 are satisfied, and corresponds to Algorithm 1. In Theorem 4.1, we need the set Q to be compact. Let us discuss this assumption in the context of Algorithm 1 that relies on our two formulations:

(1) For the formulation using exponential cones (5), a variable W_{fk} (resp. H_{kn}) equal to zero will correspond to U_{fk} (resp. T_{kn}) going to $-\infty$, which is not bounded. As recommended by Mosek, we use additional artificial component-wise lower and upper bounds for U and T, namely, -35 and 10. Therefore, our implementation actually solves a component-wise bounded version of (5). However, since $e^{-35} \approx 6 \cdot 10^{-16}$ and $e^{10} \approx 2 \cdot 10^4$, numerically, these constraints do not exclude good approximations of V in practice, as long as the entries in V belong to a reasonable range which can be assumed w.l.o.g. by a proper preprocessing of V, e.g., $V \leftarrow \frac{V}{\max_{f,n} V_{fn}}$ so that $V_{fn} \in [0, 1]$ for all f, n; see, e.g., the discussion in [15, page 66].

(2) For the formulation using second-order cones (9), we can assume compactness w.l.o.g. In fact, for simplicity, let us consider the formulation (6) in variables (W, H), since the change of variables is the component-wise square root, and keeps the feasible set compact. We can w.l.o.g. add a set of normalization constraints on the rows of H, such as $\sum_n H_{kn} = \|H_{k:}\|_1 = 1$ for all k which leads to a compact set for H, since we can use the degree of freedom in scaling between the columns of W and rows of H. It remains to show that W can be assumed to be in a compact set. Let us show that the level sets of the objective function are compact, which will give the result. The objective function of (6) is the component-wise ℓ_1 norm, $\|WH\|_1$. Then, let (W', H') be an arbitrary feasible solution of (6) (such a solution can be easily constructed), and add the constraint $\|WH\|_1 \leqslant \|W'H'\|_1 = f'$ to formulation (6), which does not modify its optimal solution set. We have for all k

$$\|W_{:k}\|_1 = \|W_{:k}\|_1 \|H_{k:}\|_1 = \|W_{:k} H_{k:}\|_1 \leqslant \|WH\|_1 \leqslant f',$$

which shows that W in that modified formulation also belongs to a compact set. The second equality and first inequality follow from nonnegativity of W and H.

under these modifications that make the feasible set Q compact, we have the following corollary.

Corollary 4.2. *Both variants (5) and (9) of Algorithm 1 generate a sequence of iterates $Z^{(i)}$ whose FW gap converges according to $\tilde{\mu}^{(i)} \leqslant \frac{\Phi(Z^{(0)}) - \Phi^*}{i+1}$.*

Remark 2 (Difference-of-convex-functions optimization)**.** Algorithm 1 can also be interpreted as a special case of a difference-of-convex algorithm that minimizes the difference between two convex functions, namely $f_1(Z) - f_2(Z)$ [23]. In our case, we would have $f_1(Z) = 0$. For such problems, a convergence rate to stationary points of order $O(1/i)$ has also been derived when optimizing over a convex compact feasible set, but using a different measure of stationary [1].

Remark 3. Using a proper normalization (see Section 4.1), the original NMF problem can be tackled directly by the FW algorithm (Algorithm 2), as the linear minimization oracle can be computed in closed form. Hence one could use existing results on FW to derive a convergence rate. However, the FW algorithm applied to smooth nonconvex problems leads to a worse rate of $O(1/\sqrt{i})$ [22].

5. Numerical experiments

In this section, Algorithm 1, using both formulations (5) and (9), is tested for the computation of exact NMF for particular classes of matrices usually considered in the literature: (1) 10-by-10 matrices randomly generated with nonnegative rank 5 (each matrix is generated by multiplying two random rank-5 nonnegative matrices), (2) four 6-by-6 infinitesimally rigid factorizations with nonnegative rank 5 [20], denoted $V_{\text{inf}i}$ for $i = 1, 2, 3, 4$, and (3) four 5-by-5 slack matrices corresponding to nested hexagons, denoted $V_{a=x}$, with nonnegative ranks $3, 4, 5, 5$ depending on a parameter $x = 2, 3, 4, +\infty$, respectively. These matrices are described in more details in the Appendix A.

