Robust Binary Component Decompositions

Preprint · November 2022

CITATIONS
0

2 authors:

Christos Kolomvakis
Université de Mons
5 PUBLICATIONS **1** CITATION
SEE PROFILE

Nicolas Gillis
Université de Mons
170 PUBLICATIONS **3,284** CITATIONS
SEE PROFILE

ROBUST BINARY COMPONENT DECOMPOSITIONS

Christos Kolomvakis *Nicolas Gillis*

Department of Mathematics and Operational Research, Faculty of Engineering,
University of Mons, Rue de Houdain 9, 7000 Mons, Belgium

ABSTRACT

Semi-binary matrix factorization (semi-BMF) is a matrix decomposition model where the elements of one factor are binary. Semi-BMF can be interpreted as a generalization of k-means, and can be employed in clustering problems such as community detection. In the absence of noise, Kueng and Tropp (SIAM J. Math. Data Sc., 2021) have recently proposed a provably correct algorithm for semi-BMF that require to solve semidefinite programs (SDPs). In this paper, we extend their approach in the presence of noise. Moreover, since standard solvers for SDP rely on interior-point methods and do not scale well, we also propose a first-order method to reduce the computational costs. We test our new algorithms on synthetic data, and show that they compare favorably with the state of the art.

Index Terms— semi-binary matrix factorization, sign component decomposition, asymmetric binary component decomposition, semidefinite programming, Schur independence

1. INTRODUCTION

Matrix factorization models have been employed for a wide variety of machine learning tasks. One of the reasons of using such a model is dimensionality reduction [1]. Principal component analysis (PCA), linear discriminant analysis, and the singular value decomposition are famous examples of such models. Nonnegative matrix factorization (NMF) is a model that constraints the factors to be nonnegative. While these constraints render the problem NP-hard [2] (in contrast to the unconstrained case, like PCA), it is used for a variety of applications: hyperspectral unmixing, document classification, and community detection, to name a few; see, e.g., [3] [4] [5]. A main advantage of NMF over unconstrained models is the interpretability of the retrieved factors. In binary matrix factorization (BMF), the factors are constrained to have elements in $\{0, 1\}$. Due to the combinatorial nature of this problem, it is also hard to solve [6] [7]. We may also constrain just one of the factors to have binary elements. This model is known as semi-BMF and it has been applied in community detection and on the analysis of DNA data [8] [9]. In [10] and [11], the authors describe exact algorithms than can compute semi-BMF, when the binary factor has elements in either $\{0, 1\}$ or $\{\pm 1\}$, as long as the input matrix satisfies certain properties. The drawbacks of their algorithms, however, is that they can only be applied in the noiseless case, rendering them useless in practice. In addition, these algorithms require solving semidefinite programs (SDPs), for which standard solvers rely on interior-point methods [12], and hence are not scalable. Our goal in this paper is two fold:

1. Extend the models and algorithms proposed in [10, 11] to handle noisy input matrices.
2. Design algorithms that are less demanding in terms of memory and computations needs.

The paper is organized as follows. In Section 2, we recall the models and algorithms from Kueng and Tropp [10, 11]. In Section 3, we show how their model can be adapted in the presence of noise. In Section 4, we propose a first-order method to solve the SDPs that need to be solved in the new noisy model. In Section 5, we provide some preliminary experimental results, comparing with several state-of-the-art algorithms.

2. MODELS AND ALGORITHMS OF KUENG AND TROPP

In [10], Kueng and Tropp first consider the following matrix decomposition model.

Definition 1 (SSCD)**.** *The matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a correlation matrix and admits a rank-r SSCD if it can be decomposed as follows*

$$\mathbf{A} = \sum_{i=1}^{r} \tau_i s_i s_i^T, \tag{1}$$

where $\tau \in \Delta_+^r = \{\tau \mid \tau_i > 0 \text{ for all } i, \sum_{i=1}^r \tau_i = 1\}$, and the sign components $\mathbf{s}_i \in \{\pm 1\}^n$ for all i.

