Positive Semidefinite Matrix Factorization Based on Truncated Wirtinger Flow

Dana Lahat, Cédric Févotte

Positive Semidefinite Matrix Factorization Based on Truncated Wirtinger Flow

Dana Lahat, Cédric Févotte
*IRIT, Université de Toulouse, CNRS, Toulouse, France
{Dana.Lahat,Cedric.Fevotte}@irit.fr

***Abstract*—This paper deals with algorithms for positive semidefinite matrix factorization (PSDMF). PSDMF is a recently-proposed extension of nonnegative matrix factorization with applications in combinatorial optimization, among others. In this paper, we focus on improving the local convergence of an alternating block gradient (ABG) method for PSDMF in a noise-free setting by replacing the quadratic objective function with the Poisson log-likelihood. This idea is based on truncated Wirtinger flow (TWF), a phase retrieval (PR) method that trims outliers in the gradient and thus regularizes it. Our motivation is a recent result linking PR with PSDMF. Our numerical experiments validate that the numerical benefits of TWF may carry over to PSDMF despite the more challenging setting, when initialized within its region of convergence. We then extend TWF from PR to affine rank minimization (ARM), and show that although the outliers are no longer an issue in the ARM setting, PSDMF with the new objective function may still achieves a smaller error for the same number of iterations. In a broader view, our results indicate that a proper choice of objective function may enhance convergence of matrix (or tensor) factorization methods.**

***Index Terms*—Positive semidefinite factorization, nonnegative factorizations, phase retrieval, low-rank approximations, optimization**

I. Introduction

This paper deals with improving the local convergence of algorithms for positive semidefinite matrix factorization (PSDMF) [1], [2]. In PSDMF, the (i,j)th entry x_{ij} of a nonnegative matrix $\mathbf{X} \in \mathbb{R}^{I\times J}$ is written as an inner product of two $K \times K$ symmetric positive semidefinite (psd) matrices $\mathbf{A}_i$ and $\mathbf{B}_j$, indexed by $i = [I]$, $j = [J]$:

$$x_{ij} \cong \langle \mathbf{A}_i, \mathbf{B}_j \rangle \triangleq \mathrm{tr}\{\mathbf{A}_i^\top \mathbf{B}_j\} = \mathrm{tr}\{\mathbf{A}_i \mathbf{B}_j\} \qquad (1)$$

where $[I] \triangleq 1,\ldots,I$, $\mathrm{tr}\{\cdot\}$ denotes the trace of a matrix, and $\cong$ means equal or approximately equal, depending on the context. The minimal number K such that a nonnegative matrix $\mathbf{X}$ admits an exact PSDMF is called the *psd rank* of $\mathbf{X}$ [1]. Each psd matrix $\mathbf{A}_i$ and $\mathbf{B}_j$ may have a different rank, denoted R_{A_i} and R_{B_j}, respectively. We shall sometimes refer to the psd rank K as the "outer rank" and to R_{A_i} and R_{B_j} as "inner ranks" [3]. When $\mathbf{A}_i$ and $\mathbf{B}_j$ are constrained to be diagonal matrices $\forall i,j$, PSDMF becomes equivalent to nonnegative matrix factorization (NMF) (e.g., [4]).

PSDMF was proposed as an extension to a fundamental result [5] in combinatorial optimization that links NMF with geometry. Since it was first introduced, PSDMF attracted a significant amount of attention and has found applications in numerous fields, including combinatorial optimization [2], [6], quantum communication complexity [2], quantum information theory [7], and recommender systems [8]. Despite the increasing interest in PSDMF, only a few algorithms have been proposed so far, e.g., [3], [8], [9]. In this paper, we do not focus on a specific application but on understanding some of the optimization properties of PSDMF, in a more general perspective.

This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No. 681839 (project FACTORY).

Our work is motivated by a recent result [10] showing a link between PSDMF and affine rank minimization (ARM) (e.g., [11], [12]), and in particular with phase retrieval (PR) (e.g., [13]), which can be considered as a special case of ARM. We explain their link to PSDMF briefly in Section II, which is dedicated to background material.

In [10], we presented a new type of local optimization scheme for PSDMF that is based on alternating block gradient (ABG). In this ABG scheme, each subproblem is based on Wirtinger flow (WF) [13] or ARM [11], [12], depending on the values of the inner ranks, and the objective function is based on quadratic loss.

This paper is motivated by truncated Wirtinger flow (TWF) [14], [15], a variant of WF obtained by replacing the quadratic objective function with a non-quadratic one, together with special regularization based on *truncating* excessive values from the gradient and the objective function. These excessive values do not appear in the quadratic objective. TWF improves the convergence rate also in the noise-free case, and requires appropriate initial conditions. In this paper, starting from Section III, we build an ABG algorithm in which each subproblem is based on the principles of TWF, and study to which extent the ideas of TWF can be applied to PSDMF.