In order to make Algorithm 1 practically more effective, we incorporate two improvements in the context of Exact NMF:

(1) Sparsity patterns integration: in NMF, entries of W and H are expected to be equal to zero at optimality. Hence, when some entries of W or H are sufficiently close to zero, fixing them to zero for all remaining iterations reduces the number of variables and hence accelerates the subsequent iterations of Algorithm 1; see Appendix B for the details.
(2) Final refinement: once a solution is generated by Algorithm 1 for the formulation based on exponential cones, it can typically be slightly improved by applying a standard NMF algorithm (we use a few iterations of A-HALS [14]). This is due to the bound constraints on W and H; see Section 4.1.

For each of the matrices, we run our Algorithm 1 for 750 iterations for nested hexagons and random matrices and 3000 iterations for infinitesimally rigid matrices, and compare with the state-of-the-art algorithms from [29] with Multi-Start 1 heuristic "ms1" and the Rank-by-rank heuristic "rbr". For each method, we run 100 initializations with SPARSE10, as recommended in [29], with target precision 10^{-6}. Note that a different random matrix is generated each time for the experiments on random matrices, following the procedure described in the Appendix A. All tests are preformed using Matlab R2021b on a laptop Intel CORE i7-11800H CPU @2.3GHz 16GB RAM. The code is available from `https://bit.ly/3FqMqhD`.

Table 1 reports the number of successes over 100 attempts to compute the exact NMF of the input matrices, where the success is defined as obtaining a solution where $\frac{\|V-WH\|_F}{\|V\|_F}$ is below the target precision, namely 10^{-6}.

We observe the following:

- All algorithms find exact NMFs in all runs for random matrices. It is well-known that factorizing randomly generated matrices is typically easier [29]. This shows that Algorithm 1 with both formulations (5) and (9) is also able to compute exact NMFs in this scenario, which is reassuring.
- Looking at the nonrandom matrices, Algorithm 1 with both formulations (5) and (9), and "ms1" from [29] are the only algorithms able to compute an exact

Table 1. Comparison of the two variants of Algorithm 1 for (5) and (9) with the algorithm from [29] with the "ms1" and "rbr" heuristics. Each run is performed with 100 initializations to compute the factorizations of matrices described in Appendix A. In bold, we indicate the algorithm that found the most exact NMFs.

		Algorithm 1 for (5)	Algorithm 1 for (9)	Algorithm from [29] with "ms1"	Algorithm from [29] with "rbr"
Matrices		**/100**	**/100**	**/100**	**/100**
Random matrices		100	100	100	100
Inf. Rig. Fac.	V_{inf1}	5	**7**	6	0
	V_{inf2}	16	38	34	**97**
	V_{inf3}	14	33	14	**90**
	V_{inf4}	16	**20**	15	0
Nested hexagons	$V_{a=2}$	100	100	100	100
	$V_{a=3}$	100	100	100	100
	$V_{a=4}$	35	69	36	**100**
	$V_{a\to+\infty}$	17	42	20	**100**

NMF for at least some of the 100 initializations. Moreover, among these three algorithms, Algorithm 1 with (9) found most frequently an exact NMF.

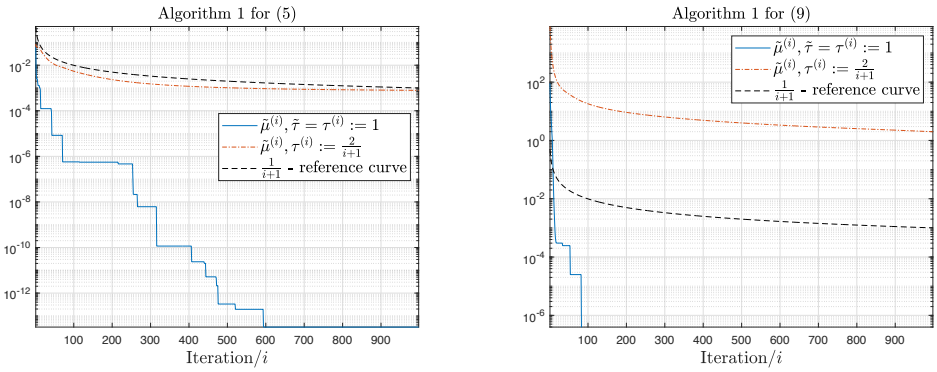
- For some matrices (namely, $V_{a=4}$ and $V_{a\to+\infty}$), "rbr" from [29] is able to compute exact NMF for all initializations, which is not the case of the other algorithms. However, "rbr" is not able to compute exact NMFs for V_{inf1} and V_{inf2}.

