Numerical Algorithms
https://doi.org/10.1007/s11075-020-01063-9

ORIGINAL PAPER

# Off-diagonal symmetric nonnegative matrix factorization

**François Moutier**[1] · **Arnaud Vandaele**[1] · **Nicolas Gillis**[1]

Received: 12 March 2020 / Accepted: 27 December 2020 / Published online: 04 February 2021


## Abstract

Symmetric nonnegative matrix factorization (symNMF) is a variant of nonnegative matrix factorization (NMF) that allows handling symmetric input matrices and has been shown to be particularly well suited for clustering tasks. In this paper, we present a new model, dubbed off-diagonal symNMF (ODsymNMF), that does not take into account the diagonal entries of the input matrix in the objective function. ODsymNMF has three key advantages compared to symNMF. First, ODsymNMF is theoretically much more sound as there always exists an exact factorization of size at most $n(n-1)/2$ where $n$ is the dimension of the input matrix. Second, it makes more sense in practice as diagonal entries of the input matrix typically correspond to the similarity between an item and itself, not bringing much information. Third, it makes the optimization problem much easier to solve. In particular, it will allow us to design an algorithm based on coordinate descent that minimizes the component-wise $\ell_1$ norm between the input matrix and its approximation. We prove that this norm is much better suited for binary input matrices often encountered in practice. We also derive a coordinate descent method for the component-wise $\ell_2$ norm, and compare the two approaches with symNMF on synthetic and document datasets.

**Keywords** Nonnegative matrix factorization · Clustering · $\ell_1$ norm · Coordinate descent

✉ François Moutier
francois.moutier@umons.ac.be

Arnaud Vandaele
arnaud.vandaele@umons.ac.be

Nicolas Gillis
nicolas.gillis@umons.ac.be

[1] Department of Mathematics and Operational Research, Faculté Polytechnique, Université de Mons, Rue de Houdain 9, 7000 Mons, Belgium

## 1 Introduction

Nonnegative matrix factorization (NMF) is a widely used linear dimension reduction (LDR) technique which extracts useful information in images, documents, or more generally nonnegative datasets. Given a nonnegative matrix $A \in \mathbb{R}_+^{m\times n}$ and a positive integer $r < \min(m, n)$, NMF aims at finding two nonnegative matrices $W \in \mathbb{R}_+^{m\times r}$ and $H \in \mathbb{R}_+^{n\times r}$ such that the low-rank matrix $WH^T$ approximates the input matrix $A$, which means that $A_{ij} \approx (WH^T)_{ij}$ for $i = 1, ..., m$ and $j = 1, ..., n$. The design and algorithmic implementation of refined NMF models for various applications is still a very active area of research; see [8, 11, 12, 22] and the references therein.

When the input matrix $A \in \mathbb{R}_+^{n\times n}$ is symmetric, it makes sense to look for a low-rank approximation which is symmetric as well. For this purpose, symmetric nonnegative matrix factorization (symNMF) seeks a matrix $H \in \mathbb{R}_+^{n\times r}$ such that $HH^T$ approximates $A$, that is $A_{ij} \approx (HH^T)_{ij}$ for $1 \le i, j \le n$. SymNMF is mainly used as a clustering method. In fact, the matrix $A$ usually represents the similarity measured between each pair of a set of $n$ elements. The symNMF $HH^T$ of $A$ amounts to decomposing $A$ into $r$ rank-1 factors:

$$A \approx HH^T = \sum_{k=1}^{r} H_{:,k} H_{:,k}{}^T.$$

Since the rank-1 factors are nonnegative, there is no cancellation and $A$ is approximated via the sum of $r$ rank-1 nonnegative matrices. The non-zero entries of a rank-1 factor correspond to a square submatrix of $A$ with mostly positive entries, that is, to a cluster within $A$ where all elements are highly connected. SymNMF has been used successfully in many different settings and was proved to compete with standard clustering techniques such as normalized cut, spectral clustering, k-means, and spherical k-means; see [7, 18–20, 28–30] and the references therein, and see also Section 5.2 where we compare symNMF models to k-means and spectral clustering.

In order to find the matrix $H$, the symNMF problem is mainly tackled by solving the following optimization problem:

$$\min_{H\ge 0} \quad ||A - HH^T||_F^2, \tag{1}$$

which is non-convex and NP-hard to solve [9]. Nevertheless, several local schemes were developed in order to obtain acceptable solutions—typically such algorithms are guaranteed to converge to first-order stationary points of (1); see for example [17, 19, 24, 26].

**Outline and contribution of the paper** In this work, we introduce a closely related variant of symNMF where the diagonal entries of the input matrix $A$ are not taken into account, that is, we are looking for a low-rank approximation $HH^T$ such that

$$A_{ij} \approx (HH^T)_{ij} \quad \text{for } i \ne j. \tag{2}$$

It has to be noted that this idea has already been used in the context of approximation of correlation matrices [6]. However, the nonnegativity of the factor $H$ is not enforced; hence the problem is rather different, being a symmetric eigenvalue problem efficiently solvable.

Throughout this paper, we will refer to this problem as off-diagonal SymNMF (ODsymNMF) and focus on solving

$$\min_{H \geq 0} \|A - HH^T\|_{\mathrm{OD},p} \quad \text{where} \quad \|A - HH^T\|_{\mathrm{OD},p} = \left( \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \left| A - HH^T \right|_{ij}^{p} \right)^{\frac{1}{p}}. \tag{3}$$

Although this model might be surprising at first (one may say odd), we describe its advantages and why it is meaningful in practice in Section 2. In Section 3, we develop two local algorithms based on coordinate descent (CD) to tackle the cases $p = 1$ and $p = 2$.

In Section 4, we propose an initialization scheme for ODsymNMF that is particularly crucial when $p = 1$ as it is more sensitive to initialization than when $p = 2$. In Section 5, we perform some numerical experiments on synthetic and real examples (document datasets) highlighting the validity of the ODsymNMF model.

## 2 The why of ODsymNMF

In this section, we discuss the advantages of ODsymNMF compared to symNMF. We also show that ODsymNMF for $p = 1$ is an ideal model in the rank-1 case when $A$ is binary.

### 2.1 Advantages of ODsymNMF

Let us describe the three most important advantages of ODsymNMF compared to symNMF.

**From a practical point a view** When the entries of $A$ correspond to the similarity between items, the detection of clusters is made more complicated by the overlap between clusters. As illustrated in the toy example 1 below, the sum of the two desired clusters $H_{:,1}{H_{:,1}}^T$ and $H_{:,2}{H_{:,2}}^T$ is not equal to the input matrix $A$. In the case where the diagonal entries are not taken into account, then the decomposition of $A$ into $H_{:,1}{H_{:,1}}^T + H_{:,2}{H_{:,2}}^T$ is exact in the sense that $\|A - HH^T\|_{\mathrm{OD},p} = 0$. Since a diagonal entry represents the similarity between an item and itself, it should be a large value for most similarity measures. In order to approximate these large values, the optimization in symNMF methods deteriorates the quality of the cluster detection (see Section 5 where we show that ignoring the diagonal entries leads to a better clustering accuracy).

*Example 1* For the matrix

$$\underbrace{\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}}_{A} \approx \underbrace{\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{H_{:,1}H_{:,1}^T} + \underbrace{\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}}_{H_{:,2}H_{:,2}^T},$$

symNMF is unable to perfectly recover these two clusters (in fact, one eigenvalue of $A$ is negative hence symNMF cannot exactly reconstruct this matrix even for $r$ larger than two; see below), while ODsymNMF perfectly does so as it does not take into account diagonal entries.

**From a theoretical point of view** The *cp-rank* of a matrix $A$ is the minimum positive integer $r$ such that there exists an exact factorization $A = HH^T$ where $H$ is an $n$-by-$r$ nonnegative matrix [1]. The *cp-rank* of a symmetric matrix $A$ is said to be infinite when no exact factorization $HH^T$ exists for any value of $r$. This is the case for the matrix $A$ in example 1 since $A$ has one negative eigenvalue, namely, $1 - \sqrt{2}$, while all approximations of the form $HH^T$ are positive definite hence

$$\min_{H \in \mathbb{R}_+^{n \times r}} \|A - HH^T\|_F \ \geq \ \sqrt{2} - 1$$

for any value of $r$ (this follows from the Eckart-Young theorem). On the contrary, ODsymNMF is much more sound as there always exists an exact factorization with $H$ having at most $K$ columns where $K$ is half the number of non-zero off-diagonal entries of $A$. In particular, when $A$ has only positive off-diagonal entries, we have $K = \frac{n(n-1)}{2}$. Such a factorization is obtained by using a column $H_{:,\ell}$ for each pair of entries $A_{pq} = A_{qp} \neq 0$ such that, for $i \neq j$,

$$(H_{:,\ell}H_{:,\ell}^T)_{ij} = \begin{cases} A_{pq} = A_{qp} & \text{if } (i,j) \in \{(p,q),(q,p)\}, \\ 0 & \text{otherwise,} \end{cases}$$

which can be achieved for example by choosing

$$H_{i,\ell} = \begin{cases} 1 & \text{if } i = p, \\ A_{pq} & \text{if } i = q, \\ 0 & \text{otherwise.} \end{cases}$$