Without any additional properties regarding either $\mathbf{s}_i$ or τ_i, computing the SSCD is intractable. However, if the sign components satisfy the following property, the computation of the SSCD can be performed in polynomial time [10].

The authors acknowledge the support by the F.R.S.-FNRS and the FWO under EOS project O005318F-RG47. Nicolas Gillis also acknowledges the Francqui foundation.

Definition 2 (Schur independence, sign vectors)**.** *A collection of sign vectors,* $\{\mathbf{s}_1, , \ldots, \mathbf{s}_r\} \subset \{\pm 1\}^n$*, is Schur independent if the set* $\{\mathbf{e}\} \cup \{\mathbf{s}_i \circledast \mathbf{s}_j : 1 \leq i < j \leq r\} \subset \mathbb{R}^n$ *is linearly independent, where* $\mathbf{e}$ *is the vector of all ones, and* $\circledast$ *is the Hadamard (elementwise) product.*

Under Schur independence, [10] proposed an algorithm to compute an SSCD of a given correlation matrix $\mathbf{A}$ as in (1) using SDPs; see Algorithm 1. The rationale behind their algorithm is the geometric properties of the elliptope,

$$\mathcal{E}_n = \{\mathbf{X} \in \mathbb{H}_n : \operatorname{diag}(\mathbf{X}) = \mathbf{e} \text{ and } \mathbf{X} \succeq \mathbf{0}\},$$

where $\mathbb{H}_n$ is the set of real symmetric $n \times n$ matrices. The elliptope contains all $n \times n$ correlation matrices. In fact, given an orthogonal basis column space of $\mathbf{A}$, $\mathbf{U} = \operatorname{orth}(\mathbf{A})$, the set $\{\mathbf{X} \in \mathcal{E}_n \mid \operatorname{tr}(\mathbf{U}^\top \mathbf{X} \mathbf{U}) = n\}$ is a polyhedral face of $\mathcal{E}_n$ [13]. Hence maximizing a linear functional over this set will provide a vertex of that face, which will be a rank-one sign matrix $\mathbf{s}_i \mathbf{s}_i^\top$ which are the only vertices of the elliptope, $\mathcal{E}_n$. Once a component is identified, it can be deflated from $\mathbf{A}$, and the same trick can be applied iteratively.

Algorithm 1 Noiseless SSCD [10]

Input: $\mathbf{A} = \sum_{i=1}^r \tau_i \mathbf{s}_i \mathbf{s}_i^\top \in \mathcal{E}_n$, $\tau \in \Delta^r$ and $\mathbf{s}_i \in \{\pm 1\}^n$.
Output: Recover τ and $\{\mathbf{s}_i\}_{i=1}^r$, up to permutation.

1: $k = 1$.
2: **while** $\mathbf{A} \neq \mathbf{0}$ **do**
3: Let $\mathbf{U} = \operatorname{orth}(\mathbf{A}) \in \mathbb{R}^{n \times r}$.
4: Generate $\mathbf{g} \in \mathbb{R}^n$ randomly and solve

$$\max_{\mathbf{X} \in \mathcal{E}_n} \mathbf{g}^\top \mathbf{X} \mathbf{g} \quad \text{such that} \quad \operatorname{tr}(\mathbf{U}^\top \mathbf{X} \mathbf{U}) = n, \quad (2)$$

to obtain $\mathbf{X}^\star = \mathbf{s}_k \mathbf{s}_k^\top$ for $\mathbf{s}_k \in \{\pm 1\}^n$.
5: Calculate ζ from

$$\max_{\zeta \in \mathbb{R}} \zeta \quad \text{such that} \quad \zeta \mathbf{A} + (1 - \zeta) \mathbf{X}^* \succeq \mathbf{0}.$$

6: Update $\mathbf{A} \leftarrow \zeta^* \mathbf{A} + (1 - \zeta^*) \mathbf{X}^* \succeq \mathbf{0}$ and $k \leftarrow k + 1$.
7: **end while**
8: Compute $\tau = \operatorname{argmin}_{\mathbf{y} \in \Delta^r} ||\mathbf{A} - \sum_{i=1}^r \mathbf{y}_i \mathbf{s}_i \mathbf{s}_i^\top||_2$.

