A Non-commutative Extension of Lee-Seung's Algorithm for Positive Semidefinite Factorizations

Yong Sheng Soh
Department of Mathematics
National University of Singapore
10 Lower Kent Ridge Road, Singapore 119076
and
Institute of High Performance Computing
1 Fusionopolis Way, #16-16 Connexis, Singapore 138632
matsys@nus.edu.sg

Antonios Varvitsiotis
Engineering and Systems Design Pillar
Singapore University of Technology and Design
8 Somapah Road, Singapore 487372
avarvits@gmail.com

Abstract

Given a data matrix $X \in \mathbb{R}_+^{m\times n}$ with non-negative entries, a Positive Semidefinite (PSD) factorization of X is a collection of $r \times r$-dimensional PSD matrices $\{A_i\}$ and $\{B_j\}$ satisfying the condition $X_{ij} = \mathrm{tr}(A_i B_j)$ for all $i \in [m],\ j \in [n]$. PSD factorizations are fundamentally linked to understanding the expressiveness of semidefinite programs as well as the power and limitations of quantum resources in information theory. The PSD factorization task generalizes the Non-negative Matrix Factorization (NMF) problem in which we seek a collection of r-dimensional non-negative vectors $\{a_i\}$ and $\{b_j\}$ satisfying $X_{ij} = a_i^T b_j$, for all $i \in [m],\ j \in [n]$ – one can recover the latter problem by choosing matrices in the PSD factorization to be diagonal. The most widely used algorithm for computing NMFs of a matrix is the Multiplicative Update algorithm developed by Lee and Seung, in which non-negativity of the updates is preserved by scaling with positive diagonal matrices. In this paper, we describe a non-commutative extension of Lee-Seung's algorithm, which we call the Matrix Multiplicative Update (MMU) algorithm, for computing PSD factorizations. The MMU algorithm ensures that updates remain PSD by congruence scaling with the matrix geometric mean of appropriate PSD matrices, and it retains the simplicity of implementation that the multiplicative update algorithm for NMF enjoys. Building on the Majorization-Minimization framework, we show that under our update scheme the squared loss objective is non-increasing and fixed points correspond to critical points. The analysis relies on Lieb's Concavity Theorem. Beyond PSD factorizations, we show that the MMU algorithm can be also used as a primitive to calculate block-diagonal PSD factorizations and tensor PSD factorizations. We demonstrate the utility of our method with experiments on real and synthetic data.

1 Introduction

Let $X \in \mathbb{R}_+^{m\times n}$ be a $m \times n$ dimensional matrix with non-negative entries and $r \in \mathbb{N}$ a user-specified parameter. An *r-dimensional positive semidefinite (PSD) factorization* of X is given by two families of $r \times r$ PSD matrices $A_1, \ldots, A_m$ and $B_1, \ldots, B_n$ satisfying

$$X_{ij} = \operatorname{tr}(A_i B_j),\ i \in [m],\ j \in [n]. \tag{1}$$

Every non-negative matrix admits an r-dimensional PSD factorization for an appropriate value of $r \in \mathbb{N}$–we may, for instance, take $A_i = \operatorname{diag}(X_{i:})$ and $B_j = \operatorname{diag}(e_j)$, a choice that corresponds to an n-dimensional PSD factorization. The smallest $r \in \mathbb{N}$ for which X admits an r-dimensional PSD factorization is called the *PSD-rank* [6].

PSD factorizations are of fundamental importance to a wide range of areas, most notably towards understanding the expressive power of linear optimization over the cone of positive semidefinite matrices, [9, 14], studying the power and limitations of quantum resources within the framework of information theory [9, 17], and as a natural non-commutative generalization of the extremely popular dimensionality reduction technique of non-negative matrix factorizations (NMFs) [25, 26, 32]. We elaborate further on the relevance of PSD factorizations to each of these areas below.

Links to Semidefinite Programming. A Semidefinite Program (SDP) is a convex optimization problem in which we minimize a linear function over the set of PSD matrices intersected with an affine subspace. SDPs are a powerful generalization of Linear Programs with extensive modeling power and tractable algorithms for solving them, e.g., see [38] and references therein. SDPs are frequently used as convex relaxations to combinatorial problems, and have many important applications including optimal power flow computation [23], robustness certification to adversarial examples in neural networks [33], and inference in graphical models [5].

Given a bounded polytope $P = \{x \in \mathbb{R}^d : c_i^\top x \le d_i,\ i \in [\ell]\} = \operatorname{conv}(v_1, .., v_k)$, a basic question concerning the expressive power of SDPs is to find the *smallest* possible SDP description of P, i.e., the minimum $r \in \mathbb{N}$ for which we can express P as the projection of an the affine slice of the cone of $r \times r$ PSD matrices. Concretely, the goal in this setting is to express P as:

$$P = \pi(\mathbb{S}_+^r \cap \mathcal{L}), \tag{2}$$

where $\mathbb{S}_+^r$ is the cone of $r \times r$ PSD matrices, $\mathcal{L}$ is an affine subspace of the space of $r \times r$ symmetric matrices and π a linear projection from the space of $r \times r$ symmetric matrices to $\mathbb{R}^d$. A representation of the form (2) is called an *extended formulation* or *PSD-lift* of P and is extremely useful for optimization purposes. Indeed, the existence of a PSD-lift immediately implies that $\min\{\langle c, x\rangle : x \in P\} = \min\{\langle \pi^\top(c), y\rangle : y \in \mathbb{S}_+^r \cap \mathcal{L}\}$, and consequently, linear optimization over P (which can be hard) corresponds to an SDP (which can be solved efficiently).

To describe the connection of PSD factorizations with SDP-lifts, let S_P be the *slack matrix* of P, namely, S_P is a rectangular matrix whose rows are indexed by the facets of P, its columns indexed by extreme points v_j, and the ij-entry of S_P corresponds to the slack between the i-th facet and the j-th vertex, $S_{ij} = d_i - c_i^\top v_j$. Generalizing a seminal result by Yannakakis for LPs [40], it was shown independently in [9] and [14] that if S_P admits an r-dimensional PSD factorization, the polytope P admits a PSD-lift over the cone of $r \times r$ PSD matrices. The proof is also constructive – given a PSD factorization of S, there is an explicit description of $\mathcal{L}$ and π that gives rise to P.

An important special case of the PSD factorization problem is when the PSD factors are block-diagonal PSD matrices, where both the number of blocks and the size of each block is fixed, i.e., $A_i, B_j \in (\mathbb{S}_+^r)^k$. For a fixed and user-specified $r \in \mathbb{N}$, the least $k \in \mathbb{N}$ for which $X \in \mathbb{R}_+^{m\times n}$ admits a PSD factorization with PSD-factors in $(\mathbb{S}_+^r)^k$ is called the r-block diagonal PSD-rank of X. In terms of the geometric interpretation of PSD factorizations, block-diagonal PSD factorizations of the slack matrix S_P correspond to extended formulations of P over a Cartesian product of PSD cones, i.e., $P = \pi((\mathbb{S}_+^r)^k \cap \mathcal{L})$. In terms of relevance to optimization, extended formulations over $(\mathbb{S}_+^r)^k$ allow to perform linear optimization over P by solving block-diagonal SDPs, which can be solved numerically much faster compared to dense SDPs. In fact, most interior-point algorithms for SDPs are designed to exploit block-diagonal structure if it is present in the problem. The first systematic study of block-diagonal PSD-lifts was given in [7], where the focus was mainly on lower bounds.