This amounts to decompose $A$ as the sum of $K$ clusters containing 2 elements corresponding to each pair of non-zero entries. Of course, in practice, because of noise and model misfit, exact factorizations are not desirable; see Section 5. However, the above observation shows that ODsymNMF is much more sensible than symNMF that cannot even decompose some simple rank-2 matrices as shown for the matrix $A$ from example 1. In fact, the solution obtained by symNMF for that matrix is the following (with three digits of accuracy):

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \approx HH^T = \begin{pmatrix} 1.047 & 0.088 \\ 0.748 & 0.805 \\ 0.010 & 1.050 \end{pmatrix} \begin{pmatrix} 1.047 & 0.088 \\ 0.748 & 0.805 \\ 0.010 & 1.050 \end{pmatrix}^T = \begin{pmatrix} 1.104 & 0.854 & 0.104 \\ 0.854 & 1.207 & 0.854 \\ 0.104 & 0.854 & 1.104 \end{pmatrix}.$$

**From an algorithmic point of view** One of the most widely used optimization scheme in matrix factorization is CD which consists in updating one variable at a time while considering the other ones fixed [21, 27]. When applied to symNMF, CD requires to find the minimum of a univariate quartic non-convex polynomial, which can be done in $O(1)$ [26]. However, as pointed out in [24, 26], the drawback is that the convergence to a stationary point is not guaranteed since the minimum of the *quartic* polynomial may not be unique. As we show in Section 3, using ODsymNMF makes the optimization problem easier to solve: the sub-problem in one entry of $H$ (the others being fixed) is a *quadratic* optimization problem over the nonnegative orthant for which a closed-form solution exists. Moreover, since the optimal solution of these sub-problems is uniquely attained, convergence of CD to stationary points is guaranteed [4, 5]. Moreover, as the sub-problems of ODsymNMF are simpler, we will be able to design CD for another loss function, namely the component-wise $\ell_1$-norm which would be highly non-trivial for symNMF (see Section 3.2).

### 2.2 Rank-1 binary ODsymNMF

In many applications, the matrix $A$ is binary; hence, it is implicitly assumed that the noise is also binary [31]. This is the case for example for adjacency matrices of undirected and unweighted graphs that appear in many applications, such as community detection [10]. Although our theoretical results focus on the binary case, the algorithms we will propose in Section 3 can handle any symmetric input matrix, and we will provide numerical experiments on some non-binary real-world documents data sets in Section 5.2. For a low-rank binary input matrix and binary noise, the maximum likelihood estimator is the optimal solution of

$$\min_{H \in \mathbb{R}_+^{n \times r}} \quad ||A - HH^T||_{\mathrm{OD},0}, \tag{4}$$

where the $\ell_0$ norm counts the number of non-zero entries in $A - HH^T$, that is, the number of mismatches between $A$ and $HH^T$. An advantage of this formulation is that it produces binary solutions; see Lemma 1. Such binary solutions allow easier interpretations for most applications. However, it is not straightforward to design local schemes for (4) since the objective function is of combinatorial nature. A standard approach to deal with (4) is to replace it with its convex surrogate, the $\ell_1$-norm, where we also relax the binary constraints on $H$:

$$\min_{H \in [0,1]_+^{n \times r}} \quad \|A - HH^T\|_{\mathrm{OD},1}. \tag{5}$$

In the following, we prove that the problems in $\ell_0$ and $\ell_1$ norms, that is, (4) and (5), are equivalent for $r = 1$; see Theorem 1. Note that this equivalence was also proved in the asymmetric case, that is, for NMF [14]. This means that the $\ell_1$ norm is particularly well suited for binary input matrices, much better than the $\ell_2$ norm which generates dense solutions. In fact, in the rank-1 case, the optimal solution using the $\ell_2$ norm is always positive when $A$ is irreducible (that is, when the graph induced by $A$ is connected) which follows from the Perron-Frobenius theorem [3]; see also [14] for a discussion.

The first lemma shows that a solution of (4) can always be transformed into a binary solution with a lower objective function value; this observation is similar than in the unsymmetric case [14, Lemma 1].

**Lemma 1** *Let $h \in \mathbb{R}^n$ and let $A$ be a $n$-by-$n$ binary matrix. Applying the following simple transformation to $h$*

$$\Phi(h_i) = \begin{cases} 0 & \text{if } h_i = 0 \\ 1 & \text{otherwise} \end{cases},$$

*gives*

$$||A - \Phi(h)\Phi(h)^T||_{OD,0} \leq ||A - hh^T||_{OD,0}.$$

*Proof* There are two cases:

1. If $h_i h_j = 0$, then $\Phi(h_i)\Phi(h_j) = 0$ hence the transformation does not affect the approximation.
2. If $h_i h_j \neq 0$, then $\Phi(h_i)\Phi(h_j) = 1$. If $A_{ij} = 0$ then $||A - \Phi(h)\Phi(h)^T||_{\mathrm{OD},0} = ||A - hh^T||_{\mathrm{OD},0} = 1$ while, if $A_{ij} = 1$, $||A - hh^T||_{\mathrm{OD},0} \geq ||A - \Phi(h)\Phi(h)^T||_{\mathrm{OD},0} = 0$. □

Lemma 1 implies that the optimal solution of (4) with $r = 1$ can be assumed to be binary without loss of generality, using a simple transformation. The second lemma below shows that the same observation applies to (5).

**Lemma 2** *Let $h \in [0, 1]^n$ and let $A$ be a $n$-by-$n$ binary matrix. There exists a simple transformation to $h$ (see the proof below) that generates a binary vector $h' \in \{0, 1\}^n$ such that*

$$||A - h'h'^T||_{OD,1} \leq ||A - hh^T||_{OD,1}.$$

*Proof* Let $h \in [0, 1]^n$, and let us show that we can transform it into a binary solution with a lower objective function value. For each $i \in \{1, ..., n\}$ such that $h_i \notin \{0, 1\}$, the terms of the objective function involving $h_i$ are

$$f(h_i) = \sum_{j=1, j\neq i}^{n} \left| A_{ij} - h_i h_j \right|. \tag{6}$$

The function (6) is piecewise linear and convex; hence, minimizing it over the interval [0, 1] leads to a global minimum equal to 0, 1, or one of the breakpoints $\frac{A_{ki}}{h_k}$ where $k \in \{j \mid j \neq i, h_j \neq 0\}$. Since $A$ is binary and $0 \leq h \leq 1$, we have that $\frac{A_{ki}}{h_k}$ is either equal to 0 or is larger than 1. Therefore, 0 or 1 is a global minimum of $f(h_i)$ over the interval [0, 1], and replacing $h_i$ by 0 or 1 will decrease the objective function. □

Lemmas 1 and 2 imply that the $\ell_0$ and $\ell_1$ norm formulations of ODsymNMF are equivalent in the following sense.

**Theorem 1** *Any optimal solution of the rank-1 problem (4) can be transformed into a binary optimal solution which is also optimal for the rank-1 problem (5), and vice versa.*

*Proof* By Lemmas 1 and 2, we know that we can transform any solution into a binary solution with a smaller objective function value. For these binary solutions, the entries of the residual $P = A - hh^T$ belong to $\{-1, 0, 1\}$. Since $\|P\|_{\text{OD},0} = \|P\|_{\text{OD},1}$ for any matrix $P \in \{-1, 0, 1\}^{n \times n}$, the binary optimal solutions for one problem are also optimal for the other problem. □

Theorem 1 shows that the $\ell_1$ relaxation (5) is particularly well suited for binary input matrices. In Section 3.2, we design a CD scheme for this problem and, in Section 5, we illustrate this observation with some numerical experiments, showing that it outperforms the $\ell_2$ norm in this scenario.

## 3 Coordinate descent schemes for ODsymNMF

Coordinate descent (CD) is among the most intuitive methods to solve optimization problems [27]. At each iteration, all variables are fixed but one which is then optimized exactly or inexactly depending on the difficulty of the corresponding univariate problem. For symNMF (1) using the Frobenius norm, when all entries of $H$ are fixed except one, the optimal value of the univariate problem is the root of a polynomial of the type $x^3 + ax + b$ which can be computed in closed-form [26].