In summary, Algorithm 1 competes favorably with the algorithms proposed in [29], and appears to be more robust in the sense that it computes exact NMFs in all the tested cases. In addition, the second-order cone formulation, Algorithm 1 with (9), performs slightly better than with the exponential cone formulation, Algorithm 1 with (5).

Computational time In terms of computational time, Algorithm 1 performs similarly to algorithm from [29] for the considered matrices, but it does not scale as well. The main reason is that it relies on interior-point methods, while [29] relies on first-order methods (more precisely, exact BCD). For example, for the infinitesimally rigid matrices, Algorithm 1 and [29] take between 2 and 16 seconds. We would report slower performance for Algorithm 1 compared to [29] for larger matrices. Hence a possible direction of research would be to use faster methods to tackle the conic optimization problems.

Numerical validation of the rate of convergence As a simple empirical validation of the rate of convergence proposed in Section 4, we report in Figure 1 the evolution of the minimum FW gap computed along iterations by Algorithm 1 (for (5) and (9)) for one tested matrix, namely V_{inf2}. Similar behaviours were observed for all the tested input matrices: additional figures are given in Appendix C. Note that we also integrated a variant of Algorithm 1 for which $\tau^{(i)} := \frac{2}{i+1}$ (a standard choice in the literature for FW algorithms). In Figure 1, we observe that the behaviour of the minimal FW gap is in line with the theoretical results from Section 4. Furthermore, the choice $\tilde{\tau} = \tau^{(i)} := 1$ leads to faster decrease of the minimal FW gap encountered along iterations, as expected.

Impact of the initialization We also investigated the impact of using an improved initialization for our Algorithm 1 with (9), based on a rank-one NMO slightly



(a) Algorithm 1 for (5)

(b) Algorithm 1 for (9)

Figure 1. Evolution of the minimum FW gap computed along iterations by Algorithm 1 for (5) (left) and (9) (right) for the matrix V_{inf2}.

perturbed with nonnegative random uniform values. More precisely, we set $Z^{(0)} = Z_{K=1} + dR\frac{\|Z_{K=1}\|_F}{\|R\|_F}$ where $Z_{K=1}$ is the rank-one NMO computed with the methodology detailed in Section 2.2.1, R is a matrix of appropriate size of uniformly distributed random numbers in the interval $(0, 1)$, and d is a parameter to be defined by the user. In our numerical experiments, we chose values for d within the interval $[0.01, 0.05]$. We find that Algorithm 1 with (9) using that dedicated initialization found respectively 74, 65 and 52 successes for matrices $V_{a\to+\infty}$, V_{inf2} and V_{inf4}, respectively, a marked improvement over the default random initialization (where it had 42, 38, 20 successes, respectively). For the other tested matrices, it did not change the result significantly (only a few additional successes). Therefore a possible direction of research would be the design of more advanced strategies for the initialization of (U, T).

6. Conclusion

In this paper, we introduced two formulations for computing exact NMFs, namely (1) and (6) that are under- and upper-approximation formulations for NMF, respectively. For each formulation, we used a particular change of variables that enabled the use of two convex cones, namely the exponential and second-order cones, to reformulate these problems as the minimization of a concave function over a convex feasible set. In order to solve the two optimization problems, we proposed Algorithm 1 that relies on the resolution of successive linear approximations of the objective functions and the use of interior-point methods. We also showed that our optimization scheme relying on successive linearizations is a special case of the Frank-Wolfe (FW) algorithm. Using an appropriate measure of stationarity, namely the FW gap, we showed in Theorem 4.1 that the minimal FW gap generated by our algorithm converges as $\mathcal{O}(\frac{1}{i})$, where i is the iteration index. We believe this type of global convergence rate to a stationary point is new for NMF. We showed that Algorithm 1 is able to compute exact NMFs for several classes of nonnegative matrices (namely, randomly generated matrices, infinitesimally rigid matrices, and slack matrices of nested hexagons) and as such demonstrate its competitiveness compared to recent methods from the literature. However, we have only tested Algorithm 1 on a limited number of nonnegative matri-