Links to quantum information theory. Consider two parties, Alice and Bob, that try to generate samples (i, j) following some joint distribution $P(i, j)$. For this section, it is crucial to think of the distribution $P(i, j)$ as being arranged in an entrywise non-negative matrix, and we use P to simultaneously refer to both the distribution and its matrix representation. Clearly, if P is not a product distribution, Alice and Bob should either communicate or share some common information to be able to generate samples according to P. In the *correlation generation problem*, the goal is to find the least amount of shared resources that are needed to achieve this task. The considered resources can be either classical (shared randomness), quantum (shared entangled state) or hybrid.

In the quantum case, correlation generation boils down to finding a quantum state $\rho \in \mathbb{S}_+^{r^2}$ with $\mathrm{tr}(\rho) = 1$ and quantum measurements E_i, F_j (i.e., $E_i, F_j \in \mathbb{S}_+^r$ and $\sum_i E_i = \sum_j F_j = I$) such that

$$P(i, j) = \mathrm{tr}((E_i \otimes F_j)\rho), \ i \in [m], \ j \in [n]. \tag{3}$$

The least $r \in \mathbb{N}$ for which a factorization of the form (3) is possible is given by the (logarithm) of the psd-rank of the matrix P [17]. Moreover, the proof of [17] is constructive, in the sense that, given a r-dimensional PSD factorization of P, there is an explicit description of a quantum state $\rho \in \mathbb{S}_+^{r^2}$ and measurement operators acting on $\mathbb{C}^r$ that satisfy (3), e.g., see [6, Proposition 3.8].

Moving beyond purely quantum protocols, there has been recent interest in hybrid classical-quantum protocols, motivated by the fact that near-term quantum devices can only operate reliably on a limited number of qubits [28]. Specifically, assuming that their quantum capabilities are limited to manipulating s qubits, hybrid classical-quantum protocols that allow to generate samples from a joint distribution $P(i, j)$, correspond to PSD factorizations of P where the PSD factors are block-diagonal, with block-size at most 2^s. Moreover, the minimum amount of classical resources required in a classical-quatum protocol with s qubits, is given by the 2^s-block diagonal PSD-rank of P.

Links to nonnegative matrix factorizations. An r-dimensional *nonnegative matrix factorization* (NMF) of $X \in \mathbb{R}_+^{m\times n}$ [25, 32] is specified by two families of r-dimensional entrywise nonnegative vectors $a_1, \ldots, a_m \in \mathbb{R}_+^r$ and $b_1, \ldots, b_n \in \mathbb{R}_+^r$ satisfying

$$X_{ij} = \langle a_i, b_j \rangle, \ i \in [m], \ j \in [n]. \tag{4}$$

NMF is a widely used dimensionality reduction tool that gives *parts-based representation* of the input data, as it only allows for additive, not subtractive, combinations. To make this point clear, note that an equivalent reformulation of an r-dimensional NMF (4) is $X = AB$ where $A \in \mathbb{R}_+^{m\times r}$ is the matrix whose rows are the a_i's and the matrix $B \in \mathbb{R}_+^{r\times n}$ has as columns the b_j's, or equivalently,

$$X_{:j} \in \mathrm{cone}(A_{:1}, \ldots, A_{:r}), \ j \in [n]. \tag{5}$$

The equivalent viewpoint for NMFs given in (5) is more amenable to interpretation, as it gives a representation of each column of X (i.e., each data point) as nonnegative (and thus additive) combination of the r columns of A, and the columns of B give the coefficients of the conic combination. NMF factorizations have applications in many areas, notable examples including document clustering [39], music analysis [8], speech-source separation [34] and cancer-class identification [10]. For a comprehensive discussion on NMFs the reader is referred to the survey [11] and references therein.

NMF factorizations are a special case of PSD factorizations where the $r \times r$ PSD matrices A_i and B_j are *diagonal*, i.e., we have that $A_i = \mathrm{diag}(a_i)$ and $B_j = \mathrm{diag}(b_j)$ for some vectors $a_i, b_j \in \mathbb{R}_+^r$ (recall that a diagonal matrix is PSD iff its diagonal entries are nonnegative). Moreover, a PSD factorization of X, $X_{ij} = \mathrm{tr}(A_i B_j)$, for which all the PSD factors A_i, B_j commute corresponds to an NMF factorization. In this sense, PSD factorizations are a non-commutative generalization of NMF factorizations.

Interpretability of PSD factorizations. An equivalent way to define an r-dimensional NMF for a data matrix X (cf. (4)) is through the existence of a liner mapping $\mathcal{A} : \mathbb{R}^r \to \mathbb{R}^m$ satisfying

$$X_{:j} \in \mathcal{A}(\mathbb{R}_+^r) \ \text{ for all } j \in [n] \quad \text{ and } \quad \mathcal{A}(\mathbb{R}_+^r) \subseteq \mathbb{R}_+^n. \tag{6}$$

Consequently, the mapping $\mathcal{A}$ (or rather, the image of the extreme rays of the cone $\mathbb{R}_+^r$ under $\mathcal{A}$), describe a latent space that can generate all data points $X_{:j}$ via nonnegative combinations.

Analogously, in the setting of PSD factorizations, the existence of an r-dimensional PSD factorization of X (cf. (1)) is equivalent to the existence of a linear mapping $\mathcal{A} : \mathbb{S}^r \to \mathbb{R}^m$ satisfying

$$X_{:j} \in \mathcal{A}(\mathbb{S}_+^r) \ \text{ for all } j \in [n] \quad \text{ and } \quad \mathcal{A}(\mathbb{S}_+^r) \subseteq \mathbb{R}_+^n. \tag{7}$$

Comparing (6) and (7), the difference between NMF and PSD factorizations is immediately apparent. In the setting of PSD factorizations the latent space is *infinite-dimensional*, and specifically, it is the image of the extreme rays of the cone of $r \times r$ PSD matrices (i.e., all matrices $uu^\top$ where $u \in \mathbb{R}^r$) under $\mathcal{A}$. In this latent space, each data point $X_{:j}$ is represented by a PSD matrix $B_j \in \mathbb{S}^r_+$, and using its spectral decomposition $B_j = \sum_{i=1}^r \lambda_i u_i u_i^\top$, leads to the representation $X_{:j} = \sum_i \lambda_i \, \mathcal{A}(u_i u_i^\top)$. Additional details and explicit examples demonstrating the qualitative difference in expressive power between NMF and PSD factorizations are given in Section 6.