Our main result is that in the case where all inner ranks are equal to 1, a setting equivalent to locally minimizing a TWF criterion in each subproblem, we observe phenomena analogous to those in TWF as predicted by our geometrical analysis in Section IV, provided we are in the basin of attraction of a global minimum. This special case of PSDMF, known as *Hadamard square root* or *phaseless rank*, is of particular interest (e.g., [3], [16], [17]). When some of the inner ranks are larger than one, there is no need for truncation, as we show in Section IV. Yet, we still observe faster convergence to the global minimum with our non-quadratic objective, a matter than needs to be further looked into.

Although we focus on the Poisson log-likelihood, as in [15], we keep in mind that the same principles are likely to hold also for other objective functions; see further discussion in Section VI. To the best of our knowledge, this paper is the first to suggest that changing the objective function within the same algorithmic framework in a matrix factorization problem can result in improved numerical properties, in the noise-free case, provided proper initialization and regularization (e.g., truncation) when necessary.

In this paper, we use real-valued notations. However, all our results extend straightforwardly to the complex-valued case.

II. A Link Between PSDMF and Phase Retrieval

In PSDMF applications, the psd matrices are often of low rank, i.e., $R_{A_i}, R_{B_j} < K$ (e.g., [17]). Hence, the psd matrices can be written as

$$\mathbf{A}_i \triangleq \mathbf{U}_i\mathbf{U}_i^\mathsf{T} \quad \text{and} \quad \mathbf{B}_j \triangleq \mathbf{V}_j\mathbf{V}_j^\mathsf{T}\,, \tag{2}$$

where $\mathbf{U}_i \in \mathbb{R}^{K\times R_{A_i}}$ and $\mathbf{V}_j \in \mathbb{R}^{K\times R_{B_j}}$ are full-rank factor matrices (factors, for short). With this change of variables, the model in (1) can be written as

$$x_{ij} \cong \langle \mathbf{U}_i\mathbf{U}_i^\mathsf{T}, \mathbf{V}_j\mathbf{V}_j^\mathsf{T}\rangle = \operatorname{tr}\{\mathbf{U}_i\mathbf{U}_i^\mathsf{T}\mathbf{V}_j\mathbf{V}_j^\mathsf{T}\} = \|\mathbf{U}_i^\mathsf{T}\mathbf{V}_j\|_F^2\,. \tag{3}$$

In the special case where $J = 1$, (3) can be written as

$$y_i \cong \|\mathbf{U}_i^\mathsf{T}\mathbf{V}\|_F^2\,, \tag{4}$$

where we replaced $\mathbf{V}_j$ with $\mathbf{V}$ and x_{ij} with y_i. When $\mathbf{U}_i$ are given, finding $\mathbf{V}$ from y_i, $i = [I]$, in (4) is equivalent to ARM (e.g., [11], [12]). In this case, $\mathbf{U}_i$ can be regarded as *sensing matrices* and $\mathbf{V}$ as the signal of interest. If all factors are vectors: $\mathbf{v} \in \mathbb{R}^K$ and $\mathbf{u}_i \in \mathbb{R}^K$ $\forall i$, (4) can be written as

$$y_i \cong \langle \mathbf{u}_i\mathbf{u}_i^\mathsf{T}, \mathbf{v}\mathbf{v}^\mathsf{T}\rangle = |\mathbf{u}_i^\mathsf{T}\mathbf{v}|^2 = |\langle \mathbf{u}_i, \mathbf{v}\rangle|^2\ ,\ i = [I]\,. \tag{5}$$

When all vectors $\mathbf{u}_i$ are given, the problem of estimating $\mathbf{v}$ from the phaseless observations y_i in (5) is known as (generalized) PR [18]. In a PR context, $\mathbf{u}_i$ are sometimes referred to as *sensing vectors* and $\mathbf{v}$ as a *signal* of interest. We conclude that PR (5) and ARM (4) are special cases of PSDMF [10].