ces for exact NMF. In the future we plan to test it on a larger library of nonnegative matrices and also to compute approximate NMFs in data analysis applications, in order to better understand the behavior of Algorithm 1 along with the two formulations, (5) and (9).

Further works also include:

- The design of more advanced strategies for the initialization of (U, T).
- The elaboration of alternative formulations for (5) and (9) to deal with the non-uniqueness of the NMF models. In particular, we plan to develop new formulations that discard solutions of the form $V = \tilde{W}\tilde{H} = (WE)\left(E^{-1}H\right)$ for a given solution (W, H) and for invertible matrices E such that $WE \geqslant 0$ and $E^{-1}H \geqslant 0$. For example, one could remove the permutation and scaling ambiguity for the columns of W and rows of H, to remove some degrees of freedom in the formulations. We refer the interested reader to [11] and [13, Chapter 4], and the references therein, for more information on the non-uniqueness of NMF.
- The use of our framework for other closely related problems; in particular the computation of symmetric NMFs which requires $H = W^\top$; this problem is closely related to completely positive matrices [4]. Symmetric NMF can be used for data analysis and in particular for various clustering tasks [21].
- The use of upper-approximations that are more accurate than linearizations for concave functions such as second-order models, and study the convergence for such models.

Acknowledgements

We thank the two reviewers for their careful reading and insightful feedback that helped us improve the paper significantly.

References

[1] H. Abbaszadehpeivasti, E. de Klerk, and M. Zamani, *On the rate of convergence of the difference-of-convex algorithm (DCA)*, arXiv preprint arXiv:2109.13566 (2021).

[2] S. Arora, R. Ge, R. Kannan, and A. Moitra, *Computing a nonnegative matrix factorization—provably*, SIAM Journal on Computing 45 (2016), pp. 1582–1611.

[3] S. Basu, R. Pollack, and M.F. Roy, *On the combinatorial and algebraic complexity of quantifier elimination*, Journal of the ACM (JACM) 43 (1996), pp. 1002–1045.

[4] A. Berman and N. Shaked-Monderer, *Completely positive matrices*, World Scientific, 2003.

[5] A. Cichocki, R. Zdunek, A.H. Phan, and S.i. Amari, *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*, John Wiley & Sons, 2009.

[6] K.L. Clarkson, *Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm*, ACM Trans. Algorithms 6 (2010).

[7] J.E. Cohen and U.G. Rothblum, *Nonnegative ranks, decompositions, and factorizations of nonnegative matrices*, Linear Algebra and its Applications 190 (1993), pp. 149–168.

[8] J. Dewez, *Computational approaches for lower bounds on the nonnegative rank*, Ph.D. diss., UCLouvain, 2022.

[9] J. Dewez, N. Gillis, and F. Glineur, *A geometric lower bound on the extension complexity of polytopes based on the f-vector*, Discrete Applied Mathematics 303 (2021), pp. 22–388.

[10] M. Frank and P. Wolfe, *An algorithm for quadratic programming*, Naval Research Logistics Quarterly 3 (1956), pp. 95–110.