2 Prior Works on PSD Factorizations and Summary of Results

A canonical starting point for finding an (approximate) r-dimensional PSD factorization of a given matrix $X \in \mathbb{R}^{m\times n}_+$ is to solve the non-convex optimization problem

$$\inf \sum_{i,j} (X_{ij} - \mathrm{tr}(A_i B_j))^2 \quad \text{s.t.} \quad A_1, \dots, A_m, B_1, \dots B_n \in \mathbb{S}^r_+, \tag{8}$$

aiming to find an approximate r-dimensional PSD factorization that minimizes the square loss over all entries of X. Fixing one of the two families of matrix variables, say the A_i's, problem (8) is separable with respect to $B_1, \dots, B_m$. Consequently, a reasonable solution approach for (8) is to alternate between updating the A_i's and B_j's by solving the sub-problems:

$$A_i \leftarrow \arg\inf \ \sum_{i,j} (X_{ij} - \mathrm{tr}(A_i B_j))^2 \quad \text{s.t.} \quad A_1, \dots, A_m \in \mathbb{S}^r_+ \tag{9}$$

$$B_j \leftarrow \arg\inf \ \sum_{i,j} (X_{ij} - \mathrm{tr}(A_i B_j))^2 \quad \text{s.t.} \quad B_1, \dots, B_n \in \mathbb{S}^r_+ \tag{10}$$

The two sub-problems in each update step are symmetric in the variables A_i and B_j, with the small modification where we replace X with its transpose. As such, for the remainder of this discussion, we only focus on the sub-problem (10) corresponding to fixing the A_i's and updating the B_j's. Moreover, (10) is separable with respect to each variable B_i, so it suffices to focus on

$$\inf \sum_{i} (X_{ij} - \mathrm{tr}(A_i B_j))^2 \quad \text{s.t.} \quad B_j \in \mathbb{S}^r_+. \tag{11}$$

Lastly, to simplify notation we omit subscripts, and specifically, we denote by x the j-th column of X and by B the PSD matrix variable B_j. Defining $\mathcal{A} : \mathbb{S}^r \to \mathbb{R}^m$ to be the linear map $\mathcal{A}(Z) = (\langle A_1, Z\rangle, \ \dots \ , \langle A_m, Z\rangle)$, problem (11) can be then equivalently written as

$$\inf \ \|x - \mathcal{A}(B)\|_2^2 \quad \text{s.t.} \quad B \in \mathbb{S}^r_+. \tag{12}$$

The optimization problem (12) is convex, and in fact, falls within the well-studied class of convex quadratic SDPs. Nevertheless, there is no closed-form solution for this family of optimization problems, and consequently, typical solution strategies rely on numerical optimization, e.g., see [35].

Summary of results. In this paper we introduce and study an iterative algorithm (Algorithm 1) we call the *Matrix Multiplicative Update* (MMU) algorithm for computing PSD factorizations. The MMU algorithm builds on the Majorization-Minimization framework, and as discussed in the previous section, the main workhorse is an iterative algorithm for the convex quadratic SDP (12).

From a computational perspective, the iterates of the MMU algorithm are updated via conjugation with appropriately defined matrices, so our method has the advantage of being *simple to implement* and moreover, the PSDness of the iterates is *automatically guaranteed*. From a theoretical perspective, the squared loss objective is non-increasing along the algorithms' trajectories (Theorem 1) and moreover, its fixed points satisfy the first-order optimality conditions (Theorem 2). The analysis of the MMU algorithm relies on the use of several operator trace inequalities (including Von Neumann's trace inequality and Lieb's Concavity Theorem).

An important feature of the MMU algorithm is that if it is initialized with block-diagonal PSD matrices, the same block-diagonal structure is *preserved throughout its execution*, which leads to an algorithm for calculating block-diagonal PSD factorizations. In particular, if the MMU algorithm is initialized with *diagonal PSD matrices*, the iterates remain diagonal PSD throughout, and as it turns out, our algorithm in this case reduces to Lee-Seung's seminal Multiplicative Update algorithm for computing NMFs [26]. Moreover, we show how the MMU algorithm can be used as a primitive to calculate PSD factorizations of nonnegative tensors. In terms of numerical experiments, we demonstrate the utility of our method for both synthetic and real data (CBCL image dataset).

Existing work. All existing algorithms for computing PSD factorizations employ the alternating minimization approach described in the previous section, where we fix one set of variables and minimize over the other, and essentially boil down into finding algorithms for the convex problem (12).

Projected Gradient Method (PGM). The first approach for computing PSD factorizations is based on applying PGM to (12), alternating between a gradient step to minimize the objective and a projection step onto the set of PSD matrices [37]. The latter projection step uses the following useful fact: Given the spectral decomposition $C = U\mathrm{diag}(\lambda_i)U^\top$ of a matrix $C \in \mathbb{S}^n$, the projection onto the PSD cone is $U\mathrm{diag}(\max(0, \lambda_i))U^\top$ [15]. The vanilla PGM has slow convergence rate, so the authors in [37] also propose an accelerated variant that incorporates a momentum term.

Coordinate Descent. The authors in [37] also propose a different algorithm combining the ideas of coordinate descent and a change of variables that allows them to also control the rank of the PSD factors, which was popularized by the seminal work of Burer and Monteiro for solving rank-constrained SDPs [3]. Concretely, the authors use the parameterization $A_i = a_i a_i^\top$, and $B_j = b_j b_j^\top$, where $a_i \in \mathbb{R}^{r \times r_{A_i}}$, and $b_j \in \mathbb{R}^{r \times r_{B_j}}$ for some fixed $r_{A_i}, r_{B_j} \in \mathbb{N}$, and optimize using a coordinate descent scheme over the entries of the matrices a_i and b_j. In this setting, problem (12) is a quartic polynomial in the entries of b. Thus, its gradient is a cubic polynomial, and its roots can be found using Cardano's method and careful book-keeping (for a similar approach see also [29]).

Connections to Affine Rank Minimization and Phase Retrieval. A different set of algorithms developed in [19, 20, 21] is based on the connections between computing PSD factorizations with the affine rank minimization (ARM) and the phase retrieval (PR) in signal procesing. First, recall that the PSD-ARM problem focuses on recovering a low-rank matrix from affine measurements:

$$\min \quad \mathrm{rank}(B) \quad \text{s.t.} \quad \mathcal{A}(B) = x,\ B \in \mathbb{S}^r_+.$$

Here, $\mathcal{A}$ is a known linear map representing measurements while x is known vector of observations. Due to the non-convexity of the rank function, a useful heuristic initially popularized in the control community is to replace the rank by the trace function, e.g., see [30] and [31], in which case the resulting problem is an instance of an SDP. A different heuristic for PSD-ARM is to find a PSD matrix of rank at most k that minimizes the squared loss function, i.e.,

$$\inf \quad \|x - \mathcal{A}(B)\|_2^2 \quad \text{s.t.} \quad B \in \mathbb{S}^r_+,\ \mathrm{rank}(B) \leq k, \tag{13}$$

where alternatively, the rank constraint can be enforced by parametrizing the PSD matrix variable $B \in \mathbb{S}^r_+$ as $B = bb^\top$ with $b \in \mathbb{R}^{r \times k}$. The point of departure for the works [19, 20, 21] is that problem (13) corresponds exactly to the sub-problem (12) encountered in any alternate minimization strategy for computing PSD factorizations, albeit with an additional rank constraint. In view of this, any algorithm from the signal processing literature developed for ARM can be applied to (12).