Back from PR or ARM to PSDMF: Consider a random noise model acting independently on each observation in (4). Then, for a broad range of noise models, the log-likelihood of the model parameters given an observation can be written as $\mathrm{d}(y_i, \|\mathbf{U}_i^\mathsf{T}\mathbf{V}\|_F^2)$, where $\mathrm{d}(\zeta,\eta)$ is a divergence between any two nonnegative numbers ζ,η, and where $\mathrm{d}(\zeta,\eta) \geq 0$ with $\mathrm{d}(\zeta,\eta) = 0$ if and only if $\zeta = \eta$. For example, $\mathrm{d}(\zeta,\eta) = \frac{1}{2}(\zeta-\eta)^2$ leads to the quadratic objective function. Then, a maximum likelihood estimate (MLE) to the ARM problem in (4) can be found by minimizing, with respect to (w.r.t.) $\mathbf{V}$, the objective function

$$\sum_{i=1}^{I} \mathrm{d}(y_i, \|\mathbf{U}_i^\mathsf{T}\mathbf{V}\|_F^2)\,. \tag{6}$$

Similarly, based on (3), an MLE to PSDMF can be found by minimizing, w.r.t. $\mathbf{U}_i$ and $\mathbf{V}_j$, $\forall i,j$, the objective function

$$\sum_{i=1}^{I}\sum_{j=1}^{J} \mathrm{d}(x_{ij}, \|\mathbf{U}_i^\mathsf{T}\mathbf{V}_j\|_F^2)\,. \tag{7}$$

Looking at (6) and (7), we see how any ARM method for (6) can be used to optimize (7) : given $\{\mathbf{U}_i\}_{i=1}^I$, apply ARM subsequently to update each $\mathbf{V}_j$ in (7). Then change roles: fix $\{\mathbf{V}_j\}_{j=1}^J$ and apply the method to update $\mathbf{U}_i$ subsequently for each i. Repeat until a stopping criterion is achieved. The alternating optimization strategy for PSDMF has already been proposed, e.g., in [3], [8], [9], [19] . The link between these methods and PR is pointed out in [10]. The ABG method in [10] is based on decreasing the objective function in (6) by gradient descent with a quadratic objective function, as in [11], [12]. Addressing the PR problem associated with (5) via gradient descent with a quadratic objective function is termed WF [13].

III. Extending TWF to ARM

Starting from this section, we gradually build an ABG method for PSDMF based on TWF [15]. For the sake of simplicity, we focus on the Poisson log-likelihood, as in [15]. In this case, the objective function in (7) is based on the Kullback-Leibler divergence (KLD): $\mathrm{d}_{\mathrm{KL}}(\zeta,\eta) \triangleq \zeta\log(\frac{\zeta}{\eta}) - \zeta + \eta$. PSDMF with a Poisson log-likelihood thus consists of decreasing the objective function

$$f = \sum_{i=1}^{I}\sum_{j=1}^{J} \|\mathbf{U}_i^\mathsf{T}\mathbf{V}_j\|_F^2 - x_{ij}\log(\|\mathbf{U}_i^\mathsf{T}\mathbf{V}_j\|_F^2) + x_{ij}\log(x_{ij}) - x_{ij} \tag{8}$$

whose gradient w.r.t. $\mathbf{V}_j \in \mathbb{C}^{K\times R_{B_j}}$ is

$$\nabla_{\mathbf{V}_j} f = 2\sum_{i=1}^{I} \frac{\left(\|\mathbf{U}_i^\mathsf{T}\mathbf{V}_j\|_F^2 - x_{ij}\right)}{\|\mathbf{U}_i^\mathsf{T}\mathbf{V}_j\|_F^2}\mathbf{U}_i\mathbf{U}_i^\mathsf{T}\mathbf{V}_j\,. \tag{9}$$

In the complex-valued case, change $\cdot^\mathsf{T}$ to $\cdot^\mathsf{H}$ and divide $\nabla_{\mathbf{V}_j} f$ by 2. For a specific j, (8) and (9) are the low-rank generalizations of the Poisson objective and gradient in TWF [15]. One can thus obtain ARM and PSDMF algorithms based on the Poisson log-likelihood by replacing the quadratic objective function and gradient in [10]–[12] with those in (8) and (9).

Our first step, of extending TWF from rank-1 signal recovery to ARM, and of stating PSDMF with a Poisson log-likelihood, is thus complete. We point out that [15, Sec. 8] mention extending TWF to ARM, but only when the sensing vectors are rank-1. However, due to the symmetric nature of the subproblems, in the general PSDMF case we do not restrict the inner ranks in any of the factors.

IV. A Geometric Explanation

The second step in extending TWF to PSDMF is revisiting the truncation idea. For this purpose, we follow in the steps of [15] and inspect the locus of the gradient in (9) at the proximity of the solution.

We begin with the case where all factors are vectors, which is equivalent to the setting in TWF. The locus of the gradient in (9) for a specific value of j is illustrated in Fig. 1 (left). This figure is a reconstruction of [15, Figure 2.1] using the same parameters. It shows that the desired values of the gradient, around the origin, are rather small (see inset) compared with the outliers, which are few but very large. These large and undesired gradient values occur only in specific directions. A similar examination will reveal that the quadratic objective function does not have outliers (the gradient for the quadratic case is similar to (9), but without the denominator).