Let us introduce our general CD framework for ODsymNMF. If we optimize the $(k, l)$th entry of $H$, the univariate problem to solve is the following:

$$\min_{H_{k,l} \geq 0} \left( \sum_{i \neq j} \left| A_{i,j} - \left( \sum_{t=1}^{r} H_{:,t} H_{:,t}^T \right)_{i,j} \right|^p \right)^{\frac{1}{p}}. \tag{7}$$

To simplify the presentation, let us focus on one rank-1 factor, say $H_{:,l} H_{:,l}{}^T$, and denote $P$ the residual matrix $P = A - \sum_{t=1, t \neq l}^{r} H_{:,t} H_{:,t}{}^T$ corresponding to this factor. Let us also denote the vector $h = H(:, l)$. When optimizing the entries of $h = H(:, l)$ in CD, we face the following rank-1 ODsymNMF problem:

$$\min_{h \geq 0} \left( \sum_{i \neq j} \left( P_{i,j} - h_i h_j \right)^p \right)^{\frac{1}{p}}. \tag{8}$$

CD can be applied by solving iteratively rank-1 ODsymNMF problems for each column $H_{:,l}$ with $l = 1, ..., r$ where the entries of each column are themselves solved via CD; see Algorithm 1.

**Algorithm 1** $H = ODsymNMF(A, H_0)$.

INPUT: $A \in \mathbb{R}^{n \times n}$, $H_0 \in \mathbb{R}_+^{n \times r}$
OUTPUT: $H \in \mathbb{R}_+^{n \times r}$
$H \leftarrow H_0$
$R \leftarrow A - HH^T$
**while** stopping criterion not satisfied **do**
  **for** $l = 1 : r$ **do**
    $P \leftarrow R + H_{:,l}H_{:,l}^T$
    $H_{:,l} \leftarrow rank\_one\_ODsymNMF(P, H_{:,l})$
    $R \leftarrow P - H_{:,l}H_{:,l}^T$
  **end for**
**end while**

It remains to show how to apply CD to rank-1 ODsymNMF. In the next two subsections, we will see how to do so for the Frobenius-norm ($p = 2$) and for the component-wise $\ell_1$-norm ($p = 1$).

### 3.1 ODsymNMF with the Frobenius norm

When $p = 2$ in the optimization problem (8), we are looking for the solution minimizing the least-squares error between $P$ and its rank-1 approximation $hh^T$ without taking into account the diagonal entries. This problem can be written as

$$\min_{h \geq 0} f(h), \qquad \text{where } f(h) = \frac{1}{4} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(P_{i,j} - h_i h_j\right)^2. \tag{9}$$

For the $k$th entry of $h$, with $k \in \{1, ..., n\}$, the objective function can be decomposed as follows:

$$f(h) = \frac{1}{4} \sum_{\substack{i=1 \\ i \neq k}}^{n} \sum_{\substack{j=1 \\ j \neq i \\ j \neq k}}^{n} \left(P_{i,j} - h_i h_j\right)^2 + \frac{1}{4} \sum_{\substack{j=1 \\ j \neq k}}^{n} \left(P_{k,j} - h_k h_j\right)^2 + \frac{1}{4} \sum_{\substack{i=1 \\ i \neq k}}^{n} \left(P_{i,k} - h_i h_k\right)^2. \tag{10}$$

Since the matrix $P$ is symmetric, the last two terms of the right-hand side of (10) are equal to one another. This expression shows that the sub-problem in the entry $h_k$ is a

quadratic optimization problem whose optimal solution is either 0 or the single root of the equation $\nabla f(h)_k = 0$ where $\nabla f(h)$ represents the gradient of $f(h)$. We have

$$\nabla f(h)_k = \sum_{\substack{j=1 \\ j \neq k}}^{n} (h_k h_j^2 - P_{k,j} h_j) = a_k h_k - b_k \tag{11}$$

where $a_k = \|h\|_2^2 - h_k^2$ and $b_k = h^T P_{:,k} - h_k P_{k,k}$. The optimal value $h_k^+$ that minimizes (10) over the nonnegative orthant is

$$h_k^+ = \max\left(0, \frac{b_k}{a_k}\right). \tag{12}$$

Due to the computation of $a_k$ and $b_k$, the update of one variable with (12) can be done in $\mathcal{O}(n)$. Therefore, algorithm 1 runs in $\mathcal{O}(n^2 r)$ for updating once the $nr$ entries of $H$ since lines 7, 8, and 9 run each in $\mathcal{O}(n^2)$. However, algorithm 1 requires to store the residual matrices $P$ and $R$ which have $\mathcal{O}(n^2)$ entries. Even when the matrix $A$ is sparse, these residual matrices are usually dense which leads to a memory cost of $\mathcal{O}(n^2)$. In the following, we show how to tackle the case of large sparse matrices more efficiently by avoiding the computation of $P$ and $R$, reducing the computational costs to $\mathcal{O}(Kr)$ and the memory cost to $\mathcal{O}(K)$ where $K$ is the number of nonzero entries of $A$.

**Avoiding the explicit computation of the residual matrix** In order to compute (12), we need to compute $a_k$ and $b_k$ that depend on $P$. After some calculations by simply expanding $P$, we obtain that the optimal solution for $h_{k,l}$, all other variables being fixed, is given by

$$h_{k,l}^+ = \max\left(0, \frac{b_{k,l}}{a_{k,l}}\right),$$

where $a_{k,l} = \|H_{:,l}\|_2^2 - H_{k,l}^2$ and

$$b_{k,l} = {H_{:,l}}^T A_{:,k} - H_{:,k}(H^T H)_{:,l} - H_{k,l}(A_{k,k} + H_{k,l}^2 - \|H_{:,l}\|_2^2 - \|H_{k,:}\|_2^2).$$

Algorithm 2 uses these expressions to avoid the computation of $P$ and $R$, but produces the same output as Algorithm 1.

**Algorithm 2** $H = \text{ODsymNMF-}\ell_2(A, H_0)$.

1: INPUT: $A \in \mathbb{R}^{n \times n}$, $H_0 \in \mathbb{R}_+^{n \times r}$
2: OUTPUT: $H \in \mathbb{R}_+^{n \times r}$
3: $H \leftarrow H_0$
4: **for** $l = 1 : r$ **do**
5: $\quad C_l \leftarrow ||H_{:,l}||_2^2$
6: **end for**
7: **for** $k = 1 : n$ **do**
8: $\quad L_k \leftarrow ||H_{k,:}||_2^2$
9: **end for**
10: $D \leftarrow H^T H$
11: **while** stopping criterion not satisfied **do**
12: $\quad$ **for** $l = 1 : r$ **do**
13: $\quad\quad$ **for** $k = 1 : n$ **do**
14: $\quad\quad\quad a_{k,l} \leftarrow C_l - H_{k,l}^2$
15: $\quad\quad\quad b_{k,l} \leftarrow (H_{:,l})^T A_{:,k} - H_{k,:} D_{:,l} + H_{k,l}(C_l + L_k - A_{k,k} - H_{k,l}^2)$
16: $\quad\quad\quad H_{k,l}^+ \leftarrow \max(0, \frac{b_{k,l}}{a_{k,l}})$
17: $\quad\quad\quad C_l \leftarrow C_l + (H_{k,l}^+)^2 - H_{k,l}^2$
18: $\quad\quad\quad L_k \leftarrow L_k + (H_{k,l}^+)^2 - H_{k,l}^2$
19: $\quad\quad\quad D_{l,:} \leftarrow D_{l,:} - H_{k,:} H_{k,l} + H_{k,:}^+ H_{k,l}^+$
20: $\quad\quad\quad D_{:,l} \leftarrow (D_{l,:})^T$
21: $\quad\quad$ **end for**
22: $\quad$ **end for**
23: **end while**

Let us analyze the computational cost and memory of algorithm 2. The precomputations of $\|H_{k,:}\|_2^2$, $\|H_{:,l}\|_2^2$ in $\mathcal{O}(nr)$ (see lines 4–9) and of $D$ in $\mathcal{O}(nr^2)$ (see line 10) allow computing the optimal value $H_{k,l}^+$ in $\mathcal{O}(n)$ when $A$ is dense due to the product ${H_{:,l}}^T A_{:,k}$. It is therefore possible to apply one iteration of CD in $\mathcal{O}(n^2 r)$ operations. When $A$ contains $K$ nonzero entries, the computational complexity drops to $\mathcal{O}\left(r \max\left(K, nr\right)\right)$ since the computation of ${H_{:,l}}^T A$ can be done in $\mathcal{O}(K)$ operations. This result implies that that when $K = \mathcal{O}(n)$, which is the case for sparse matrices, algorithm 2 runs in $\mathcal{O}(nr^2)$ operations per iteration. In terms of memory, algorithm 2 only needs to store $A$ and $H$, for a cost of $\mathcal{O}(K + nr)$.

### 3.2 ODsymNMF with the component-wise $\ell_1$-norm

The $\ell_1$-norm is usually used to tackle Laplacian noise but is also a well-known surrogate of the $\ell_0$-norm in the presence of binary noise. In fact, we showed in Section 2.2 that for the ODsymNMF model using the $\ell_1$-norm is equivalent to using the $\ell_0$-norm in the rank-1 binary case. For symNMF with the $\ell_1$-norm, the univariate problem arising when using CD is a sum of absolute value of quadratic terms. Such a function is non-convex in general, making it difficult to optimize within a CD method.