The main algorithms considered in [19, 20, 21] are Singular Value Projection (SVP) [16], Procrustes Flow [36], and variants thereof. In terms of convergence guarantees, for affine maps $\mathcal{A}$ obeying the Restricted Isometry Property [4], both algorithms converge to an optimal solution. Nevertheless, it is unclear whether these guarantees carry over when applied to the PSD factorization problem.

Roadmap. In Section 3 we derive our MMU algorithm for computing PSD factorizations and in Section 4 we show that its fixed points correspond to KKT points. In Section 5 we give various theoretical applications of the MMU algorithm and in Section 6 we go from theory to practise and apply the MMU algorithm to synthetic and real datasets.

3 A Matrix Multiplicative Update Algorithm for PSD Factorizations

In this section we describe our algorithm for computing (approximate) PSD factorizations of a matrix X. As we discussed, our method is an alternating minimization approach in which we alternate between optimizing over the variables $\{A_i\}$ and $\{B_j\}$. The sub-problem in each update step is symmetric in the variables $\{A_i\}$ and $\{B_j\}$, with the small modification whereby we replace X with its transpose. As such, in the remainder of this discussion, we assume that the variables $\{A_i\}$ are fixed and we perform the update on the variables $\{B_j\}$. The resulting sub-problem is given by (12).

Majorization-Minimization (MM) Framework. Our algorithm is an instance of the (MM) framework, e.g. see [22] and references therein. To briefly describe this approach, suppose we need to solve the optimization problem $\min\{F(x) : x \in \mathcal{X}\}$. The MM framework relies on the existence of a parametrized family of auxilliary functions $u_x : \mathcal{X} \to \mathbb{R}$, one for each $x \in \mathcal{X}$, where:

$$F(y) \leq u_x(y), \text{ for all } y \in \mathcal{X} \text{ and } F(x) = u_x(x). \tag{14}$$

Based on these two properties, F is nonincreasing under the update rule:

$$x^{\text{new}} = \operatorname{argmin}\{u_{x^{\text{old}}}(y) : y \in \mathcal{X}\}, \tag{15}$$

as can be easily seen by: $F(x^{\text{new}}) \leq u_{x^{\text{old}}}(x^{\text{new}}) \leq u_{x^{\text{old}}}(x^{\text{old}}) = F(x^{\text{old}})$.

We conclude with two important remarks concerning the MM framework. First, note that although the iterates generated by the MM update rule (23) are nonincreasing in objective function value, there is in general no guarantee that they converge to a minimizer. Secondly, for the MM approach to be of any use, the auxilliary functions employed at each iteration need to be easy to optimize.

Matrix Geometric Mean. Our choice of auxilliary functions relies on the well-studied notion of a geometric mean between a pair of positive definite matrices, whose definition we recall next. For additional details and omitted proofs the reader is referred to [2, 24]. The matrix geometric mean of two positive definite matrices C and D is given by

$$C\#D = C^{1/2}(C^{-1/2}DC^{-1/2})^{1/2}C^{1/2}, \tag{16}$$

or equivalently, it is the unique positive definite solution of the Riccati equation

$$XC^{-1}X = D, \tag{17}$$

in the matrix variable X. The matrix geometric mean also has a nice geometric interpretation in terms of the Riemannian geometry of the manifold of positive definite matrices, and specifically, $C\#D$ is the midpoint of the unique geodesic joining C and D. Finally, the matrix geometric mean is symmetric in its two arguments $C\#D = D\#C$ and also satisfies $(C\#D)^{-1} = C^{-1}\#D^{-1}$.

The MMU Algorithm for PSD Factorizations. The main step for deriving our algorithm for approximately computing PSD factorizations is to apply the MM framework, with a meticulously chosen auxilliary function, to the convex quadratic SDP (12). Our main result is the following:

Theorem 1. *Consider a fixed vector $x \in \mathbb{R}^m_+$ and let $\mathcal{A} : \mathbb{S}^r \to \mathbb{R}^m$ be the linear map defined by $Z \mapsto \mathcal{A}(Z) = (\operatorname{tr}(A_1Z), \ldots, \operatorname{tr}(A_mZ))$, for some fixed $r \times r$ positive definite matrices $A_1, \ldots, A_m$. Then, the objective function $\|x - \mathcal{A}(B)\|_2^2$ is non-increasing under the update rule*

$$B_{\text{new}} = W(\mathcal{A}^\top x)W, \quad \textit{where} \quad W = ([\mathcal{A}^\top\mathcal{A}](B_{\text{old}}))^{-1}\#(B_{\text{old}}),$$

and moreover, if initialized with a positive definite matrix, the iterates remain positive definite.

Proof. First, note that if the A_i's and B_{old} are all positive definite, the update rule is well-defined. Indeed, we have $[\mathcal{A}^\top\mathcal{A}](B_{\text{old}}) = \sum_{k=1}^m \operatorname{tr}(A_kB_{\text{old}})A_k$ is also positive definite, and thus invertible.

Set $F(B) := \|x - \mathcal{A}(B)\|_2^2$ and define the function

$$u_{B_{\text{old}}}(B) := F(B_{\text{old}}) + \langle \nabla F(B_{\text{old}}), B - B_{\text{old}}\rangle + \langle B - B_{\text{old}}, T(B - B_{\text{old}})\rangle, \tag{18}$$

where $T : \mathbb{S}^r \to \mathbb{S}^r$ is the operator given by

$$T(Z) = W^{-1}ZW^{-1} \quad \text{and} \quad W = ([\mathcal{A}^\top\mathcal{A}](B_{\text{old}}))^{-1}\#(B_{\text{old}}).$$

The claim of the theorem will follow as an immediate consequence of the MM framework, as long as we establish that $u_{B_{\text{old}}}(B)$ is an auxilliary function, i.e., it satisfies the two properties given in (14).