The idea of TWF [15] is to avoid these outliers and thus better focus on the desired values. Chen and Candès provide coefficients that allow to detect elements in the gradient and the objective function that are associated with undesired contributions. They show that this procedure achieves faster convergence and better resilience to noise than without truncation, provided an appropriate initialization.

Next, we study the outlier problem in the general low-rank case. As an example, we look at the locus of the gradient in (9) when $\mathbf{U}_i$ are now 2×2 "sensing matrices" instead of 2×1 vectors. The "signal" vector is the same as in the previous plot. Figure 1 (right) depicts the locus. We observe that the outliers are significantly smaller and fewer. As in the one-dimensional example in Fig. 1 (left), the outliers occur only in specific directions. Otherwise, the shape of the locus did not change much. We observed similar trends with other values of ranks and matrix sizes. This indicates that truncation will have little to no effect as one gets further away from the case where all inner ranks are equal to one. Our numerical results in Section VI validate this.

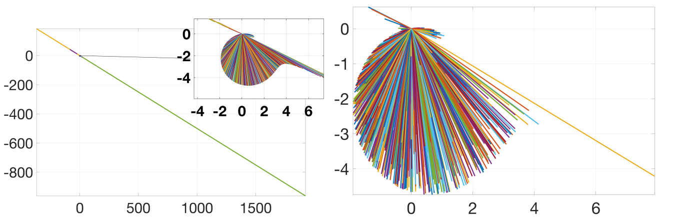

Fig. 1. Comparison of size of outliers in gradient for different dimensions of $\mathbf{U}_i$. Each plot depicts the locus of the gradient $\left(\|\mathbf{U}_i^\mathsf{T}\mathbf{z}\|_F^2 - \|\mathbf{U}_i^\mathsf{T}\mathbf{v}\|_F^2\right) \|\mathbf{U}_i^\mathsf{T}\mathbf{z}\|_F^{-2}\mathbf{U}_i\mathbf{U}_i^\mathsf{T}\mathbf{v}$ with $\mathbf{v} = \begin{bmatrix}2.7 & 8\end{bmatrix}^\mathsf{T}$, $\mathbf{z} = \begin{bmatrix}3 & 6\end{bmatrix}^\mathsf{T}$. The entries of $\mathbf{U}_i$ are drawn independently from $\mathcal{N}(0,1)$ and then normalized s.t. $\|\mathbf{U}_i\|_F = 1$. Each plot depicts 10^3 independent draws of $\mathbf{U}_i$. In each plot, the axes are fit to the data, no zoom and no cropping (except inset in left plot). $\mathbf{U}_i$ is a 2×1 vector as in [15, Fig. 2.1] (left) and a 2×2 matrix (right).

V. A PSDMF Algorithm Based on TWF

Algorithm 1 describes one subproblem in our TWF-based ABG scheme for updating $\{\mathbf{V}_j\}_{j=1}^J$ given $\{\mathbf{U}_i\}_{i=1}^I$. In the special case where all inner ranks are equal to one, Algorithm 1 is essentially equal to TWF [15].

The outliers discussed in Section IV are avoided by using a regularized version of the gradient, $\nabla_{\mathbf{V}_j} f_{\text{tr}}$, in Algorithm 1. The regularization $\mathbb{I}_{\mathcal{E}_1^{ij} \cap \mathcal{E}_2^{ij}}$ applied to the ith element of the gradient sets it to zero (i.e., does not add it to the sum) if at least one of the following events occurs:

$$\mathcal{E}_1^{ij} \triangleq \left\{\alpha^{\text{lb}} \leq \frac{\|\mathbf{U}_i^\mathsf{T}\mathbf{V}_j\|_F}{\|\mathbf{V}_j\|_F} \leq \alpha^{\text{ub}}\right\} \tag{10}$$

$$\mathcal{E}_2^{ij} \triangleq \left\{\left|\|\mathbf{U}_i^\mathsf{T}\mathbf{V}_j\|_F^2 - y_i\right| \leq \frac{\alpha^{\text{h}}\|\mathbf{U}_i^\mathsf{T}\mathbf{V}_j\|_F \cdot \|\mathbf{q}_j - \mathbf{x}_j\|_1}{I\|\mathbf{V}_j\|_F}\right\}$$

where $\mathbf{x}_j$ is the jth column of $\mathbf{X}$ and $\mathbf{q}_j$ is a vector whose ith entry is equal to $\|\mathbf{U}_i^\mathsf{T}\mathbf{V}_j\|_F^2$. The truncation rules in (10) generalize the truncation rules in [15] to the low-rank case.