With ODsymNMF, the quadratic terms disappear and we obtain a univariate convex problem. When $p = 1$ in (8), we have to minimize a sum of absolute values:

$$\min_{h \geq 0} f(h), \qquad \text{where } f(h) = \frac{1}{2} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \left| P_{i,j} - h_i h_j \right|. \tag{13}$$

As with the $\ell_2$-norm, let us focus on the $k$th variable: we have

$$f(h) = \frac{1}{2} \sum_{\substack{i=1 \\ i \neq k}}^{n} \sum_{\substack{j=1 \\ j \neq i \\ j \neq k}}^{n} \left| P_{i,j} - h_i h_j \right| + \frac{1}{2} \sum_{\substack{j=1 \\ j \neq k}}^{n} \left| P_{k,j} - h_k h_j \right| + \frac{1}{2} \sum_{\substack{i=1 \\ i \neq k}}^{n} \left| P_{i,k} - h_i h_k \right|. \tag{14}$$

The first term of the right-hand side of (14) does not involve $h_k$, and the last two terms are equal when $P$ is symmetric. Hence, the terms containing $h_k$ in the objective function $f(h)$ are $\sum_{i=1, i \neq k}^{n} \left| P_{i,k} - h_i h_k \right|$. Hence, defining $a \in \mathbb{R}^{n-1}$ as $a = h(\mathcal{K})$ where $\mathcal{K} = \{1, 2, \ldots, n\} \backslash \{k\}$, and $b \in \mathbb{R}^{n-1}$ as $b = P(k, \mathcal{K})$, finding the optimal value of $h_k$ requires solving

$$\min_{x \geq 0} \quad \sum_{i=1}^{n} |a_i x - b_i|. \tag{15}$$

The objective is a convex piecewise linear non-differentiable function, and this problem is a constrained weighted median problem. There exists an algorithm in $\mathcal{O}(n)$ operations to solve the weighted median problem [16]. In the constrained case, because (15) is convex, if the optimal solution $x^*$ is negative, we can replace it by zero to obtain the optimal solution. For the sake of completeness, Algorithm 7 in Appendix presents a simple algorithm for this constrained weighted median problem running in $\mathcal{O}(n \log n)$ operations (as it requires sorting the entries of a vector of length $n$). Finally, algorithm 3 summarizes our algorithm for the rank-1 ODsymNMF with $\ell_1$-norm.

**Algorithm 3** $h = rank_one$ODsymNMF-$\ell_1(P, h_0)$.

1: INPUT: $P \in \mathbb{R}^{n \times n}$, $h_0 \in \mathbb{R}^n_+$
2: OUTPUT: $h \in \mathbb{R}^n_+$
3: $h \leftarrow h_0$
4: **for** $k = 1 : n$ **do**
5: $\quad \mathcal{K} = \{1, 2, \ldots, n\} \backslash \{k\}$
6: $\quad a \leftarrow h(\mathcal{K})$
7: $\quad b \leftarrow P(k, \mathcal{K})$
8: $\quad h_k^+ \leftarrow$ `constrained_weighted_median`$(a, b)$ % see Algorithm 7
9: **end for**

Since algorithm 7 requires $\mathcal{O}(n \log n)$ operations, algorithm 3 runs in $\mathcal{O}(n^2 \log n)$. As noted above, the $\log n$ factor could be removed by using the weighted median algorithm from [16]. Overall, updating once each entry of $H$ using the $\ell_1$-norm when

the residual matrix $R$ is available has a computational complexity of $\mathcal{O}\left(n^2 r \log n\right)$ operations which is the same as for the $\ell_2$-norm, up to the logarithmic factor.

**Avoiding the explicit computation of the residual matrices** As for the $\ell_2$ norm, in case of a sparse input matrix $A$, we would like to avoid the computation of the residual matrices $P$ and $R$. Similarly, as for the $\ell_2$ norm, we substitute the expression $P = A - \sum_{t=1,t\neq l}^{r} H_{:,t} {H_{:,t}}^T$ in the updates of $H_{k,l}$; see algorithm 4 for the details. Since the residual matrix is not stored, the main difference lies in the computation of the terms $\sum_{t=1,t\neq l}^{r} H_{i,t} H_{k,t}$ which occurs $\mathcal{O}(n)$ times for the update of one entry. The overall computational complexity of algorithm 4 is therefore $\mathcal{O}(n^2 r^2)$ operations. As opposed to algorithms 1 and 3 running in $\mathcal{O}(n^2 r)$ operations, avoiding the storage of a $n$-by-$n$ dense matrix increases the computational cost. Moreover, unfortunately, the $\ell_1$-norm does not allow the sparsity of the input matrix $A$ to have any kind of effect in the overall complexity because $A$ is never multiplied by any other matrix during the updates (see lines 5–14 in algorithm 4). In summary, we can reduce the memory cost to $\mathcal{O}(K)$, while the computational cost slightly increases, to $\mathcal{O}(n^2 r^2)$ operations.

**Algorithm 4** $H = \text{ODsymNMF-}\ell_1(A, H_0)$.

1: INPUT: $A \in \mathbb{R}^{n\times n}$, $H_0 \in \mathbb{R}_+^{n\times r}$
2: OUTPUT: $H \in \mathbb{R}_+^{n\times r}$
3: $H \leftarrow H_0$
4: **while** stopping criterion not satisfied **do**
5:  **for** $l = 1 : r$ **do**
6:  **for** $k = 1 : n$ **do**
7:  **for** $i = 1 : n$ **do**
8:  $a_i \leftarrow H_{i,l}$
9:  $b_i \leftarrow A_{i,k} - H_{k,:} {H_{i,:}}^T + H_{i,l} H_{k,l}$
10:  **end for**
11:  $\mathcal{K} = \{1, 2, \ldots, n\} \backslash \{k\}$
12:  $H_{k,l}^+ \leftarrow$ `constrained_weighted_median`$(a(\mathcal{K}), b(\mathcal{K}))$ % see Algorithm 7
13:  **end for**
14:  **end for**
15: **end while**

### 3.3 Summary and convergence of the algorithms

Table 1 summarizes the complexity of the algorithms proposed in this section. The three lines of the table concern respectively:

- The problem (3) for $p = 2$, denoted ODsymNMF-$\ell_2$ and solved with algorithm 2,
- The problem (3) for $p = 1$, denoted ODsymNMF-$\ell_1$ and solved with algorithms 1 and 3 where a residual matrix is used,

**Table 1** Summary of the computational and memory complexities

| | General form | | Dense case $K = \mathcal{O}(n^2)$ | | Sparse case $K = \mathcal{O}(n)$ | |
|---|---|---|---|---|---|---|
| | # flops | Memory | # flops | Memory | # flops | Memory |
| ODsymNMF-$\ell_2$ Algorithm 2 | $\mathcal{O}(r \max(K, nr))$ | $\mathcal{O}(\max(K, nr))$ | $\mathcal{O}(n^2 r)$ | $\mathcal{O}(n^2)$ | $\mathcal{O}(nr^2)$ | $\mathcal{O}(nr)$ |
| ODsymNMF-$\ell_1$ Algorithm 1 and 3 | $\mathcal{O}(n^2 r)$ | $\mathcal{O}(n^2)$ | $\mathcal{O}(n^2 r)$ | $\mathcal{O}(n^2)$ | $\mathcal{O}(n^2 r)$ | $\mathcal{O}(n^2)$ |
| ODsymNMF-$\ell_1$ Algorithm 4 | $\mathcal{O}(n^2 r^2)$ | $\mathcal{O}(\max(K, nr))$ | $\mathcal{O}(n^2 r^2)$ | $\mathcal{O}(n^2)$ | $\mathcal{O}(n^2 r^2)$ | $\mathcal{O}(nr)$ |

– The problem (3) for $p = 1$, denoted ODsymNMF-$\ell_1$ and solved with algorithm 4 where the use of a residual matrix is avoided.

**Convergence** The result [4; 5, Proposition 2.7.1] guarantees that every limit point of an exact cyclic CD is a stationary point, given that

1. The objective function is continuously differentiable,
2. Each block of variables is required to belong to a closed convex set,
3. The minimum computed at each iteration for a given block of variables is uniquely attained, and
4. The objective function values in the interval between all iterates and the next (which is obtained by updating a single block of variables) are monotonically decreasing.

For the $\ell_2$ norm, the subproblems in one variable are quadratic problems in one variable (see above); hence, the four conditions above are satisfied. Therefore, every limit point of algorithm 2 is a stationary point. Note that there is at least one limit point since algorithm 2 decreases the objective function monotonically, and the level sets of ODsymNMF-$\ell_2$, that is, $\{H \geq 0 \mid \|A - HH^T\|_{\text{OD},2} \leq c\}$ for some constant $c$, are compact (Bolzano-Weierstrass theorem).

For the $\ell_1$ norm, differentiability does not hold; hence, we can only guarantee the convergence of the objective function values (which decreases monotonically and is bounded below), as well as the existence of a limit point, as for the $\ell_2$ norm. In fact, for non-differentiable objectives, counter examples exist even when all the other assumptions above are satisfied; see for example [2, Example 14.5]. However, in practice, we have observed convergence in all our numerical experiments. An interesting direction of further research would be to characterize situations in which our coordinate descent algorithm for ODsymNMF-$\ell_1$ is guaranteed to converge (or maybe adapt our algorithm so that convergence is guaranteed). As far as we know, current convergence results do not apply to ODsymNMF-$\ell_1$ [21, 27].