[11] X. Fu, K. Huang, N.D. Sidiropoulos, and W.K. Ma, *Nonnegative matrix factorization for signal and data analytics: Identifiability, algorithms, and applications.*, IEEE Signal Process. Mag. 36 (2019), pp. 59–80.
[12] N. Gillis, *Approximation et sous-approximation de matrices par factorisation positive : algorithmes, complexite et applications*, Master's thesis, Universite Catholique de Louvain, 2007.
[13] N. Gillis, *Nonnegative Matrix Factorization*, SIAM, Philadelphia, 2020.
[14] N. Gillis and F. Glineur, *Accelerated multiplicative updates and hierarchical ALS algorithms for nonnegative matrix factorization*, Neural Computation 24 (2012), pp. 1085–1105.
[15] N. Gillis, *et al.*, *Nonnegative matrix factorization: Complexity, algorithms and applications*, Unpublished doctoral dissertation, Université catholique de Louvain. Louvain-La-Neuve: CORE (2011).
[16] N. Gillis and F. Glineur, *Using underapproximations for sparse nonnegative matrix factorization*, Pattern recognition 43 (2010), pp. 1676–1687.
[17] M. Jaggi, *Sparse convex optimization methods for machine learning*, Ph.D. diss., ETH Zurich, 2011.
[18] M. Jaggi, *Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization*, in *Proceedings of the 30th International Conference on Machine Learning*, Proceedings of Machine Learning Research Vol. 28, 17–19 Jun, Atlanta, Georgia, USA. PMLR, 2013, pp. 427–435.
[19] J. Kim and H. Park, *Fast nonnegative matrix factorization: An active-set-like method and comparisons*, SIAM Journal on Scientific Computing 33 (2011), pp. 3261–3281.
[20] R. Krone and K. Kubjas, *Uniqueness of nonnegative matrix factorizations by rigidity theory*, SIAM Journal on Matrix Analysis and Applications 42 (2021), p. 134–164.
[21] D. Kuang, C. Ding, and H. Park, *Symmetric Nonnegative Matrix Factorization for Graph Clustering*, in *Proceedings of the 2012 SIAM International Conference on Data Mining*. pp. 106–117.
[22] S. Lacoste-Julien, *Convergence rate of Frank-Wolfe for non-convex objectives*, arXiv preprint arXiv:1607.00345 (2016).
[23] H.A. Le Thi and T. Pham Dinh, *DC programming and DCA: thirty years of developments*, Mathematical Programming 169 (2018), pp. 5–68.
[24] A. Moitra, *An Almost Optimal Algorithm for Computing Nonnegative Rank*, in *Proc. of the 24th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA '13)*. 2013, pp. 1454–1464.
[25] Mosek APS, *Optimization software*. Available at `https://www.mosek.com/`.
[26] R. Sharma and K. Sharma, *A new dual based procedure for the transportation problem*, European Journal of Operational Research 122 (2000), pp. 611–624.
[27] Y. Shitov, *The nonnegative rank of a matrix: Hard problems, easy solutions*, SIAM Review 59 (2017), pp. 794–800.
[28] M. Tepper and G. Sapiro, *Nonnegative matrix underapproximation for robust multiple model fitting*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2059–2067.
[29] A. Vandaele, N. Gillis, F. Glineur, and D. Tuyttens, *Heuristics for exact nonnegative matrix factorization*, J. of Global Optim 65 (2016), p. 369–400.
[30] S.A. Vavasis, *On the complexity of nonnegative matrix factorization*, SIAM Journal on Optimization 20 (2010), pp. 1364–1377.

Appendix A. Factorized matrices

In this appendix, we describe the matrices considered for the numerical experiments in Section 5:

- Randomly generated matrices: It is standard in the NMF literature to use randomly generated matrices to compare algorithms (see, e.g., [19]). In this paper, we used the standard approach: $V = WH \in \mathbb{R}_+^{F\times N}$ where each entry of $W \in \mathbb{R}_+^{F\times K}$ and $H \in \mathbb{R}_+^{K\times N}$ is generated using the uniform distribution in the interval $[0, 1]$, and $K \leqslant \min(F, N)$. With this approach, $\operatorname{rank}(V) = \operatorname{rank}_+(V) = K$ with probability one. In the numerical experiments reported in Section 5, we used $F = N = 10$ and $K = 5$.
- Infinitesimally rigid factorizations: in this paper, we consider four infinitesimally rigid factorizations for 5×5 matrices with positive entries and with nonnegative rank equal to four from [20]:

$$V_{\text{inf1}} = \begin{pmatrix} 573705 & 806520 & 167622 & 246500 & 531659 \\ 397096 & 39600 & 299176 & 63720 & 274120 \\ 131646 & 403260 & 30269 & 226915 & 264510 \\ 9114 & 85160 & 311182 & 827468 & 851798 \\ 147857 & 3200 & 351037 & 599025 & 697755 \end{pmatrix},$$