In [11], authors show that SSCD can be used to compute the following matrix factorization.

Definition 3 (Asymmetric BCD, ABCD)**.** *The matrix* $\mathbf{C} \in \mathbb{R}^{n \times m}$ *admits an ABCD if*

$$\mathbf{C} = \mathbf{Z} \mathbf{W}^T, \quad (3)$$

where $\mathbf{Z} \in \{0, 1\}^{n \times r}$ *and* $\mathbf{W} \in \mathbb{R}^{m \times r}$.

Remark 1. *ABCD described in [11] is the same as semi-BMF. For the remainder of the paper, when referring to ABCD, we are referring to the model and algorithm described in [11].*

If $\mathbf{Z}$ is Schur independent and $\mathbf{W}$ is full rank, then (3) can be computed in polynomial time. However, the definition of Schur independence for a set of binary vectors is different from Definition 2.

Definition 4 (Schur independence, binary vectors)**.** *A set of binary vectors,* $\{\mathbf{z}_1, \ldots, \mathbf{z}_r\} \subseteq \{0, 1\}^n$*, is Schur independent if the set* $\{\mathbf{e}\} \cup \{\mathbf{z}_i : 1 \leq i \leq r\} \cup \{\mathbf{z}_i \circledast \mathbf{z}_j : 1 \leq i < j \leq r\} \subset \mathbb{R}^n$ *is linearly independent.*

Due to space limitations, we do not provide the details on how ABCD is solved via SSCD, and refer the interested reader to [11].

3. ROBUST ABCD

In order to solve ABCD in the presence of noise relying on the approach of Kueng and Tropp, we need to adapt SSCD in the presence of noise.

Let us consider $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{N}$, where $\mathbf{A} \in \mathbb{H}_n$ admits an SSCD with Schur independent sign components, and $\mathbf{N} \in \mathbb{R}^{n \times n}$ is additive noise. Let us see how we can adapt the different steps of Algorithm 1:

Step 3. The basis for the column space of $\mathbf{A}$, $\mathbf{U} \in \mathbb{R}^{n \times r}$, is unknown but can be replaced by the first r left singular vectors of $\tilde{\mathbf{A}}$, denoted $\tilde{\mathbf{U}} \in \mathbb{R}^{n \times r}$. If the noise is sufficiently small, the column spaces of $\mathbf{U}$ and $\tilde{\mathbf{U}}$ will be close to one another [14].

Step 4. First, it can be proven that $\operatorname{tr}(\mathbf{U}^\top \mathbf{X} \mathbf{U}) \leq n$ holds for all $\mathbf{X} \in \mathcal{E}_n$, and hence the constraint $\operatorname{tr}(\mathbf{U}^\top \mathbf{X} \mathbf{U}) = n$ can be replaced with $\operatorname{tr}(\mathbf{U}^\top \mathbf{X} \mathbf{U}) \geq n$. Second, the SDP (2) is infeasible in general when $\mathbf{U}$ is replaced by $\tilde{\mathbf{U}}$, because the hyperplane $\{\mathbf{X} \mid \operatorname{tr}(\tilde{\mathbf{U}}^\top \mathbf{X} \tilde{\mathbf{U}}) = n\}$ intersects $\mathcal{E}_n$ only when the basis $\tilde{\mathbf{U}}$ coincide with the column space of a matrix within $\mathcal{E}_n$. This will happen with probability zero when $r < n$ and $\mathbf{N}$ is randomly generated. We therefore compute an optimal solution, $\mathbf{X}^*$, of

$$\max_{\mathbf{X} \in \mathcal{E}_n} \mathbf{g}^\top \mathbf{X} \mathbf{g} \quad \text{such that} \quad \operatorname{tr}(\tilde{\mathbf{U}}^\top \mathbf{X} \tilde{\mathbf{U}}) \geq n(1 - \epsilon), \quad (4)$$

where ϵ is a hyperparameter. We can prove by duality that the trace constraint is active, and hence replace it with equality for simplicity.