## 4 A new initialization scheme for ODsymNMF

In this section, we discuss the initialization of ODsymNMF algorithms, and propose a new very efficient greedy initialization scheme.

As far as we know, the only two strategies to initialize $H$ in symNMF are either at random [19] (e.g., using the uniform distribution in the interval [0,1] for each entry of $H$) or with the zero matrix of appropriate dimension [26]. However, these initializations have some drawbacks:

- When initializing $H$ randomly, the first iterates are trying at first to approximate a matrix which is highly perturbed ($A - \sum_{k=2}^{r} H_{:k} H_{:k}^T$) with a randomly generated matrix ($H_{:1} H_{:1}^T$) which is not very reasonable. Hence, the first steps are wasting the global computational effort.
- When initializing $H$ with the zero matrix, the solution found has a particular structure where the first factor is dense and the other ones are sparser. The reason is that the first factor is given more importance since it is optimized first hence it will be close to the best rank-one approximation of $A$ [26].

Note that these drawbacks apply to any $\ell_p$ norm symNMF model.

We propose in the following a greedy strategy that adapts to the norm used, and that does not have the drawbacks mentioned above while having a low computational cost (roughly $r$ iterations of our CD methods; see Table 2 below). It consists in constructing each column of $H$ sequentially by selecting non-zero entries depending on the non-zero entries of $A$. Our approach is summarized in Algorithm 5. It works as follows, the matrix $H$ is initialized with the zero matrix, and a residual matrix $R$ is initialized as the input matrix $A$ and will be updated after each column of $H$ is constructed (steps 3 and 4). The columns of $H$ are computed sequentially by repeating the following steps: for $j = 1, ..., r$,

- Initialization. The weighting vector $w$ is set as the vector of all ones of dimension $n$, and the index set $I$ as the empty set (steps 6 and 7). The vector $w$ will represent the importance of the entries of $H(:, j)$ while the index set $I$ will correspond to the non-zero entries of $H(:, j)$.
- Loop over $\{1, 2, \ldots, n\}$:
  - Find the most important element, denoted $k$: it is the element maximizing $Rw$, that is, $k = \text{argmax}_j (Rw)_j$ (step 11). Note that the very first time this loop is entered, $Rw$ is the sum of the entries in the rows of $A$ so the first element selected is the element corresponding to the row of $A$ with the largest $\ell_1$ norm. This element $k$ is added to $I$ (step 12).
  - The entry $H_{k,j}$ is then updated optimally by taking into account the information already contained in the cluster, that is, $R_{I,I}$ and $H_{I,j}$. That way the value obtained for $H_{k,j}$ is based on the values already updated, and is optimized according with the closed-form solutions derived in the previous sections (step 17). This update is referred to as $Optimize\_hk$ in algorithm 5.

For ODsymNMF-$\ell_1$, this update will use the weighted median algorithm (see step 8 of Algorithm 3) while for ODsymNMF-$\ell_2$ this update is the solution of (10) (see step 16 of Algorithm 2). For the first updated entry of a column, we set $H_{k,j} = 1$ (step 14).

- The weighting vector $w$ is updated: it is equal to the sum of the columns of $A$ in the index set $I$, that is, $w = \sum_j A(:, I)$ (step 15 or 18). This step is particularly important in the first few iterations because the weighting vector $w$ has a strong impact on the selection process. It allows to add indices in $I$ highly connected to the indices already in $I$.

- The residual is updated (step 21).

**Algorithm 5** $H = Greedy\_init(A, r)$.

1: INPUT: $A \in \mathbb{R}^{n \times n}, r \in \mathbb{N}^+$
2: OUTPUT: $H \in \mathbb{R}_+^{n \times r}$
3: $H \leftarrow 0^{n \times r}$
4: $R \leftarrow A$
5: **for** $j = 1 : r$ **do**
6: $\quad w \leftarrow 1^n$
7: $\quad I \leftarrow \{\}$
8: $\quad$ **for** $i = 1 : n$ **do**
9: $\quad\quad s \leftarrow Rw$
10: $\quad\quad s_I \leftarrow -\infty$
11: $\quad\quad [m, k] \leftarrow max(s)$
12: $\quad\quad I \leftarrow I \cup \{k\}$
13: $\quad\quad$ **if** $i = 1$ **then**
14: $\quad\quad\quad H_{k,j} \leftarrow 1$
15: $\quad\quad\quad w \leftarrow A_{:,k}$
16: $\quad\quad$ **else**
17: $\quad\quad\quad H_{k,j} \leftarrow Optimize\_hk(R_{I,I}, H_{I,j}, k)$
18: $\quad\quad\quad w \leftarrow w + A_{:,k}$
19: $\quad\quad$ **end if**
20: $\quad$ **end for**
21: $\quad R \leftarrow R - H_{:,j}(H_{:,j})^T$
22: **end for**

The computational complexity of Algorithm 5 is $\mathcal{O}(rn^3)$, which makes it too expensive in most applications; and this is not desirable: the initialization scheme should have a low computational cost compared to the optimization scheme. This heavy computational cost comes from step 9 where the computation of the product between $R$ and $w$ is $\mathcal{O}(n^2)$. However, during the last iterations, it becomes less and less necessary to compute this product since the update of the weighting vector $w$ in step 18 has a diluted effect, rendering the step 9 less useful. Therefore, we found that updating $w$ only a small number of times works as well in practice. In particular, using a multiple of the rank $r$ (typically $2r$) works very well. Moreover, to keep the

spatial complexity low, the residual matrix $R$ is not computed explicitly (as in our CD schemes). These two changes lead to Algorithm 6: it is the same as Algorithm 5 except that $s$ is only updated for $2r$ iterations, while $R$ is not computed explicitly.

**Algorithm 6** $H = Greedy\_init(A, r, p)$.

INPUT: $A \in \mathbb{R}^{n\times n}$, $r \in \mathbb{N}^+$, $p \in \{1, 2\}$
OUTPUT: $H \in \mathbb{R}_+^{n\times r}$
$H \leftarrow 0^{n\times r}$
**for** $j = 1 : r$ **do**
  $w \leftarrow 1^n$
  $I \leftarrow FALSE^n$
  $C_j \leftarrow 0$
  **for** $i = 1 : n$ **do**
    **if** $i < 2r$ **then**
      $s \leftarrow Aw - H_{:,1:j-1}\left(H_{:,1:j-1}^T w\right)$
      $s_I \leftarrow -\infty$
    **else**
      $s_I \leftarrow -\infty$
    **end if**
    $[m, k] \leftarrow max(s)$
    **if** $i = 1$ **then**
      $H_{k,j} \leftarrow 1$
      $w \leftarrow A_{:,k}$
    **else**
      **if** $p = 2$ **then**
        $b \leftarrow H_{I,j}^T A_{I,k} - \left(H_{I,j}^T H_{I,1:j-1}\right) H_{k,1:j-1}^T$
        **if** $b > 0$ **then**
          $H_{k,j}^+ \leftarrow \frac{b}{C_j}$
        **else**
          $H_{k,j}^+ \leftarrow 0$
        **end if**
      **else**
        $R_{I,k} \leftarrow A_{I,k} - H_{I,1:j-1} H_{k,1:j-1}^T$
        $H_{k,j}^+ \leftarrow$ `constrained_weighted_median`$(R_{I,k}, H_{I,j})$ %see Algorithm 7
      **end if**
      $w \leftarrow w + A_{:,k}$
    **end if**
    $I_k \leftarrow TRUE$
    $C_j \leftarrow C_j + H_{k,j}^2$
  **end for**
**end for**

The overall computational cost of Algorithm 6 is $\mathcal{O}(r^2 n^2)$ operations so it represents $r$ iterations of Algorithms 2 and 3. Unfortunately the sparsity of the input has

**Table 2** Time (in ms) to run the experimentations

| | | $\ell_1$-norm | | Frobenius-norm | |
|---|---|---|---|---|---|
| | | Greedy | $r \times$Algorithm 4 | Greedy | $r \times$Algorithm 2 |
| $n = 100$ | $r = 2$ | 76.9 | 77.5 | 74.9 | 74.5 |
| | $r = 5$ | 80.1 | 91.7 | 77.0 | 77.8 |
| | $r = 10$ | 88.0 | 142 | 83.2 | 87.4 |
| $n = 500$ | $r = 2$ | 98 | 115 | 85 | 101 |
| | $r = 5$ | 133 | 251 | 109 | 161 |
| | $r = 10$ | 181 | 623 | 141 | 289 |

The input matrices $A$ are balanced, i.e., if $n = 100$ and $r = 10$, $A$ is composed of 10 disjointed clusters of size 10 each. For each of the two norms used, the time spent in the Greedy initialization is always less than $r$ iterations of Algorithms 2 and 3

no impact on the computational cost, but the replacing of the residual matrix reduces the spatial complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(K)$.