Clearly, we have that $u_{B_{\text{old}}}(B_{\text{old}}) = F(B_{\text{old}})$, so it only remains to show the domination property, that is, $u_{B_{\text{old}}}(B) \leq F(B)$, for all $B \in \mathbb{S}^r_+$. In fact, we show a slightly stronger result, namely that

$u_{B_{\text{old}}}(B) \leq F(B)$ holds for all symmetric matrices $B \in \mathbb{S}^r$. To see this we use the second order Taylor expansion of F at B_{old}, which as F is quadratic in B, is given by

$$F(B) = F(B_{\text{old}}) + \langle \nabla F(B_{\text{old}}), B - B_{\text{old}} \rangle + \|\mathcal{A}(B - B_{\text{old}})\|_2^2. \tag{19}$$

Comparing the expressions (18) and (19), to show that $F(B) \leq u_{B_{\text{old}}}(B)$ for all $B \in \mathbb{S}^r$ it suffices to check that the operator $T - \mathcal{A}^\top \mathcal{A}$ is positive; i.e., $\langle Z, [T - \mathcal{A}^\top \mathcal{A}](Z) \rangle \geq 0$ for any matrix $Z \in \mathbb{S}^r$. This claim is the main technical part of the proof, deferred to Lemma 3 in the Appendix.

Furthermore, the fact that $T - \mathcal{A}^\top \mathcal{A}$ is a positive operator, also implies that T is itself a positive operator. Consequently, the MM update (23) obtained by using the auxilliary function (18) can be calculated just by setting the gradient equal to zero, and is given by

$$B_{\text{new}} = B_{\text{old}} - T^{-1}\left([\mathcal{A}^\top \mathcal{A}](B_{\text{old}}) - \mathcal{A}^\top(x)\right). \tag{20}$$

Moreover, as $T^{-1}(Z) = WZW$ and $W = ([\mathcal{A}^\top \mathcal{A}](B_{\text{old}}))^{-1} \# (B_{\text{old}})$ it follows that

$$B_{\text{old}} = W([\mathcal{A}^\top \mathcal{A}](B_{\text{old}}))W = T^{-1}([\mathcal{A}^\top \mathcal{A}](B_{\text{old}})), \tag{21}$$

where for the first equality we use the unicity property of the matrix geometric mean (recall (17)). Subsequently, using (21), the MM update rule in (20) simplifies to the following:

$$B_{\text{new}} = T^{-1}(\mathcal{A}^\top \mathbf{x}) = W(\mathcal{A}^\top \mathbf{x})W. \tag{22}$$

Lastly, since the A_i's are PSD, it follows that $\mathcal{A}^\top \mathbf{x} = \sum_i x_i A_i$ is a conic combination of PSD matrices (recall that $x \in \mathbb{R}^m_+$), and thus, it is itself PSD. Consequently, B_{new} is PSD. In fact, if the matrices A_i and B_{old} are positive definite, the updated matrix B_{new} is also positive definite. □

Having established an iterative method for problem (12) that is non-increasing in value and retains PSDness, we can incorporate this as a sub-routine in our alternating optimization scheme for computing PSD factorizations. The pseudocode of the resulting method is given in Algorithm 1.

Algorithm 1 Matrix Multiplicative Update algorithm for computing PSD factorizations

```
Input: A matrix $X \in \mathbb{R}^{m \times n}_{\geq 0}$, parameter $r \in \mathbb{N}$
Output: $\{A_1, \ldots, A_m\}, \{B_1, \ldots, B_n\} \subseteq \mathbb{S}^r_+$, $X_{ij} \approx \mathrm{tr}(A_i B_j)$ for all $i, j$
  while stopping criterion not satisfied:
        $A_i \leftarrow V_i(\mathcal{B}^\top x_i)V_i$   where   $V_i = ([\mathcal{B}^\top \mathcal{B}](A_i))^{-1} \# A_i$,   $x_i = X_{i:}$
        $B_j \leftarrow W_j(\mathcal{A}^\top x_j)W_j$   where   $W_j = ([\mathcal{A}^\top \mathcal{A}](B_j))^{-1} \# (B_j)$,  $x_j = X_{:j}$        (23)
```

4 Fixed Points of the MMU Algorithm

In this section, we show that the fixed points of the MMU algorithm satisfy the Karush-Kuhn-Tucker (KKT) optimality conditions for problem (8). Letting $\{A_i^*\}_{i \in [m]}, \{M_i^*\}_{i \in [m]}$ and $\{B_j^*\}_{j \in [n]}, \{\Lambda_j^*\}_{j \in [n]}$ be pairs of primal-dual optimal solutions of (8) with zero duality gap, it is straightforward to verify that the KKT conditions are

$$\begin{aligned}
\mathrm{tr}(A_i^* M_i^*) = \mathrm{tr}\big(B_j^* \Lambda_j^*\big) = 0, &\quad i \in [m], j \in [n] \\
\mathcal{B}^\top(X_{:i}) - [\mathcal{B}^\top \mathcal{B}](A_i^*) = M_i^*, &\quad i \in [m] \\
\mathcal{A}^\top(X_{:j}) - [\mathcal{A}^\top \mathcal{A}](B_j^*) = \Lambda_j^*, &\quad j \in [n].
\end{aligned}$$

Furthermore, assuming that the primal optimal solutions $\{A_i^*\}_{i \in [m]}$ and $\{B_j^*\}_{j \in [n]}$ are all *positive definite*, it follows immediately from the complementary slackness conditions that $M_i^* = \Lambda_j^* = 0$ for all $i \in [m], j \in [n]$. Consequently, in the special case of positive definite optimal solutions $\{A_i^*\}_{i \in [m]}$ and $\{B_j^*\}_{j \in [n]}$, the KKT conditions reduce to

$$\mathcal{B}^\top(X_{:i}) = [\mathcal{B}^\top \mathcal{B}](A_i^*),\ i \in [m] \quad \text{and} \quad \mathcal{A}^\top(X_{:j}) = [\mathcal{A}^\top \mathcal{A}](B_j^*),\ j \in [n]. \tag{24}$$

Based on the preceding discussion, in the next result (whose proof follows by Lemma 4 in the Appendix) shows that we can interpret our MMU algorithm as a fixed-point method for satisfying the KKT optimality conditions corresponding to problem (8).

Theorem 2. *If $\{A_i\}_{i \in [m]}$ and $\{B_j\}_{j \in [n]}$ are positive definite fixed points of the update rule of the MWU algorithm given in* (23)*, then they also satisfy the KKT conditions* (24).

5 Applications of the MMU algorithm

Block-diagonal (BD) PSD factorizations. If the MMU algorithm is initialized with BD positive definite matrices with *the same* block structure, the BD structure is preserved at each update. Indeed, as $[\mathcal{A}^\top\mathcal{A}](B_{\rm old}) = \sum_{k=1}^m \operatorname{tr}(A_k B_{\rm old})A_k$ we see that $[\mathcal{A}^\top\mathcal{A}](B_{\rm old})^{-1}$, and thus, $([\mathcal{A}^\top\mathcal{A}](B_j))^{-1}\#(B_j)$ share the same block structure. Lastly, by definition of the MMU algorithm (23), $B_{\rm new}$ is also block-diagonal with the same structure. Thus, if initialized with BD-PSD matrices, the MMU algorithm gives a method for computing a BD-PSD factorization.