Steps 5-6. Steps 5-6 of Algorithm 1 perform a deflation. This is useful when $\mathbf{A}$ belongs to a simplicial face of the elliptope $\mathcal{E}_n$. This is however not the case for $\tilde{\mathbf{A}}$, and this deflation step cannot produce a meaningful result: in fact, because of the noise and low-rankness of $\mathbf{A}$, $\tilde{\mathbf{A}}$ is typically not positive semidefinite and does not belong to the elliptope. This is interesting to notice, and we actually preprocess the input matrix, $\tilde{\mathbf{A}}$, by projecting it onto the elliptope. We observe that this preprocessing improves the performance of our algorithm. Moreover, in the noisy case, (4) will not produce a rank-one sign matrix. Hence after computing $\mathbf{X}^*$, we let $\mathbf{u}_1$ be the first left singular vector of $\mathbf{X}$, and set $\mathbf{s}_1 = \operatorname{sign}(\mathbf{u}_1)$.

Once $\mathbf{s}_1$ is identified, how can we extract more components? Of course, we could generate another $\mathbf{g}$ randomly, hoping to obtain another rank-one sign matrix. However, there is no guarantee that this will work, and could require many attempts to find all sign components. To alleviate this, we use an orthogonalization procedure for the random $\mathbf{g}$ that we generate: the next $\mathbf{g}$ is orthogonal to the computed $\mathbf{s}_k$'s so that $\mathbf{s}_k\mathbf{s}_k^\top$ cannot maximize the (nonnegative) objective, $\mathbf{g}^\top\mathbf{X}\mathbf{g} \geq 0$, since $\mathbf{g}^\top\mathbf{s}_k\mathbf{s}_k^\top\mathbf{g} = 0$. Note that this idea could also be used in Algorithm 1 to replace its deflation step which is computationally (slightly) more demanding than simply generating $\mathbf{g}$'s orthogonal to the previously extracted $\mathbf{s}_k$'s. Still, in practice, because of the noise, the computed rank-one factor $\mathbf{s}_k\mathbf{s}_k^\top$ at step k might not be satisfactory. This can be verified by checking that $\mathrm{tr}(\tilde{\mathbf{U}}^\top\mathbf{s}_k\mathbf{s}_k^T\tilde{\mathbf{U}})$ is large enough, otherwise we start again by generating another $\mathbf{g}$.

Algorithm 2 summarizes our noisy variant of Algorithm 1.

Algorithm 2 Noisy SSCD

Input: A matrix $\tilde{\mathbf{A}} = \sum_{i=1}^r \tau_i\mathbf{s}_i\mathbf{s}_i^\top + \mathbf{N}$, where $\tau \in \Delta^r$ and $\mathbf{s}_i \in \{\pm 1\}^n$, $\epsilon > 0$.

Output: Recover τ and $\{\mathbf{s}_i\}_{i=1}^r$, up to permutation, given that $\|\mathbf{N}\|_2$ is sufficiently small.

1: $\mathbf{P} = \mathbf{I}_n$.
2: Project $\tilde{\mathbf{A}}$ to $\mathcal{E}_n$.
3: **for** $k = 1 : r$ **do**
4: Let $\tilde{\mathbf{U}}$ contain the first r left singular vectors of $\tilde{\mathbf{A}}$.
5: **repeat**
6: Generate $\mathbf{g} \in \mathbb{R}^n$ randomly, set $\mathbf{g} \leftarrow \mathbf{P}\mathbf{g}$, and solve

$$\max_{\mathbf{X}\in\mathcal{E}_n} \mathbf{g}^\top\mathbf{X}\mathbf{g} \quad \text{s.t.} \quad \mathrm{tr}(\tilde{\mathbf{U}}^\top\mathbf{X}\tilde{\mathbf{U}}) = n(1-\epsilon),$$

to obtain $\mathbf{X}^\star$.
7: Compute $\mathbf{u_1}$, the first left singular vector of $\mathbf{X}^\star$, and set $\mathbf{s}_k = \mathrm{sign}(\mathbf{u_1})$.
8: **until** a suitable sign vector $\mathbf{s}_k$ is computed
9: $\mathbf{P} = (\mathbf{I}_n - \frac{(\mathbf{P}\tilde{\mathbf{s}}_k)(\mathbf{P}\tilde{\mathbf{s}}_k)^T}{(\|\mathbf{P}\tilde{\mathbf{s}}_k)\|_2^2})\mathbf{P}$
10: **end for**
11: Compute $\tau = \mathrm{argmin}_{\mathbf{y}\in\Delta^r} \|\mathbf{A} - \sum_{i=1}^r \mathbf{y}_i\mathbf{s}_i\mathbf{s}_i^\top\|_2$.