Table 2 compares the computational times of the greedy initialization (Algorithm 6) and $r$ iterations of the corresponding ODsymNMF algorithm. We observe that the cost of the greedy initialization is of the order $r$ iterations of the ODsymNMF algorithms; in fact, it is always smaller (except for $n = 100$, $r = 2$, and for the Frobenius norm) while being significantly smaller as $n$ and $r$ increase. Hence, the greedy initialization is relatively cheap compared to running the main loops of the OD-symNMF algorithms.

## 5 Numerical experiments

In this section, we will compare the performances of the CD methods designed in Section 3 with the CD method for the usual symNMF model [26] on synthetic and real examples. Our code is available from https://sites.google.com/site/nicolasgillis/code and the numerical examples presented below can be directly run from this online code. All tests are performed using Matlab R2018a on a laptop Intel CORE i5-5200U CPU @2.2GHz 8Go RAM.

### 5.1 Synthetic examples

The main goal of the tests on synthetic examples is to show the robustness of the $\ell_1$-norm ODsymNMF when binary noise is added and the effectiveness of the greedy initialization proposed in Section 4. The different experimental setups used are the following:

– *Algorithms.* We compare our $\ell_2$-norm and $\ell_1$-norm ODsymNMF algorithms with the symNMF algorithm of [26].

– *Initialization.* We compare the greedy initialization described in Section 4 with the zero and random initializations.
– *Benchmark matrices.* The idea is to start from an input matrix for which the clustering solution $H^*$ is known and then add binary noise to that matrix. The benchmark matrices used are composed of multiple clusters of balanced sizes, that is, the matrix $A$ is a block diagonal matrix whose blocks have different size and are made up of all ones:

$$A = \begin{bmatrix} 1 \ldots 1 & 0 \ldots & & \\ \vdots \ddots \vdots & \vdots & & \\ 1 \ldots 1 & 0 \ldots & & \\ 0 \ldots 0 & 1 \ldots 1 & & \\ \vdots \quad \vdots & \vdots \ddots \vdots & & \\ & 1 \ldots 1 & & \\ & & & \ddots \end{bmatrix} \text{ with } H^* = \begin{bmatrix} 1 & 0 \ldots \\ \vdots & \vdots \\ 1 & 0 \ldots \\ 0 & 1 \\ \vdots & \vdots \\ & 1 \\ & & \ddots \end{bmatrix}.$$

To generate such matrices, we need the sizes of the clusters which we store in the vector $S$. For example, $S = [10\ 10\ 5]$ means $A$ contains 2 cliques of size 10 each and a clique of size 5 so that $A$ is a $25 \times 25$ binary matrix.
– *Evaluation metric.* The availability of the ground-truth $H^*$ allows us to quantify the performance of a clustering algorithm. We use a variation of the metric described in [23] that quantifies the level of correspondence between the clusters found $H$ and the ground truth:

$$Accuracy \ = \ 1 - \max_{P \in [1,2,\ldots,k]} \sqrt{\frac{||H_P - H^*||_F^2}{rn}} \in [0, 1], \tag{16}$$

where $[1, 2, \ldots, k]$ is the set of permutations of $\{1, 2, \ldots, k\}$ and $H_P$ is the matrix $H$ whose columns are rearranged according to the permutation $P$.

When the binary noise added is random, the experiment is repeated 30 times and we report the average accuracy of the solutions computed; this is also done when the random initialization is used.

#### 5.1.1 Random binary noise

In this first experiment, we use 10 clusters of size 10, and the noise level $\delta \in [0, 1]$ is the probability to perturb an entry of $A$. In other words, for each entry of $A$, there is a probability of $\delta$ that this entry is flipped (from 1 to 0, and vice versa).

Table 3 reports the accuracy when the noise level is fixed to 10%. For the random initialization, symNMF and ODsymNMF-$\ell_2$ perform similarly while ODsymNMF-$\ell_1$ performs badly. The reason is that ODsymNMF-$\ell_1$ is much more sensitive to initialization because it is intrinsically a more difficult problem (for $r = 1$, it is NP-hard, which is not the case for symNMF). For the greedy initialization, ODsymNMF-$\ell_1$ outperforms symNMF and ODsymNMF-$\ell_2$. This was expected as

**Table 3** Summary of the accuracies obtained for the three types of initialization in each problem

| 10% random noise | | Initialization type | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Random | | | Zero | | | Greedy | | |
| | | OD-$\ell_1$ | OD-$\ell_2$ | Sym | OD-$\ell_1$ | OD-$\ell_2$ | Sym | OD-$\ell_1$ | OD-$\ell_2$ | Sym |
| 2 clusters | Balanced | 55 | 91 | **91** | 51 | 29 | **91** | **98** | 91 | 91 |
| | Unbalanced | 60 | 87 | **88** | 62 | 29 | **85** | **94** | 87 | 88 |
| 5 clusters | Balanced | 64 | 90 | **90** | 64 | 88 | **90** | **96** | 90 | 90 |
| | Unbalanced | 66 | 89 | **89** | 64 | 55 | **90** | **96** | 90 | 90 |
| 10 clusters | Balanced | 76 | 90 | **90** | 73 | 68 | **90** | **98** | 90 | 90 |
| | Unbalanced | 78 | 86 | **88** | 73 | 68 | **88** | **93** | 89 | 88 |

OD-$\ell_1$ stands for ODsymNMF-$\ell_1$, OD-$\ell_2$ stands for ODsymNMF-$\ell_2$ and Sym stands for symNMF. The highest accuracy for each initialization type is in bold

ODsymNMF-$\ell_1$ is a better model in this scenario (Section 2.2), given that we can provide a good initial solution which is made possible through the greedy initialization. Moreover, the greedy initialization leads symNMF and ODsymNMF-$\ell_2$ to similar or better results. For this reason, we only keep the greedy initialization for the remainder of our numerical experiments.

Figure 1 provides the accuracy for the different models depending on the noise level. For low levels of noise ($\delta \leq 0.15$), ODsymNMF-$\ell_1$ recovers a very good clustering (accuracy above 90%). For larger noise levels, the performances deteriorate

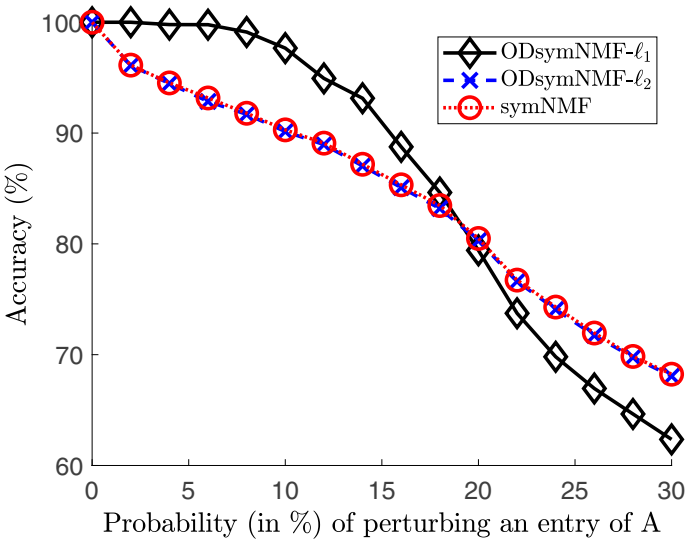


**Fig. 1** Evolution of the accuracy when random binary noise is added. The input matrix is composed of 10 cliques of size 10 each

rapidly. Above a certain noise level (around 18%), ODsymNMF-$\ell_1$ performs worse than the two other models. We believe the reason is that the dataset, above a certain noise level, is very far from a low-rank matrix and ODsymNMF-$\ell_1$ tends to get stuck more easily in bad local minima. As expected, ODsymNMF-$\ell_2$ and symNMF perform similarly since the difference between these two models is the diagonal of $A$ composed of $n$ elements. However, recall that ODsymNMF-$\ell_2$ is computationally cheaper and has convergence guarantee.

### 5.1.2 Adversarial binary noise

In order to highlight the differences between ODsymNMF-$\ell_1$ and ODsymNMF-$\ell_2$, let us construct the following adversarial example:

$$A = \begin{bmatrix} 1 & \dots & 1 & 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 1 & \dots & 1 & 0 & \dots & 0 & \vdots & & \vdots \\ 0 & \dots & 0 & 1 & \dots & 1 & \vdots & & \vdots \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & 1 & \dots & 1 & 0 & \dots & 0 \\ 0 & \dots & \dots & \dots & \dots & 0 & & & \\ \vdots & & & & & \vdots & \mathbb{I}_m & & \\ 0 & \dots & \dots & \dots & \dots & 0 & & & \end{bmatrix},$$

where the matrix $A$ is composed of two cliques and of $m$ isolated elements (identity matrix $\mathbb{I}_m$). The adversarial noise consists in adding connections between the cliques and the isolated elements. Here, the noise level is the number of connections added between an isolated element and the 2 cliques. As long as this number of connections does not exceed half the size of the cliques, we can expect ODsymNMF-$\ell_1$ to recover the ground truth. This is in fact what is observed in Fig. 2.