Recovering Lee-Seung's algorithm for NMF. Diagonal matrices can be considered as block-diagonal in a trivial manner. Nevertheless, by the preceding discussion, if initialized with diagonal PSD matrices, the iterates of the MMU algorithm remain diagonal PSD throughout. In this special case, our MMU algorithm reduces to Lee-Seung's (LS) seminal Multiplicative Update algorithm for computing NMFs [26]. LS's algorithm is perhaps the most widely used method for computing NMFs as it has succeeded to identify meaningful features in a diverse collection of real-life data sets and is extremely simple to implement. Specifically, LS's updates are

$$A \leftarrow A \circ \frac{XB^\top}{ABB^\top} \text{ and } B \leftarrow B \circ \frac{A^\top X}{A^\top AB}, \tag{25}$$

where $X \circ Y, X/Y$ denote the componentwise multiplication, division of two matrices respectively. Setting $A_i = \operatorname{diag}(a_i)$ and $B_j = \operatorname{diag}(b_j)$, the MMU algorithm updates B_j as $B_j \leftarrow B_j\Big(\sum_{i=1}^m \langle a_i, b_j\rangle A_i\Big)^{-1}\Big(\sum_{i=1}^m X_{ij}A_i\Big)$, which is also a diagonal PSD matrix. Setting $A^\top = \begin{pmatrix} a_1^\top & \dots & a_m^\top\end{pmatrix}$ and $B = \begin{pmatrix} b_1 & \dots & b_n\end{pmatrix}$, this coincides with LS's update rule (25).

PSD factorizations for nonnegative tensors. Motivated by PSD factorizations of nonnegative matrices, [18] define an r-dimensional PSD factorization of a nonnegative tensor T (with n indices of dimension d) as a collection of PSD matrices $C^{(1)}_{i_1}, \dots, C^{(n)}_{i_n} \in \mathbb{S}^r_+$ for all $i_k \in [d]$ such that $T_{i_1 \dots i_n} = \operatorname{sum}(C^{(1)}_{i_1} \circ \dots \circ C^{(n)}_{i_n})$, for all $i_k \in [d]$, where $\circ$ denotes the Schur product of matrices and $\operatorname{sum}(X) = \sum_{ij} X_{ij}$. The motivation for studying tensor PSD factorizations comes from the fact that they characterize the quantum correlation complexity for generating multipartite classical distributions [18]. We now show how our MMU algorithm can be used as a primitive to calculate tensor PSD factorizations. For simplicity of presentation we restrict to $n = 3$ and consider

$$\inf \sum_{i_1,i_2,i_3} \Big(T_{i_1i_2i_3} - \operatorname{sum}(C^{(1)}_{i_1} \circ C^{(2)}_{i_2} \circ C^{(3)}_{i_3})\Big)^2 \quad \text{subject to} \quad C^{(1)}_{i_1}, C^{(2)}_{i_2}, C^{(3)}_{i_3} \in \mathbb{S}^r_+. \tag{26}$$

As in the case of PSD factorizations we employ a block coordinate descent approach. Specifically, fixing all matrices $C^{(1)}_{i_1}, C^{(2)}_{i_2}$ the optimization problem (26) is separable wrt each $C^{(3)}_{i_3}$ for all $i_3 \in [d]$. Thus, defining the map $\mathcal{A} : \mathbb{S}^r \to \mathbb{R}^{d^2},\ X \mapsto (\langle X, C^{(1)}_{i_1} \circ C^{(2)}_{i_2}\rangle)_{i_1,i_2}$, we need to solve

$$\arg\inf \|\operatorname{vec}(T_{::i_3}) - \mathcal{A}(C)\|_2^2 \quad \text{subject to} \quad C \in \mathbb{S}^r_+, \tag{27}$$

for all $i_3 \in [d]$. Note that $\mathcal{A}^\top : \mathbb{R}^{d^2} \to \mathbb{S}^r$ where $x = (x_{x_1x_2}) \mapsto \sum_{i_1,i_2 \in [d]} x_{i_1i_2} C^{(1)}_{i_1} \circ C^{(2)}_{i_2}$ is a PSD matrix, as the Schur product of PSD matrices is PSD. Thus, Theorem 1 gives an update rule that preserves PSDness and for which the objective function (27) is nonincreasing.

6 Numerical experiments

Damping. The implementation of the MMU algorithm requires us to compute inverses and square-roots of certain positive definite matrices. These operations become ill-conditioned whenever the input matrices contain eigenvalues that are close to zero. To mitigate such issues, we apply a damping operation. First, when computing $([\mathcal{B}^\top\mathcal{B}](A_i))^{-1}$ in (23), we instead compute $([\mathcal{B}^\top\mathcal{B}](A_i) + \epsilon I)^{-1}$. Second, when computing the matrix square root $X^{1/2}$ in the process of computing the matrix geometric mean (23) using the expression (16), we instead compute $(X + \epsilon I)^{1/2}$. In our implementations, we apply a choice of $\epsilon = 10^{-8}$.

Distance matrices. Let $v \in \mathbb{R}^n$ be a vector and let M be a $n \times n$ matrix whose entries are $M_{ij} = (v_i - v_j)^2$. M is known as a *distance matrix* and it admits the following 2-dimensional PSD factorization

$$M_{ij} = \mathrm{tr}(A_i B_j), \qquad \text{where} \qquad A_i = \begin{pmatrix} 1 \\ v_i \end{pmatrix} \begin{pmatrix} 1 \\ v_i \end{pmatrix}^\top, B_j = \begin{pmatrix} -v_j \\ 1 \end{pmatrix} \begin{pmatrix} -v_j \\ 1 \end{pmatrix}^\top.$$

We generate a random $v \in \mathbb{R}^n$ with $n = 20$ where each entry is drawn from the standard normal distribution. We apply our algorithm to compute a 2-dimensional factorization whereby we perform 500 iterations over 50 random initializations. We compute the normalized squared error loss of the factorization from the data matrix, and we plot the error over each iteration $\mathrm{Err} = \sum_{i,j} (\mathrm{tr}(A_i B_j) - M_{ij})^2 / \sum_{i,j} M_{ij}^2$ in Figure 1. Our experiments suggest that, with sufficient random initializations, our algorithm finds a PSD factorization that is close to being exact.

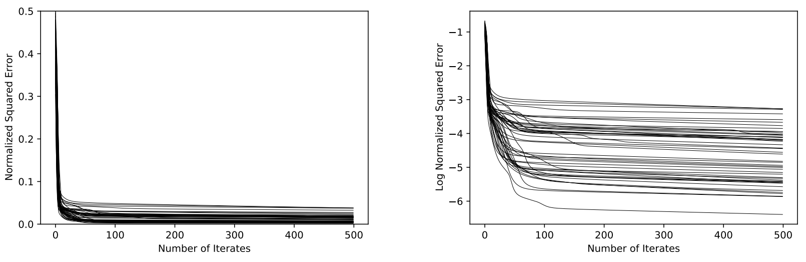

Figure 1: Performance of the MWU algorithm for computing a PSD factorization of a distance matrix. Different curves correspond to different random initializations.