$$V_{\text{inf2}} = \begin{pmatrix} 30893 & 319912 & 149770 & 873 & 111428 \\ 383490 & 87990 & 5580 & 628440 & 587250 \\ 560076 & 1030324 & 331070 & 288045 & 350647 \\ 203830 & 305184 & 277512 & 264376 & 205933 \\ 90911 & 142936 & 500784 & 618842 & 609633 \end{pmatrix},$$

$$V_{\text{inf3}} = \begin{pmatrix} 948201 & 723609 & 958755 & 591858 & 397953 \\ 222448 & 218040 & 30429 & 348793 & 15825 \\ 329588 & 7189 & 623001 & 12012 & 469185 \\ 467424 & 160704 & 115092 & 835504 & 343912 \\ 1114797 & 932972 & 975775 & 997164 & 636096 \end{pmatrix},$$

$$V_{\text{inf4}} = \begin{pmatrix} 88076 & 294646 & 658787 & 902872 & 244559 \\ 2216 & 4216 & 596705 & 652698 & 250465 \\ 279360 & 180864 & 769506 & 1051380 & 391634 \\ 553284 & 826606 & 765406 & 293965 & 883775 \\ 696039 & 897917 & 148301 & 832169 & 169525 \end{pmatrix}.$$

  These matrices have been shown to be challenging to factorize. We refer the reader to [20] for more details.
- Nested hexagons problem: computing an exact NMF is equivalent to tackle a well-known problem in computational geometry which is referred to as nested polytope problem. Here we consider a family of input matrices whose exact NMF corresponds to finding a polytope nested between two hexagons; see [13], Chapter

2] and the references therein. Given $x > 1$, $V_{a=x}$ is defined as

$$\frac{1}{x}\begin{pmatrix} 1 & x & 2x-1 & 2x-1 & x & 1 \\ 1 & 1 & x & 2x-1 & 2x-1 & x \\ x & 1 & 1 & x & 2x-1 & 2x-1 \\ 2x-1 & x & 1 & 1 & x & 2x-1 \\ 2x-1 & 2x-1 & x & 1 & 1 & x \\ x & 2x-1 & 2x-1 & x & 1 & 1 \end{pmatrix}$$

which satisfies $\text{rank}(V_{a=x}) = 3$ for any $x > 1$. The inner hexagon is smaller than the outer one with a ratio of $\frac{a-1}{a}$. We consider three values for a:

- $a = 2$: the inner hexagon is twiced smaller than the outer one and we can fit a triangle between the two, thus $\text{rank}_+(V_a) = 3$.
- $a = 3$: the inner hexagon is $2/3$ smaller than the outer one and we can fit a rectangle between the two, thus $\text{rank}_+(V_a) = 4$.
- $a = 4$: $\text{rank}_+(V_a) = 5$.
- $a \to +\infty$, which gives:

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

with $\text{rank}_+(V) = 5$.

Appendix B. Sparsity Patterns Integration

This appendix details the SPI procedure for quadratic cones, similar rationale has been followed for exponential cones. Due to nonnegative constraints on the entries of W and H, one can expect sparsity patterns for the solutions (W, H), as for the solution (U, T) of (9) since $W_{fk} = G(U_{fk}) = \sqrt{U_{fk}}$ and $H_{kn} = G(T_{kn}) = \sqrt{T_{kn}}$. Obviously, the sparsity for the solutions is reinforced by the sparsity of the input matrix V. One can observe that the objective function Φ from (9) is not L-smooth on the interior of the domain, that is the non-negative orthant. In the case the (f, k)-entry of the current iterate U^{i-1} tends to zero, the corresponding entry of the gradient of Φ w.r.t. U tends to ∞ which therefore ends the optimization process. In order to tackle this issue and enables the solution to reach the desired tolerance of 10^{-6}, we integrated an additional stage within the second building block of Algorithm 1. This additional stage is referred to as "Sparsity Pattern Integration". Let us illustrate its principle on the following case: the entry $U^{i-1}_{\bar{f},\bar{k}}$ tends to zero. Let us now fix $U_{\bar{f},\bar{k}}$ to zero, drop this variable from the optimization process and observe the impact on the constraints of (7) in which variable $U_{\bar{f},\bar{k}}$ is involved; the inequality constraints identified by index $f = \bar{f}$ are:

$$\sqrt{U_{\bar{f},1}}\sqrt{T_{1,n}} + ... + \sqrt{U_{\bar{f},\bar{k}}}\sqrt{T_{\bar{k},n}} + ... + \sqrt{U_{\bar{f},K}}\sqrt{T_{K,n}} \geqslant V_{\bar{f},n} \text{ for } n \in \mathcal{N}.$$

First, since $\sqrt{U_{\bar{f},\bar{k}}} = 0$, there is no more constraints on $\sqrt{T_{\bar{k},n}}$ for N inequalities identified by index $f = \bar{f}$. Second, for the problem (9) and its successive linearizations, it is then clear that N conic variables $t_{\bar{f},\bar{k},n}$ (and hence the N associated conic constraints) can be dropped from the optimization process. Finally, the linear term $\left[\nabla_U \Phi\left(U^{i-1}, T^{i-1}\right)\right]_{\bar{f},\bar{k}} U_{\bar{f},\bar{k}}$ is removed from the current linearizations of Φ. The same rationale is followed for the case entries of the current iterate for T tend to zero.

On a practical point of view, at each activation of SPI, Algorithm 1 checks if entries of the current iterates (U^{i-1}, T^{i-1}) are below a threshold th defined by the user. Hence the corresponding entries of U and T are set to zero so that a sparsity pattern is determined, that are the indices of these entries. The successive linearizations of (9) are automatically updated based on the current sparsity pattern with the approach explained above.

Let us illustrate the impact of triggering the SPI procedure on the solutions obtained for the factorization of the following matrix V:

$$V = \begin{pmatrix} 0 & 1 & 2 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 & 2 & 1 \\ 1 & 0 & 0 & 1 & 2 & 2 \\ 2 & 1 & 0 & 0 & 1 & 2 \\ 2 & 2 & 1 & 0 & 0 & 1 \\ 1 & 2 & 2 & 1 & 0 & 0 \end{pmatrix}. \tag{B1}$$

The nonnegative rank of (B1) is known and is equal to 5. Algorithm 1 is used to compute an exact NMF of V with the following input parameters:

- $K = 5 = \text{rank}_+(V)$,
- $th = 10^{-3}$,
- the maximum number of iterations defined by parameter *maxiter* is set to 500 and the SPI procedure is triggered at iteration 400.

Figure B1 displays the evolution of $\frac{\|V-WH\|_F}{\|V\|_F}$ along iterations for V (B1) with a factorization rank $K = 5$. One can observe that, once the SPI is activated, the relative Frobenius error drops from 5 10^{-4} to 8 10^{-9}, hence below the tolerance of 10^{-6} such that we assume an exact NMF (W, H) is found. For this experiment, we obtain:

$$W = \begin{pmatrix} 0 & 1.4748 & 0.9259 & 0 & 0 \\ 0.7824 & 0 & 1.8517 & 0 & 0 \\ 0 & 0 & 0.9259 & 0 & 1.4716 \\ 0 & 0 & 0 & 0.6024 & 1.4716 \\ 0.7824 & 0 & 0 & 1.2049 & 0 \\ 0 & 1.4748 & 0 & 0.6024 & 0 \end{pmatrix},$$

$$H = \begin{pmatrix} 0 & 0 & 1.2781 & 0 & 0 & 1.2781 \\ 0 & 0.6780 & 1.3561 & 0.6780 & 0 & 0 \\ 0 & 0 & 0 & 1.0801 & 1.0801 & 0 \\ 1.6599 & 1.6599 & 0 & 0 & 0 & 0 \\ 0.6796 & 0 & 0 & 0 & 0.6796 & 1.3591 \end{pmatrix}.$$