Why does Algorithm 2 work? Let us provide a recovery guarantee for Algorithm 2. Although this result is relatively weak, we will quantify the noise level allowed, δ, and how to choose ϵ, in an extended version of this work.

Theorem 1. *Let $\mathbf{A} = \sum_{i=1}^r \tau_i\mathbf{s}_i\mathbf{s}_i^T \in \mathbb{R}^{n\times n}$, $\mathbf{N} \in \mathbb{R}^{n\times n}$ be the noise such that $\|\mathbf{N}\|_2 \leq \delta$, and $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{N}$. For δ sufficiently small, there exists some $\epsilon \geq 0$ such that Algorithm 2 recovers $\{\mathbf{s}_i\}_{i=1}^r$ exactly, with probability one.*

Sketch of the proof. First, let us denote $\tilde{\mathbf{U}}(\delta)$ the column space of the first r singular vectors of $\tilde{\mathbf{A}}$. This subspace changes smoothly with δ, and coincide with that of $\mathbf{U}$ for $\delta = 0$ [14]. In the noiseless SDP (2), we can prove that if the $\mathbf{s}_i$'s generating $\mathbf{A}$ are Schur independent, then $\mathrm{tr}(\mathbf{U}^\top\mathbf{s}\mathbf{s}^\top\mathbf{U}) \leq n-1$ for any $\mathbf{s} \in \{\pm 1\}^n$ with $\mathbf{s} \neq \mathbf{s}_i$ for all i. The function $f(\mathbf{X}) = \mathrm{tr}(\tilde{\mathbf{U}}(\delta)^\top\mathbf{X}\tilde{\mathbf{U}}(\delta))$ changes continuously with δ, while we know $f(\mathbf{X}) \leq n$ for any $\mathbf{X} \in \mathcal{E}_n$. Therefore, for a well chosen ϵ, given that δ is sufficiently small, the only rank-one sign matrices feasible for (4) will be the rank-one components of $\mathbf{A}$.

Recall that, with probability one, the optimal solution of (4) is unique (it identifies a vertex of the face of $\mathcal{E}_n$ that contains A in its interior). We have that (4) changes smoothly with δ and ϵ [15], and coincide with (2) when $\delta = \epsilon = 0$. Hence for δ sufficiently small, there exists ϵ such that an optimal solution of (4), $\mathbf{X}^\star$, will be close to that of (4) (which is a rank-one sign vector of $\mathbf{A}$). After the post-processing of $\mathbf{X}^\star$ (step 7 of Algorithm 2), we will recover a ground truth factor $\mathbf{s}_i$. □

The full proof will be given in an extended version of this paper. Of course, in practice, the noise level, δ, might not be small enough. However, as we will see, Algorithm 2 is able to recover accurately the sign vectors that generated $\mathbf{A}$ even for relatively large noise levels.

4. FIRST-ORDER METHOD FOR SSCD

One way to solve (4) with high accuracy is through an interior point method [12]. For an $n \times n$ input correlation matrix, this requires $O(n^6)$ operations per iteration, which makes it impractical for large data sets. To reduce the computational cost, we propose a first-order method to solve (4); see Algorithm 3. Note, and this is crucial, that we do not need a high accuracy solution of (4), since we are going to postprocess the solution to obtain a rank-one sign matrix.