CBCL Face Image Dataset. In our second experiment, we apply the MMU method to compute a PSD factorization of a matrix comprising face images from the CBCL Face Database [1]. The objective of this experiment is to illustrate how computing a PSD factorization can be viewed as a representation learning algorithm that generalizes NMF. The CBCL dataset comprises 2429 images of faces, each of size 19×19 pixels. We process the images so that the pixel intensity has mean 0.5 and standard deviation 0.25, with values subsequently clipped at $[0, 1]$. The resulting data matrix has size 361×2429.

We note that an r-dimensional PSD factorization of X specified by $\{A_i\}, \{B_j\}$ gives rise to a decomposition $X_{:j} = \sum_{i=1}^r \lambda_i \, \mathcal{A}(u_i u_i^\top)$, where $B_j = \sum_{i=1}^r \lambda_i u_i u_i^\top$ is the spectral decomposition. Subsequently, one can view the collection $\{\mathcal{A}(u_i u_i^\top) : \|u_i\| = 1\}$ as basic constituents from which all face images are expressible via non-negative linear combinations. This is analogous to NMF in which we express every data point as non-negative linear combinations from a collection of non-negative basis vectors – these are precisely the linear image of a matrix factor B on standard basis vectors. In this experiment, given a specific face image $X_{:j}$, we show its decomposition as sums of constituents identified by the MMU algorithm – these are the images $\mathcal{A}(u_i u_i^\top)$, where u_i are the eigenvectors of B_j.

As our baseline, we compute a 27-dimensional NMF of the CBCL data matrix over 500 iterations. In Figure 2 we illustrate the decomposition of one of images from the dataset into these 27 constituents. Next, we calculate a 7-dimensional PSD factorization of the CBCL data matrix over 500 iterations and illustrate the decomposition of the same image in the new basis in Figure 3. We compute a 7-dimensional PSD factorization because a symmetric matrix of dimension 7×7 has 28 degrees of freedom, which is comparable with our NMF example. We note that the constituents learned from the PSD factorization appear to capture global features, a phenomenon that has been been also observed for NMF applied to datasets beyond CBCL [27].

Last, we apply the MMU algorithm to compute a block-diagonal PSD factorization with 9 blocks of size 2×2 over 500 iterations (the number of degrees of freedom is 27), and we illustrate the decomposition in Figure 4. In this instance, constituents contain more localized features. An advantage of learning a continuum of basic building blocks is that one can express certain geometries in the data that is otherwise not possible using a finite number of building blocks. As an illustration of this intuition, in Figure 5, we show a continuum of atoms corresponding to a single 2×2 block which captures a transition between the nose and the nostrils.

Computational specifications. Our experiments were conducted in Python on an Intel 7-th Gen i7 processor at 2.8GHz.

Code repository. Our codes are available on the following online repository

`https://github.com/yssoh/PSD_MM`

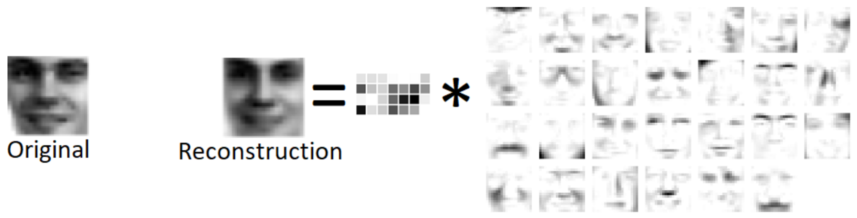

Figure 2: Image decomposition into building blocks learned from 27-dimensional NMF.

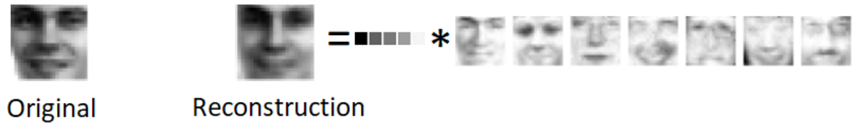

Figure 3: Image decomposition into building blocks learned from 7-dimensional PSD factorization.

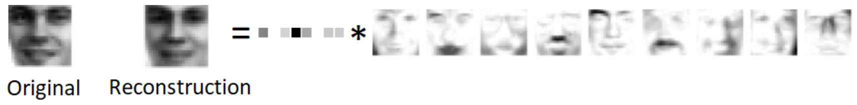

Figure 4: Image decomposition into building blocks learned using 2×2-block PSD factorization.

Figure 5: Visualization of continuum of building blocks learned using 2×2-block PSD factorization.

Funding transparency statement. Yong Sheng Soh gratefully acknowledges Ministry of Education (Singapore) Academic Research Fund (Tier 1) R-146-000-329-133. Antonios Varvitsiotis gratefully acknowledges Ministry of Education (Singapore) Start-Up Research Grant SRG ESD 2020 154 and NRF2019-NRF-ANR095 ALIAS grant. Antonios Varvitsiotis acknowledges initial discussions with Cédric Févotte, Sandra Tan, and Vincent Tan.

References

[1] MIT-CBCL Face Database, Center for Biological and Computational Learning, Massachussetts Institute of Technology. `http://cbcl.mit.edu/software-datasets/FaceData2.html`.

[2] Rajendra Bhatia. *Positive Definite Matrices*. Princeton University Press, 2007.

[3] Samuel Burer and Renato D. C. Monteiro. A Nonlinear Programming Algorithm for Solving Semidefinite Programs via Low-rank Factorization. *Mathematical Programming*, 95(2):329–357, 2003.

[4] Emmanuel J. Candes and Terence Tao. Decoding by Linear Programming. *IEEE Transactions on Information Theory*, 51:4203–4215, 2004.

[5] Murat A. Erdogdu, Yash Deshpande, and Andrea Montanari. Inference in graphical models via semidefinite programming hierarchies. In *Advances in Neural Information Processing Systems*, 2017.

[6] Hamza Fawzi, João Gouveia, Pablo A. Parrilo, Richard Z. Robinson, and Rekha R. Thomas. Positive Semidefinite Rank. *Mathematical Programming*, 153(1):133–177, 2015.

[7] Hamza Fawzi and Pablo A. Parrilo. Exponential lower bounds on fixed-size psd rank and semidefinite extension complexity. https://arxiv.org/pdf/1311.2571.pdf, 2013.

[8] Cédric Févotte, Nancy Bertin, and Jean-Louis Durrieu. Nonnegative Matrix Factorization with the Itakura-Saito Divergence: With Application to Music Analysis. *Neural Computation*, 21(3):793–830, 2009.