If backtracking line search is used, [15, Sec. 2.3] suggest a dedicated truncation procedure. Algorithm 2 extends it to the general low-rank case. In Algorithm 2, in analogy to [15, Sec. 2,3], $\mathbf{P}_j \triangleq \nabla_{\mathbf{V}_j} f_{\text{tr}}$, and

$$\widetilde{f}(\mathbf{V}_j) \triangleq \sum_{i \in \widetilde{\mathcal{T}}(\mathbf{V}_j)} \left(\|\mathbf{U}_i^\mathsf{T}\mathbf{V}_j\|_F^2 - x_{ij}\log(\|\mathbf{U}_i^\mathsf{T}\mathbf{V}_j\|_F^2)\right) \tag{11a}$$

$$\text{where}\quad \widetilde{\mathcal{T}}(\mathbf{V}_j) \triangleq \Big\{i \mid \|\mathbf{U}_i^\mathsf{T}\mathbf{V}_j\|_F \geq \alpha^{\text{lb}}\|\mathbf{V}_j\|_F \quad \text{and} \quad \|\mathbf{U}_i^\mathsf{T}\mathbf{P}_j\|_F \leq \alpha^{\text{p}}\|\mathbf{P}_j\|_F\Big\}. \tag{11b}$$

Equation (11) describes the truncation rules for the objective function in the backtracking line search procedure in Algorithm 2, which generalize those in [15, Sec. 2.3] to the case where both "sensing vectors" and "signal" may be matrices.

Algorithm 1 Updating $\{\mathbf{V}_j\}_{j=1}^J$ based on TWF

```
Input: $\mathbf{X} \in \mathbb{R}_+^{I \times J}$, $\mathbf{U}_1, \dots, \mathbf{U}_I$, $\mathbf{V}_1, \dots, \mathbf{V}_J$, $\alpha^{\text{ub}}$, $\alpha^{\text{lb}}$, $\alpha^{\text{h}}$.
Output: $\mathbf{V}_1, \dots, \mathbf{V}_J$.
1: $\tau \leftarrow$ Choose initial step size
2: for $j = 1 : J$ do
3:     $\nabla_{\mathbf{V}_j} f_{\text{tr}} \leftarrow 2\sum_{i=1}^I \frac{(\|\mathbf{U}_i^\mathsf{T}\mathbf{V}_j\|_F^2 - x_{ij})}{\|\mathbf{U}_i^\mathsf{T}\mathbf{V}_j\|_F^2}\mathbf{U}_i\mathbf{U}_i^\mathsf{T}\mathbf{V}_j\ \mathbb{I}_{\mathcal{E}_1^{ij}\cap\mathcal{E}_2^{ij}}$
4:     $\mu_j \leftarrow$ Choose step size (e.g., $\tau$ or Algorithm 2)
5:     $\mathbf{V}_j \leftarrow \mathbf{V}_j - \frac{\mu_j}{I}\nabla_{\mathbf{V}_j} f_{\text{tr}}$
6: end for
```

Algorithm 2 Backtracking line search

```
Input: $\tau$, $\alpha_{\text{bck}}$, $\beta_{\text{bck}}$, $\alpha^{\text{lb}}$, $\alpha^{\text{p}}$
1: while $\widetilde{f}(\mathbf{V}_j - \tau\mathbf{P}_j) > \widetilde{f}(\mathbf{V}_j) - \alpha_{\text{bck}}\tau\|\mathbf{P}_j\|_F^2$ do
2:     $\tau = \beta_{\text{bck}}\tau$
3: end while
```

VI. Numerical Experiments

We compare our proposed Poisson-based ABG, with and without truncation, to ABG [10] and to the two variants of coordinate descent (CD) [3], cyclic and Gauss-Southwell (GS), the latter with parameter $\alpha^{\text{GS}} = 0.5$. The competing methods: ABG and CD, use a Gaussian objective function. We set $\alpha_{\text{bck}} = 0.1$, $\beta_{\text{bck}} = 0.35$ for backtracking line search (Algorithm 2) for ABG [10] and for our proposed method. The initial step size for the backtracking line search is chosen by evaluating numerically the Lipschitz constant of the gradient, as described in [10]. The truncation parameters are identical to those suggested as default in [15, Sec. 2.3]: $\alpha^{\text{lb}} = 0.1$, $\alpha^{\text{ub}} = 5 = \alpha^{\text{p}}$, $\alpha^{\text{h}} = 6$.