For the projection step onto the feasible set, $\mathbb{E}_n \cap \mathbb{T}_\epsilon$, where $\mathbb{T}_\epsilon = \{\mathbf{X} \mid \mathrm{tr}(\tilde{\mathbf{U}}^\top\mathbf{X}\tilde{\mathbf{U}}) = n(1-\epsilon)\}$, we use an alternating projection strategy. We choose to first project onto the set of correlation matrices, and then onto the set $\mathbb{T}_\epsilon$. A method for the projection onto the elliptope, inspired by Dykstra's algorithm [16], is described in [17]. The projection onto $\mathbb{T}_\epsilon$ is simple, and given by $P_{\mathbb{T}_\epsilon}(\mathbf{Y}) = \mathbf{Y} - \frac{\mathrm{tr}(\mathbf{U}\mathbf{U}^T\mathbf{Y}) - n(1-\epsilon)}{r}\mathbf{U}\mathbf{U}^T$. We perform 10 iterations of this alternating strategy.

The cost function $f(\mathbf{X}) = \mathbf{g}^T\mathbf{X}\mathbf{g}$ is a linear function of $\mathbf{X}$ and its gradient is $\nabla f(\mathbf{X}) = \mathbf{g}\mathbf{g}^T$. In our implementation, we use a naive step-size, namely $1/\|\mathbf{g}\|_2^2$, which appears to perform well in practice. We will design a more sophisticated first-order method in an extended version of this work (in particular, using the Frank-Wolfe algorithm would make sense). Surprisingly, we observe that performing only 5 iterations of the projected gradient is enough to obtain competitive results on synthetic data sets.

5. NUMERICAL EXPERIMENTS

As far as we know, there do not exist heuristic algorithms for SSCD itself, only the algorithm of [10] in the noise-

Algorithm 3 Projected gradient method for SSCD (4)

Input: Initial matrix $\mathbf{X} \in \mathbb{R}^{n\times n}$, matrix $\mathbf{U} \in \mathbb{R}^{n\times r}$, a vector $\mathbf{g} \in \mathbb{R}^n$, a scalar $\epsilon > 0$.

Output: Solves (4), approximately.

$k = 1$, $\mathbf{X}_k = \mathbf{X}$, $L = \sigma_1(\mathbf{g}\mathbf{g}^T) = \|\mathbf{g}\|_2^2$
while terminating condition is false **do**
  $\mathbf{X}_{k+1} = \mathbf{X}_k - \frac{1}{L}(-\mathbf{g}\mathbf{g}^T)$
  while the alternating projection has not converged **do**
    Project $\mathbf{X}_{k+1}$ to $\mathcal{E}_n$ according to [17].
    Set $\mathbf{X}_{k+1} \leftarrow \mathbf{X}_{k+1} - \frac{\mathrm{tr}(\mathbf{U}\mathbf{U}^T\mathbf{Y}) - n(1-\epsilon)}{r}\mathbf{U}\mathbf{U}^T$
  end while
  $k \leftarrow k + 1$.
end while

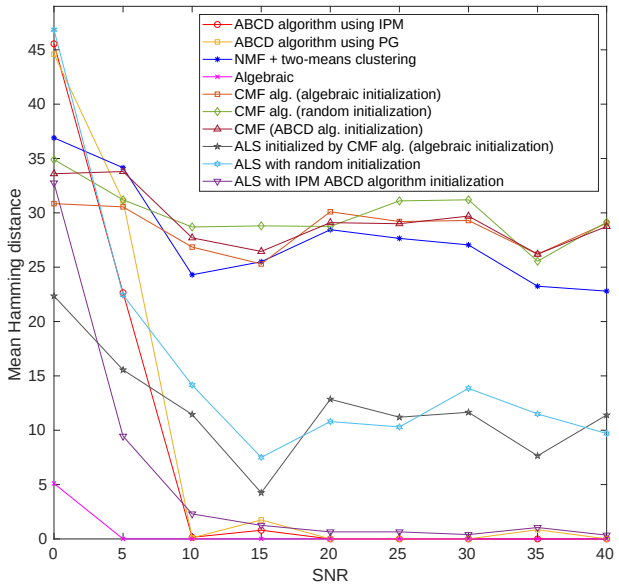

Fig. 1. Mean Hamming Distance over 20 trials.