**Conclusions** The conclusions from running these synthetic experiments are threefold: the greedy initialization outperforms the zero and random initializations, and ODsymNMF-$\ell_1$ outperforms ODsymNMF-$\ell_2$ and symNMF in the presence of binary noise while ODsymNMF-$\ell_2$ and symNMF perform similarly.

## 5.2 Document datasets

We now perform clustering of real document datasets. These documents are represented as a word-count matrix $X \in \mathbb{N}^{n \times m}$; see Table 4.

**Similarity matrix A** In order to obtain a similarity matrix starting from the word-count matrix $X$, we choose to use a simple but powerful one: the cosine angle. The similarity between documents $a$ and $b$ is then equal to $\frac{x_a^T x_b}{\|x_a\|_2 \|x_b\|_2}$. The values inside $A$ are therefore real (not binary); this implies a lot of overlap hence a difficult problem.

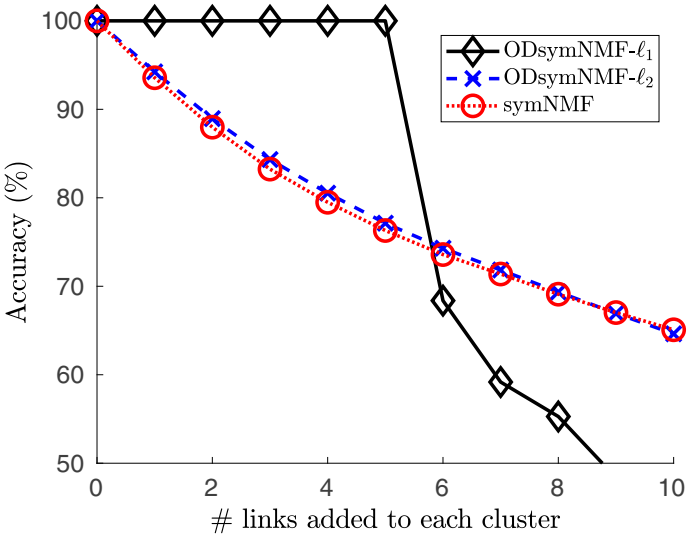


**Fig. 2** Evolution of the accuracy when adversarial binary noise is added. The input matrix is composed of 2 cliques of size 10 each and an identity matrix of size 10

The "correct" clustering is known as the documents are sorted in categories. We refer the reader to [25] for a discussion on the construction of the similarity matrix.

**Interpretation of the output *H*** Table 5 reports the accuracy (16) of symNMF, ODsymNMF-$\ell_2$, and ODsymNMF-$\ell_1$. We use the greedy initialization for all algorithms. Table 5 also reports two other widely used clustering quality measures,

**Table 4** Data of 12 documents sets from [32]

| Dataset | # documents ($= n$) | # words ($= m$) | Rank $r$ |
|---|---|---|---|
| Classic | 7094 | 41681 | 4 |
| Ohscal | 11,162 | 11,465 | 10 |
| Hitech | 2301 | 10,080 | 6 |
| Reviews | 4069 | 18,483 | 5 |
| Sports | 8580 | 14,870 | 7 |
| la1 | 3204 | 31,472 | 6 |
| la2 | 3075 | 31,472 | 6 |
| k1b | 2340 | 21,839 | 6 |
| tr11 | 414 | 6429 | 9 |
| tr23 | 204 | 5832 | 6 |
| tr41 | 878 | 7454 | 10 |
| tr45 | 690 | 8261 | 10 |

**Table 5** Comparisons between the accuracy (defined in (16)) and other common clustering metrics, namely NMI (normalized mutual information) and ARI (adjusted rand index) for each document dataset

| Dataset | Accuracy | | | NMI | | | ARI | | |
|---|---|---|---|---|---|---|---|---|---|
| | OD-$\ell_1$ | OD-$\ell_2$ | Sym | OD-$\ell_1$ | OD-$\ell_2$ | Sym | OD-$\ell_1$ | OD-$\ell_2$ | Sym |
| Classic | **66.33** | 63.67 | 63.67 | **67.74** | 67.50 | 67.50 | 22.64 | 23.27 | **23.29** |
| Ohscal | 38.08 | 43.16 | **43.24** | 82.80 | 83.89 | **83.91** | 17.78 | 21.42 | **21.46** |
| Hitech | **52.19** | 49.24 | 49.07 | **77.55** | 75.55 | 75.54 | **26.99** | 23.69 | 23.63 |
| Reviews | **70.07** | 49.55 | 49.37 | **82.64** | 71.37 | 71.43 | **54.21** | 25.89 | 25.82 |
| Sports | 48.81 | 51.41 | **51.46** | 71.86 | 73.35 | **73.35** | 20.43 | 24.14 | **24.14** |
| la1 | 40.61 | 48.81 | **49.16** | 67.08 | 72.93 | **73.29** | 14.82 | 21.46 | **22.01** |
| la2 | 39.40 | 48.62 | **48.94** | 65.21 | 73.30 | **73.44** | 10.33 | 22.24 | **22.39** |
| k1b | **66.45** | 58.68 | 57.18 | **69.46** | 68.92 | 68.31 | **33.20** | 31.26 | 29.63 |
| tr11 | 51.21 | **59.90** | 59.66 | 84.73 | **87.72** | 87.66 | 42.53 | **53.86** | 53.55 |
| tr23 | **36.76** | 35.29 | 35.29 | 67.22 | 68.86 | **68.90** | 12.09 | **13.62** | 13.33 |
| tr41 | 47.04 | **47.15** | 46.70 | 79.63 | 79.54 | **80.15** | **25.87** | 23.97 | 25.28 |
| tr45 | **43.04** | 42.61 | 42.90 | 81.51 | 83.14 | **83.58** | 22.53 | 27.41 | **28.57** |

The best results are highlighted in bold

namely the normalized mutual information (NMI) and the adjusted Rand Index (ARI). We observe that these three measures are very well correlated to each other.

We observe the following:

- As for the synthetic datasets, SymNMF and ODsymNMF-$\ell_2$ perform similarly (but bear in mind that ODsymNMF-$\ell_2$ has numerical and theoretical advantages).
- ODsymNMF-$\ell_1$ performs very differently than SymNMF and ODsymNMF-$\ell_2$. In some cases, it provides a much better accuracy (in particular, for the reviews dataset; from 50 to 70% accuracy) and, in other cases, a worse accuracy (from about 50 to 40% for la1 and la2). The reason why ODsymNMF-$\ell_1$ does not outperform SymNMF and ODsymNMF-$\ell_2$ is that the datasets are not binary, and do not follow the low-rank model very closely. However, it is interesting to observe that these models obtain rather different solutions. This means that they are able to extract different clusters within datasets. Hence, one could use aggregation techniques to obtain even better clusterings, as often done in the literature [15]. However, this is out of the scope of this paper.

In Table 6, we compare our results with two widely used clustering methods, namely k-means and spectral clustering (SC). For the sake of conciseness, we only report the accuracy which is highly correlated with the NMI and ARI (see Table 5).

We observe that symNMF models perform on average significantly better than $k$-means and SC, as already noted in the literature; see the discussion and the references in Section 1. Note also that, as expected, SC performs on average better than $k$-means.

**Table 6** Comparisons of the accuracy (in %) with other clustering methods, namely k-means and spectral clustering (SC). The best result is highlighted in bold

| Data set | OD-$\ell_1$ | OD-$\ell_2$ | Sym | k-Means | SC |
|---|---|---|---|---|---|
| Classic | **66.33** | 63.67 | 63.67 | 51.66 | 52.39 |
| Ohscal | 38.08 | 43.16 | **43.24** | 38.51 | 41.34 |
| Hitech | **52.19** | 49.24 | 49.07 | 43.75 | 49.58 |
| Reviews | **70.07** | 49.55 | 49.37 | 36.02 | 51.34 |
| sports | 48.81 | 51.41 | **51.46** | 42.75 | 41.49 |
| la1 | 40.61 | 48.81 | **49.16** | 46.29 | 48.49 |
| la2 | 39.40 | 48.62 | **48.94** | 46.38 | 48.49 |
| k1b | **66.45** | 58.68 | 57.18 | 37.42 | 36.90 |
| tr11 | 51.21 | 59.90 | **59.66** | 23.40 | 38.91 |
| tr23 | 36.76 | 35.29 | 35.29 | **45.55** | 43.54 |
| tr41 | 47.04 | **47.15** | 46.70 | 34.35 | 37.91 |
| tr45 | **43.04** | 42.61 | 42.90 | 33.23 | 39.16 |
| Average | **50.00** | 49.84 | 49.72 | 39.94 | 44.13 |

## 6 Conclusion

In this paper, we proposed a new meaningful model for symNMF by discarding the diagonal elements; we refer to this model as ODsymNMF. This allowed us to design efficient coordinate descent algorithms for the $\ell_2$ norm and $\ell_1$ norm. For the $\ell_2$ norm, our algorithm has the advantage to be computationally cheaper than the CD method of symNMF [26] (the subproblems in one variable are quadratic instead of quartic) while having convergence guarantees. For the $\ell_1$ norm, this is, to the best of our knowledge, the first algorithm of this kind for symNMF. It was made possible precisely because we discarded the diagonal elements. This $\ell_1$-norm model is better suited for binary input matrices which we theoretically proved in Section 2.2 in the rank-1 case, and empirically illustrated in Section 5.1 on synthetic datasets. We also provided numerical experiments for real document datasets, where the $\ell_2$-norm and $\ell_1$-norm models perform rather differently. Future work includes the design of other initialization strategies, as well as new symNMF-like models that would adapt to the structure of the input matrix for example using distributionally robust models as in [13].