In our first numerical experiment, whose results are shown in Fig. 2, the input is a 20×20 nonnegative matrix ($I = 20 = J$) generated from factors whose entries are drawn independently from $\mathcal{N}(0, 1)$. The factors have $K = 5$ rows. We test two settings with different values of inner ranks: (i) $R_A = 1 = R_B$ (ii) $R_A = 2$, $R_B = 1$, where $R_{A_i} = R_A$ $\forall i$ and $R_{B_j} = R_B$ $\forall j$. In each of the 200 Monte Carlo (MC) trials, the input matrix $\mathbf{X}$ and initial factors are drawn anew, and all algorithms use the same initialization. Our input matrix provides 400 observations, and the corresponding numbers of free model variables for each setting are 175, and 255, respectively. Hence, these are low-rank scenarios. The algorithms are given the true values of the ranks. Hence, achieving a global minimum is equivalent to exact factorization. The stopping criterion is 60 iterations. We define as one iteration t a pair of subproblems that update both sets of factors, indexed by i and j. Our algorithms are implemented in Matlab. We use the CD code given in https://sites.google.com/site/exactnmf/. We scale the initial factors, before running the algorithms, using the procedure in [3].

Initialization: As noted by [3], the loss function for PSDMF (7) is in general highly nonconvex. As a result, the chance of achieving a global minimum with random initialization is very small in low-rank settings and in problems whose size is not sufficiently small. Spectral initialization (e.g., [11]–[13], [15]), which is essential for TWF, is not applicable to PSDMF because the roles of "sensing matrix" and "signal" change at every subproblem, and both are unknowns that have to be estimated from the data. For these reasons, and for the sake of our preliminary study of our proposed methods, we chose another strategy, which initializes the algorithms within a sufficiently small region around the true solution. Each entry in an initialization factor is the sum of $(1 - p)$ times the true entry and p times an independently drawn number from $\mathcal{N}(0, 1)$, and the overall is normalized such that the variance of this new random variable (since we draw the entries of the true factors from $\mathcal{N}(0, 1)$, too) is one. This normalization is chosen to guarantee that we work in a regime that allows for exact recovery in WF/TWF setting (e.g., [13], [15]). We set $p = 0.1$, as this value turned out to guarantee convergence of (most of) our tests of the truncated Poisson-based objective.

The purpose of the first experiment is to compare the performance of the two objective functions when the algorithm is already within its (assumed) region of convergence (ROC). Due to lack of space, other experiments, such as determining the size of the ROC of each method and the rate of success with respect to the various model parameters (matrix size, inner and outer ranks), as well as more realistic initialization strategies, are beyond the scope of this paper.

Results: Figure 2 shows our results. Each boxplot represents the logarithm of the normalized model fit error $\log_{10}(\frac{\|\mathbf{X}-\mathbf{X}_t\|_F}{\|\mathbf{X}_t\|_F})$, for each method. The plot on the left in Fig. 2 corresponds to $R_A = 1 = R_B$, which is the direct extension of TWF to PSDMF. The plot on the right, with

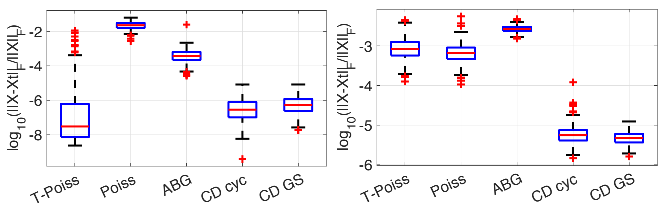

Fig. 2. Comparison of model fit error for a fixed number of iterations. PSDMF of a 20×20 matrix with randomly-generated factors. Initialization with perturbed true factors. Left: $R_A = 1 = R_B$, right: $R_A = 2$, $R_B = 1$.

$R_A = 2$, $R_B = 1$ corresponds to our ARM extension of TWF.

The results for $R_A = 1 = R_B$ are generally in accordance with TWF [15]. The smallest error is achieved by our TWF-based method. Non-truncated Poisson stopped with a relatively large error, indicating poor convergence due to outliers. ABG, which is the counterpart of our Poisson-based methods but with a Gaussian objective function, performs better than the Poisson-based objective with no truncation, but the error is still relatively large. The competing CD methods achieved the second-best results. We point out that the computational complexity of the Poisson log-likelihood variant of ABG is similar to that of quadratic ABG; it is identical in the case of all inner ranks equal to 1. The per-subproblem computational complexity of ABG [10] and its Poisson log-likelihood variant is smaller than that of CD [3].