less case. Hence, as far as we know, *Algorithm 2 is the first to handle the noisy SSCD problem.* On the contrary, Semi-BMF is rather popular, and several heuristics have been introduced recently. We will compare our approach to a tensor-based approach [18] and the matrix decomposition-based approaches in [9]. All experiments in this section are run on MATLAB R2018a on a laptop with AMD Ryzen 7 5800H @ 3.2 GHz and 16GB RAM. The code is available at https://gitlab.com/ckolomvakis/robust-binary-component-decompositions. For our first experiment, we consider $r = 4$, $n = 45$, $m = 65$. The input matrix $\mathbf{C} \in \mathbb{R}^{n\times m}$ is generated via a binary factor $\mathbf{Z} \in \{0, 1\}^{n\times r}$, whose elements have a probability of $1/2$ to be either 0 or 1, and a random matrix $\mathbf{W} \in \mathbb{R}^{r\times m}$ with standard normal elements. Random Gaussian noise $\mathbf{N} \in \mathbb{R}^{n\times m}$ is then added, as well as an additional term $1/15(\mathbf{c}\mathbf{d}^T)$, where $\mathbf{c} \in \mathbb{R}^n$ and $\mathbf{d} \in \mathbb{R}^m$ are uniformly distributed in $[0, 1]$. We define the SNR as $SNR = 10\log_{10}\left(\|\mathbf{Z}\mathbf{W}\|_F^2/\sigma_N^2\|\mathbf{N}\|_F^2\right)$, with σ_N^2 varying according to the SNR value considered. The number of Monte Carlo trials for each SNR value is set to 20. For ABCD, we solve the SDPs through cvx [19] (denoted as 'ABCD algorithm using IPM'), and through our first-order approach (denoted as 'ABCD algorithm using PG'). We set $\epsilon = 0.05$. The approaches presented in [9] are a coupled matrix factorization method (denoted as 'CMF alg.') and an ALS method for Semi-BMF (denoted as 'ALS'). The tensor-based approach from [18] is denoted as 'Algebraic'. The Hamming distance is calculated as follows: We convert the binary factor retrieved into a factor in $\{\pm 1\}^{n\times r}$. Then, we compare all columns according to the metric $d(\mathbf{s}_i, \mathbf{s}_j) = \|(\mathbf{e} - \mathbf{s}_i \circledast \mathbf{s}_j)\|_2^2/4$, where $i, j \in \{1, \ldots, r\}$ and $i \neq j$. If $d(\mathbf{s}_i, \mathbf{s}_j) > n - d(\mathbf{s}_i, \mathbf{s}_j)$, then we set $d(\mathbf{s}_i, \mathbf{s}_j) = n - d(\mathbf{s}_i, \mathbf{s}_j)$. Figure 1 displays the result. We observe that for all SNR values considered, the best performance is achieved by the 'Algebraic' method. For $SNR > 10$, our approach becomes competitive and performs better than most of the other methods considered.

Scalability Let us compare the execution time for the two methodologies presented to tackle ABCD. In this experiment, $\mathbf{C}$ is generated via a binary factor $\mathbf{Z}$, whose elements are drawn from a uniform distribution in $[0, 1]$ and then rounded to the nearest integer, while $\mathbf{W}$ has elements generated from a standard normal distribution. Standard normal noise $\mathbf{N}$ is added and the SNR for this experiment is at 13 dB. We set $\epsilon = 0.05$ for both methods. The number of trials for each instance considered is 20. The results are presented in the table below, where the mean execution time for each method is presented in seconds.

(n, m, r)	t_{IPM}	t_{PG}
$(90, 90, 6)$	247.2	82.0
$(140, 120, 5)$	1623.3	188.7
$(200, 200, 5)$	NaN	521.9
$(260, 290, 4)$	NaN	915.7

We see that cvx could not solve the last two cases. In one of the trials for $(n, m, r) = (140, 120, 5)$, the first-order method had found one element of the binary factor wrong. In the rest of the trials, all methods retrieved the binary factor exactly.

6. CONCLUSION

In this work, we extended the algorithms for SSCD and ABCD proposed in [10, 11] to handle noisy data. Furthermore, we proposed a first-order method to handle larger problems. This will pave the way to apply our ABCD algorithm to real data in future works. Future work also includes developing further our robustness analysis of the SDP-based model, and our first-order method.