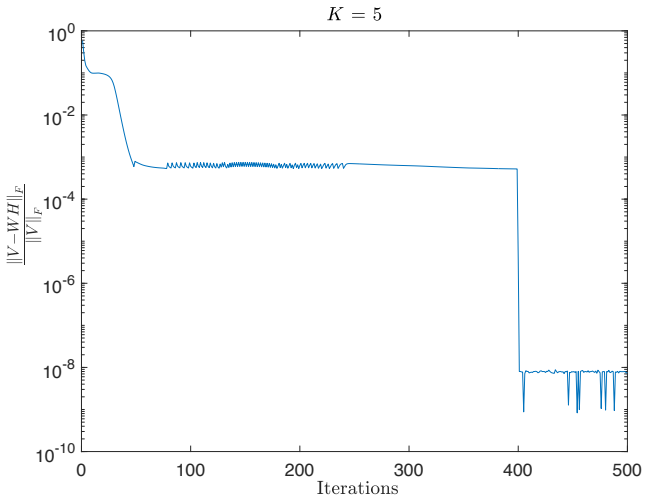
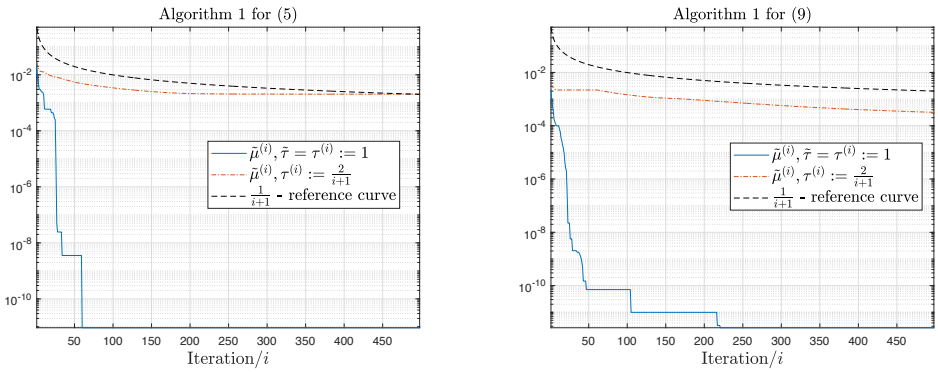


Figure B1. Evolution of $\frac{\|V-WH\|_F}{\|V\|_F}$ along iterations; SPI is triggered at iteration 400.

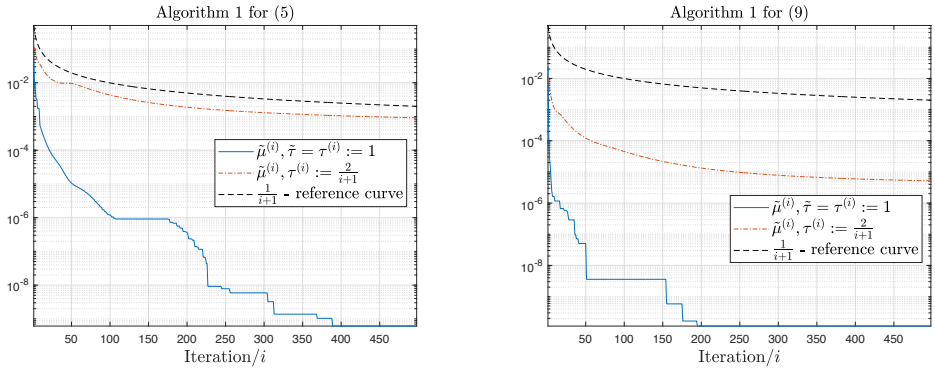


(a) Algorithm 1 for (5) (b) Algorithm 1 for (9)

Figure C1. Evolution of the minimum FW gap computed along iterations by Algorithm 1 for (5) (left) and (9) (right) for a randomly geenrated matrix.

Appendix C. Additional empirical validations of the convergence rates

In this appendix, we report in Figures C1 and C2 additional empirical validations of the convergence rate introduced in 4 for the two others classes on tested matrices, namely the random matrices and the matrices related to nested hexagons problem. For the later, we considered the particular case $a = 4$.



(a) Algorithm 1 for (5) (b) Algorithm 1 for (9)

Figure C2. Evolution of the minimum FW gap computed along iterations by Algorithm 1 for (5) (left) and (9) (right) for the matrix $V_{a=4}$.