The outliers (red crosses) indicating relatively large errors in the truncated Poisson case probably imply that the initialization is close to the boundary of the ROC, and thus not all trials converged towards a global minimum. In general, the results in Fig. 2 indicate that the outlier phenomenon observed in TWF [15] indeed appears also in PSDMF, despite the evident differences between TWF and PSDMF.

We now turn to the low-rank case, addressed in the right plot of Fig. 2. In this setting, we do not observe an advantage of truncated versus non-truncated Poisson, in accordance with our geometric analysis in Section IV and Fig. 1. Still, the Poisson objective managed to achieve a smaller error per fixed number of iterations than its Gaussian counterpart. In this setting, the CD methods achieved a smaller error than our gradient-based methods for the same number of iterations. However, one should keep in mind that CD has a larger computational complexity per iteration, see [3], [10]. Similar trends were observed when we increased the values of the inner ranks.

Our second experiment shows the effect of the Poisson log-likelihood on a different data set, a slack matrix of the regular 10-gon (a ten-sided polygon), denoted as S_{10}. S_{10} is a 10×10 matrix, whose explicit form can be found, e.g., in [3]. PSDMF of geometric data, and slack matrices in particular, is a central application of PSDMF [1], [3]. Our input matrix $\mathbf{X}$ is S_{10} normalized such that $\|\mathbf{X}\|_F = 1$. Here, we use random initialization: initial factors with dimensions $K = 5$, $R_A = 1$, $R_B = 3$, whose entries are drawn independently from $\mathcal{N}(0, 1)$.

Results: Figure 3 (left) shows our results in terms of the number of iterations each algorithm needed to achieve the same normalized model fit error $\frac{\frac{1}{2}\|\mathbf{X}-\mathbf{X}_t\|_F^2}{\|\mathbf{X}_t\|_F} < 10^{-7}$. We run 50 MC trials, each with a new initialization that is the same for all algorithms. Figure 3 (right) illustrates the evolution of the model fit error of each algorithm for one run in this setting; here, all algorithms are implemented in Matlab. Figure 3 does not show the results for truncated Poisson because this method did not converge properly, in this setting. The reason is probably not only the lack of outliers in this low-rank setting, as explained in Section IV, but also due to the fact that the truncation constants proposed by [15] were designed for factors whose entries are drawn from the standard normal distribution $\mathcal{N}(0, 1)$, an assumption that does not hold in the case of geometric data. In Fig. 3 (left), we observe that our proposed method with the Poisson objective (and no truncation) often achieves the desired model fit error with the fewest number of iterations; note that the per-iteration complexity must be taken into account. The CPU time comparison in Fig. 3 (right) is only illustrative because our Matlab implementation is not optimized for speed. Still, it indicates that the proposed method can converge much faster than its competitors, consistently with Fig. 3 (left).

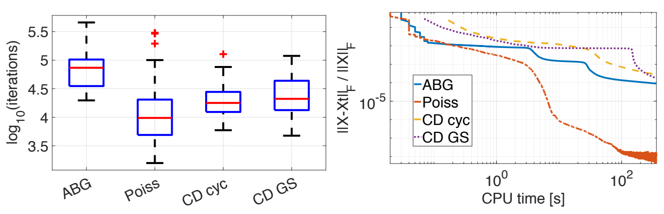

Fig. 3. PSDMF of S_{10} with $K = 5$, $R_A = 1$, $R_B = 3$. Random initialization. Left: Comparison of number of iterations for the same model fit error. Right: Evolution of error versus CPU time in one run.

Conclusion: Our results provide proof of concept to the idea of improving the convergence of exact PSDMF by changing the objective function. However, our results depend heavily on the type of data and on initialization within the ROC, in certain cases. There is still need for further study of the ROC of each PSDMF method and its dependence on the model parameters, ranks and dimensions. Our results call for further theoretical validation. Our results provide further motivation to design more efficient PSDMF methods with better global convergence. A possible direction is more recent techniques used in PR, see, e.g., [20] and references therein.

We mention that Gaussian and Poisson log-likelihoods are special cases of the β-divergence, a popular tool in NMF (e.g., [21]). Extending our algorithm to β-divergence is straightforward: the only change in the gradient in (9) is the power of the denominator, which becomes $(2-\beta)$ instead of 1. One can readily verify, by plotting diagrams similar to those in Fig. 1 for other values of β, that the size of the outliers is continuous in the value of β. It is thus worth exploring the benefit of truncation and use of other objectives also for other values of β in all concerned applications, namely PR, ARM, and PSDMF. This paper addressed only the noise-free case: study of the usefulness of PSDMF with non-Gaussian objectives in the presence of noise is left for follow-up work. In a broader view, the idea of accelerating factorizations by changing the objective function and truncating outliers might allow to accelerate other methods, and thus serve purposes beyond providing better resilience to non-Gaussian types of noise.