7. REFERENCES

[1] Sergios Theodoridis and Konstantinos Koutroumbas, *Pattern Recognition, Fourth Edition*, Academic Press, Inc., USA, 4th edition, 2008.

[2] Stephen A. Vavasis, "On the complexity of nonnegative matrix factorization," *SIAM Journal on Optimization*, vol. 20, no. 3, pp. 1364–1377, 2010.

[3] Nicolas Gillis, *Nonnegative Matrix Factorization*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2020.

[4] Ioannis Psorakis, Stephen Roberts, Mark Ebden, and Ben Sheldon, "Overlapping community detection using bayesian non-negative matrix factorization," *Phys. Rev. E*, vol. 83, pp. 066114, Jun 2011.

[5] Xiao Fu, Kejun Huang, Nicholas D. Sidiropoulos, and Wing-Kin Ma, "Nonnegative matrix factorization for signal and data analytics: Identifiability, algorithms, and applications," *IEEE Signal Processing Magazine*, vol. 36, no. 2, pp. 59–80, 2019.

[6] Zhongyuan Zhang, Tao Li, Chris Ding, and Xiangsun Zhang, "Binary matrix factorization with applications," in *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, 2007, pp. 391–400.

[7] Ravi Kumar, Rina Panigrahy, Ali Rahimi, and David Woodruff, "Faster algorithms for binary matrix factorization," in *Proceedings of the 36th International Conference on Machine Learning*, Kamalika Chaudhuri and Ruslan Salakhutdinov, Eds. 09–15 Jun 2019, vol. 97 of *Proceedings of Machine Learning Research*, pp. 3551–3559, PMLR.

[8] Martin Slawski, Matthias Hein, and Pavlo Lutsik, "Matrix factorization with binary components," in *Advances in Neural Information Processing Systems*, C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, Eds. 2013, vol. 26, Curran Associates, Inc.

[9] Mikael Sørensen, Nicholas D. Sidiropoulos, and Anan-thram Swami, "Overlapping community detection via semi-binary matrix factorization: Identifiability and algorithms," *IEEE Transactions on Signal Processing*, vol. 70, pp. 4321–4336, 2022.

[10] Richard Kueng and Joel A. Tropp, "Binary component decomposition part I: the positive-semidefinite case," *SIAM Journal on Mathematics of Data Science*, vol. 3, no. 2, pp. 544–572, 2021.

[11] Richard Kueng and Joel A. Tropp, "Binary component decomposition part II: the asymmetric case," *Arxiv preprint*, 2019.

[12] Yurii Nesterov and Arkadii Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*, Society for Industrial and Applied Mathematics, 1994.

[13] Monique Laurent and Svatopluk Poljak, "On the facial structure of the set of correlation matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 17, no. 3, pp. 530–547, 1996.

[14] G.W. Stewart and J.-g. Sun, *Matrix Perturbation Theory*, Academic Press Inc, 1990.

[15] J Frédéric Bonnans and Alexander Shapiro, *Perturbation analysis of optimization problems*, Springer Science & Business Media, 2013.

[16] James P. Boyle and Richard L. Dykstra, "A method for finding projections onto the intersection of convex sets in hilbert spaces," in *Advances in Order Restricted Statistical Inference*, Richard Dykstra, Tim Robertson, and Farroll T. Wright, Eds., New York, NY, 1986, pp. 28–47, Springer New York.

[17] Nicholas J. Higham, "Computing the nearest correlation matrix—a problem from finance," *IMA Journal of Numerical Analysis*, vol. 22, no. 3, pp. 329–343, 07 2002.

[18] Mikael Sørensen, Lieven De Lathauwer, and Nicholaos D Sidiropoulos, "Bilinear factorizations subject to monomial equality constraints via tensor decompositions," *Linear Algebra and its Applications*, vol. 621, pp. 296–333, 2021.

[19] Michael Grant and Stephen Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," `http://cvxr.com/cvx`, Mar. 2014.