**Funding** This work is supported by the Fonds de la Recherche Scientifique - FNRS and the Fonds Wetenschappelijk Onderzoek - Vlanderen (FWO) under EOS Project no O005318F-RG47, and by the European Research Council (ERC starting grant no 679515)

## Appendix: The constrained weighted median problem

Algorithm 7 provides a pseudocode to compute the solution to the constrained weighted median problem:

$$\min_{x \geq 0} \sum_i |a_i x - b_i|.$$

The algorithm works as follows:

- The set $S$ of breakpoints $\frac{b_i}{a_i}$ is initialized for all $i = 1, ..., n$ such that $a_i \neq 0$ (because when $a_i = 0$, the contribution of the $i$th term in the objective function is a constant) and the vector $a$ is then sorted and normalized according to the values in $S$,
- As the values $a_i$ correspond to the slopes, the second step of the algorithm looks for the $k$th breakpoint for which we have $\sum_{i=1}^{k-1} a_i < \sum_{i=k}^{n} a_i$ and $\sum_{i=1}^{k} a_i \geq \sum_{i=k+1}^{n} a_i$. It corresponds to a global optimum since the slope on the left is negative, and on the right is nonnegative.

**Algorithm 7** $x = constrained\_weighted\_median(a, b)$.

INPUT: $a \in \mathbb{R}_+^n$, $b \in \mathbb{R}^n$
OUTPUT: $x \in \mathbb{R}$
$S \leftarrow \emptyset$
**for** $i = 1 : n$ **do**
  **if** $a_i \neq 0$ **then**
    $S \leftarrow S \cup \{\frac{b_i}{a_i}\}$
  **end if**
**end for**
$[S, Inds] \leftarrow sort(S)$
$a \leftarrow \frac{a(Inds)}{sum(a)}$
$i \leftarrow 1$
$CumulatedSum \leftarrow 0$
**while** $CumulatedSum < 0.5$ **do**
  $CumulatedSum \leftarrow CumulatedSum + a_i$
  $x \leftarrow S_i$
  $i \leftarrow i + 1$
**end while**

## References


1. Abraham, B., Naomi, S.M.: Completely Positive Matrices. World Scientific (2003)
2. Beck, A.: First-Order Methods in Optimization, vol. 25. SIAM (2017)
3. Berman, A., Plemmons, R.J.: Nonnegative Matrices in the Mathematical Sciences, vol. 9. SIAM (1994)
4. Bertsekas, D.: Corrections for the book nonlinear programming (1999)
5. Bertsekas, D.: Nonlinear Programming, 2nd edn. Athena Scientific, Massachusetts (1999)

6. Borsdorf, R., Higham, N.J., Raydan, M.: Computing a nearest correlation matrix with factor structure. SIAM J. Matrix Anal. Applic. **31**(5), 2603–2622 (2010)
7. Chen, Y., Rege, M., Dong, M., Hua, J.: Non-negative matrix factorization for semi-supervised data clustering. Knowl. Inf. Syst. **17**(3), 355–379 (2008)
8. Cichocki, A., Zdunek, R., Phan, A.H., Amari, S.i.: Nonnegative matrix and tensor factorizations: Applications to exploratory multi-way data analysis and blind source separation. Wiley (2009)
9. Dickinson, P.J., Gijben, L.: On the computational complexity of membership problems for the completely positive cone and its dual. Comput. Optim. Applic. **57**(2), 403–415 (2014)
10. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**(3–5), 75–174 (2010)
11. Fu, X., Huang, K., Sidiropoulos, N.D., Ma, W.K.: Nonnegative matrix factorization for signal and data analytics: Identifiability, algorithms, and applications. IEEE Signal Process. Mag. **36**(2), 59–80 (2019)
12. Gillis, N.: The why and how of nonnegative matrix factorization. In: Suykens, J., Signoretto, M., Argyriou, A. (eds.) Regularization, Optimization, Kernels, and Support Vector Machines, chap. 12, pp. 257–291. Chapman & Hall/CRC, Boca Raton (2014)
13. Gillis, N., Hien, L.T.K., Leplat, V., Tan, V.Y.: Distributionally robust and multi-objective nonnegative matrix factorization. arXiv:1901.10757 (2019)
14. Gillis, N., Vavasis, S.A.: On the complexity of robust pca and $\ell_1$-norm low-rank matrix approximation. Math. Oper. Res. **43**(4), 1072–1084 (2018)
15. Gionis, A., Mannila, H., Tsaparas, P.: Clustering aggregation. ACM Trans. Knowl. Discov.from Data (TKDD) **1**(1), 4–es (2007)
16. Gurwitz, C.: Weighted median algorithms for $l_1$ approximation. BIT **30**(2), 301–310 (1990)
17. Huang, K., Sidiropoulos, N.D., Swami, A.: Non-negative matrix factorization revisited: Uniqueness and algorithm for symmetric decomposition. IEEE Trans. Signal Process. **62**(1), 211–224 (2013)
18. Kuang, D., Ding, C., Park, H.: Symmetric nonnegative matrix factorization for graph clustering. In: Proceedings of the 2012 SIAM International Conference on Data Mining, pp. 106–117. SIAM (2012)
19. Kuang, D., Yun, S., Park, H.: Symnmf: Nonnegative low-rank approximation of a similarity matrix for graph clustering. J. Glob. Optim. **62**(3), 545–574 (2015)
20. Long, B., Zhang, Z.M., Wu, X., Yu, P.S.: Relational clustering by symmetric convex coding. In: Proceedings of the 24th international conference on Machine learning, pp. 569–576. ACM (2007)
21. Peng, Z., Wu, T., Xu, Y., Yan, M., Yin, W.: Coordinate-friendly structures, algorithms and applications. Ann. Math. Sci. Applic. **1**(1), 57–119 (2016)
22. Pham, Q.M., Lachmund, D., Hào, D.N.: Convergence of proximal algorithms with stepsize controls for non-linear inverse problems and application to sparse non-negative matrix factorization. Numerical Algorithms (2020)
23. Pompili, F., Gillis, N., Absil, P.A., Glineur, F.: Two algorithms for orthogonal nonnegative matrix factorization with application to clustering. Neurocomputing **141**, 15–25 (2014)
24. Shi, Q., Sun, H., Lu, S., Hong, M., Razaviyayn, M.: Inexact block coordinate descent methods for symmetric nonnegative matrix factorization. IEEE Trans. Signal Process. **65**(22), 5995–6008 (2017)
25. Strehl, A., Ghosh, J., Mooney, R.: Impact of similarity measures on web-page clustering. In: Workshop on Artificial Intelligence for Web Search (AAAI 2000), vol. 58, p. 64 (2000)
26. Vandaele, A., Gillis, N., Lei, Q., Zhong, K., Dhillon, I.: Efficient and non-convex coordinate descent for symmetric nonnegative matrix factorization. IEEE Trans. Signal Process. **64**(21), 5571–5584 (2016)
27. Wright, S.J.: Coordinate descent algorithms. Math. Program. **151**(1), 3–34 (2015)
28. Yan, X., Guo, J., Liu, S., Cheng, X., Wang, Y.: Learning topics in short texts by non-negative matrix factorization on term correlation matrix. In: proceedings of the 2013 SIAM International Conference on Data Mining, pp. 749–757. SIAM (2013)
29. Yang, Z., Hao, T., Dikmen, O., Chen, X., Oja, E.: Clustering by nonnegative matrix factorization using graph random walk. In: Advances in Neural Information Processing Systems, pp. 1079–1087 (2012)
30. Zass, R., Shashua, A.: A unifying approach to hard and probabilistic clustering. In: Tenth IEEE International Conference on Computer Vision (ICCV'05), vol. 1, pp. 294–301. IEEE (2005)
31. Zhang, Z., Li, T., Ding, C., Zhang, X.: Binary matrix factorization with applications. In: Seventh IEEE International Conference on Data Mining (ICDM 2007), pp. 391–400. IEEE (2007)
32. Zhong, S., Ghosh, J.: Generative model-based document clustering: A comparative study. Knowl. Inf. Syst. **8**(3), 374–384 (2005)