Acknowledgment

The authors would like to thank Vincent Y. F. Tan for bringing the work on TWF to our attention.

References

[1] J. Gouveia, P. A. Parrilo, and R. R. Thomas, "Lifts of convex sets and cone factorizations," *Mathematics of Operations Research*, vol. 38, no. 2, pp. 248–264, May 2013.

[2] S. Fiorini, S. Massar, S. Pokutta, H. R. Tiwary, and R. De Wolf, "Linear vs. semidefinite extended formulations: exponential separation and strong lower bounds," in *Proc. STOC*, May 2012, pp. 95–106.

[3] A. Vandaele, F. Glineur, and N. Gillis, "Algorithms for positive semidefinite factorization," *Computational Optimization and Applications*, vol. 71, no. 1, pp. 193–219, Sep 2018.

[4] P. Paatero and U. Tapper, "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, Jun. 1994.

[5] M. Yannakakis, "Expressing combinatorial optimization problems by linear programs," *Journal of Computer and System Sciences*, vol. 43, no. 3, pp. 441–466, Dec. 1991.

[6] V. Kaibel, "Extended formulations in combinatorial optimization," *arXiv:1104.1023 [math.CO]*, Apr. 2011.

[7] R. Jain, Y. Shi, Z. Wei, and S. Zhang, "Efficient protocols for generating bipartite classical distributions and quantum states," *IEEE Trans. Inf. Theory*, vol. 59, no. 8, pp. 5171–5178, Aug 2013.

[8] C. J. Stark, "Recommender systems inspired by the structure of quantum theory," *arXiv:1601.06035 [cs.LG]*, 2016.

[9] I. Glasser, R. Sweke, N. Pancotti, J. Eisert, and I. Cirac, "Expressive power of tensor-network factorizations for probabilistic modeling," in *Proc. NeurIPS*. Curran Associates, Inc., 2019, pp. 1496–1508.

[10] D. Lahat and C. Févotte, "Positive semidefinite matrix factorization: A link to phase retrieval and a block gradient algorithm," in *Proc. ICASSP*, Barcelona, Spain, May 2020.

[11] Q. Zheng and J. Lafferty, "A convergent gradient descent algorithm for rank minimization and semidefinite programming from random linear measurements," in *Proc. NeurIPS*, 2015, pp. 109–117.

[12] S. Tu, R. Boczar, M. Simchowitz, M. Soltanolkotabi, and B. Recht, "Low-rank solutions of linear matrix equations via Procrustes flow," in *Proc. ICML*, ser. Proceedings of Machine Learning Research, vol. 48. New York, New York, USA: PMLR, Jun. 2016, pp. 964–973.

[13] E. J. Candès, X. Li, and M. Soltanolkotabi, "Phase retrieval via Wirtinger flow: Theory and algorithms," *IEEE Trans. Inf. Theory*, vol. 61, no. 4, pp. 1985–2007, April 2015.

[14] Y. Chen and E. Candès, "Solving random quadratic systems of equations is nearly as easy as solving linear systems," in *Proc. NeurIPS*. Curran Associates, Inc., 2015, pp. 739–747.

[15] Y. Chen and E. Candès, "Solving random quadratic systems of equations is nearly as easy as solving linear systems," *Communications on Pure and Applied Mathematics*, vol. 70, no. 5, pp. 822–883, 2017.

[16] A. P. Goucha and J. Gouveia, "The phaseless rank of a matrix," *arXiv:1909.02417 [math.AG]*, 2019.

[17] H. Fawzi, J. Gouveia, P. A. Parrilo, R. Z. Robinson, and R. R. Thomas, "Positive semidefinite rank," *Mathematical Programming*, vol. 153, no. 1, pp. 133–177, Oct 2015.

[18] R. W. Gerchberg and W. O. Saxton, "A practical algorithm for the determination of phase from image and diffraction plane pictures," *Optik*, vol. 35, no. 2, pp. 237–246, 1972.

[19] A. Basu, M. Dinitz, and X. Li, "Computing approximate PSD factorizations," vol. 60, Dagstuhl, Germany, Sep. 2016, pp. 2:1–2:12.

[20] Y. Yang, M. Pesavento, Y. C. Eldar, and B. Ottersten, "Parallel coordinate descent algorithms for sparse phase retrieval," in *Proc. ICASSP*, Brighton, UK, May 2019, pp. 7670–7674.

[21] C. Févotte and J. Idier, "Algorithms for nonnegative matrix factorization with the β-divergence," *Neural Comput.*, vol. 23, no. 9, pp. 2421–2456, Jun. 2011.