[9] Samuel Fiorini, Serge Massar, Sebastian Pokutta, Hans Raj Tiwary, and Ronald de Wolf. Exponential Lower Bounds for Polytopes in Combinatorial Optimization. *Journal of the ACM*, (17), 2015.

[10] Yuan Gao and George Church. Improving Molecular Cancer Class Discovery Through Sparse Non-negative Matrix Factorization. *Bioinformatics*, 21(21):3970–3975, 2005.

[11] Nicolas Gillis. *The Why and How of Nonnegative Matrix Factorization*, chapter Regularization, Optimization, Kernels, and Support Vector Machines, pages 257–291. Number Chapman & Hall/CRC in Machine Learning and Pattern Recognition Series. 2014.

[12] Nicolas Gillis and François Glineur. Accelerated Multiplicative Updates and Hierarchical ALS Algorithms for Nonnegative Matrix Factorization. *Neural Computation*, 24(4):1085–1106, 2012.

[13] Edward F. Gonzalez and Yin Zhang. Accelerating the Lee-Seung Algorithm for Nonnegative Matrix Factorization. Technical report, Rice University, 2005.

[14] João Gouveia, Pablo A. Parrilo, and Rekha Thomas. Lifts of Convex Sets and Cone Factorizations. *Mathematics of Operations Research*, 38(2):248–264, 2013.

[15] Nicholas J. Higham. Computing a Nearest Symmetric Positive Semidefinite Matrix. *Linear Algebra and its Applications*, 103:03–118, 1988.

[16] Prateek Jain, Raghu Meka, and Inderjit Dhillon. Guaranteed Rank Minimization via Singular Value Projection. In *Advances in Neural Information Processing Systems*, 2010.

[17] Rahul Jain, Yaoyun Shi, Zhaohui Wei, and Shengyu Zhang. Efficient Protocols for Generating Bipartite Classical Distributions and Quantum States. *IEEE Transctions on Information Theory*, 59:5171–5178, 2013.

[18] Rahul Jain, Zhaohui Wei, Penghui Yao, and Shengyu Zhang. Multipartite quantum correlation and communication complexities. *computational complexity volume*, 26:199–228, 2017.

[19] Dana Lahat and Cédric Févotte. Positive Semidefinite Matrix Factorization: A Link to Phase Retrieval and a Block Gradient Algorithm. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[20] Dana Lahat and Cédric Févotte. Positive Semidefinite Matrix Factorization Based on Truncated Wirtinger Flow. In *28th European Signal Processing Conference (EUSIPCO)*, 2020.

[21] Dana Lahat, Yanbin Lang, Vincent Y. F. Tan, and Cédric Févotte. Positive Semidefinite Matrix Factorization: A Connection with Phase Retrieval and Affine Rank Minimization. *IEEE Transactions on Signal Processing*, in press, 2021.

[22] Kenneth Lange. *MM Optimization Algorithms*. SIAM, 2016.

[23] Javad Lavaei and Steven H. Low. Zero Duality Gap in Optimal Power Flow Problem. *IEEE Transactions on Power Systems*, 27(1), 2012.

[24] Jimmie D. Lawson and Yongdo Lim. The Geometric Mean, Metrics, and More. In *The American Mathematical Monthly*, volume 108, pages 797–812, 2001.

[25] Daniel D. Lee and H. Sebastian Seung. Learning the Parts of Objects by Non-negative Matrix Factorization. *Nature*, 401, 1999.

[26] Daniel D. Lee and H. Sebastian Seung. Algorithms for Non-negative Matrix Factorization. In *Advances in Neural Information Processing Systems 13*, 2000.

[27] Stan Z. Li, Xin Wen Hou, Hong Jiang Zhang, and Qian Sheng Cheng. Learning Spatially Localized, Parts-based Representation. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*, 2001.

[28] Xiaodie Lin, Zhaohui Wei, and Penghui Yao. Quantum and Classical Hybrid Generations for Classical Correlations. https://arxiv.org/abs/2007.10673, 2020.

[29] Jakub Marecek and Martin Takac. A low-rank coordinate-descent algorithm for semidefinite programming relaxations of optimal power flow. *Optimisation Methods and Software*, 32(4):849–871, 2017.

[30] Mehran Mesbahi and George P. Papavassilopoulos. On the Rank Minimization Problem Over a Positive Semidefinite Linear Matrix Inequality. *IEEE Transactions on Automatic Control*, 42(2), 1997.

[31] Karthik Mohan and Maryam Fazel. New Restricted Isometry Results for Noisy Low-rank Recovery. In *IEEE International Symposium on Information Theory (ISIT)*, 2010.

[32] Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error. *Environmentrics*, 5:111–126, 1994.

[33] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems 31*, 2018.

[34] Mikkel N. Schmidt and Rasmus K. Olsson. Single-channel Speech Separation Using Sparse Non-negative Matrix Factorization. In *Ninth International Conference on Spoken Language Processing*, 2006.

[35] Kim-Chuan Toh. An inexact primal–dual path following algorithm for convex quadratic sdp. *Mathematical Programming*, 112:221–254, 2008.

[36] Stephen Tu, Ross Boczar, Max Simchowitz, Mahdi Soltanolkotabi, and Benjamin Recht. Low-rank Solutions of Linear Matrix Equations via Procrustes Flow. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 964–973, 2016.

[37] Arnaud Vandaele, François Glineur, and Nicolas Gillis. Algorithms for Positive Semidefinite Factorization. *Computational Optimization and Applications*, 71(1):193–219, 2018.

[38] Lieven Vandenberghe and Stephen P. Boyd. Semidefinite Programming. *SIAM Review*, 48(1):49–95, 1996.

[39] Wei Xu, Xin Liu, and Yihong Gong. Document Clustering Based on Non-negative Matrix Factorization. In *the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 267–273, 2003.

[40] Mihalis Yannakakis. Expressing Combinatorial Optimization Problems by Linear Programs. *Journal of Computer and System Sciences*, 43:441–466., 1991.

7 Proof of the domination property

In this section we prove the domination property, which is the last remaining ingredient in the proof of Theorem 3.

Lemma 3. *Fix $r \times r$ positive definite matrices $A_1, \dots, A_m$ and consider the linear mapping*

$$\mathcal{A} : \mathbb{S}^r \to \mathbb{R}^m \text{ where } Z \mapsto (\langle A_1, Z\rangle,\ \dots\ , \langle A_m, Z\rangle)\,.$$

Moreover, fix an $r \times r$ positive definite matrix B and set

$$W = [\mathcal{A}^\top \mathcal{A}](B)\#B^{-1}.$$

Then, we have that

$$\langle Z, WZW\rangle \geq \langle Z, [\mathcal{A}^\top \mathcal{A}](Z)\rangle, \text{ for all } Z \in \mathbb{S}^r. \tag{28}$$

Proof. First, we remark that $[\mathcal{A}^\top \mathcal{A}](B) = \sum_{k=1}^m \langle A_k, B\rangle A_k$ is the sum of positive definite matrices and hence positive definite. Subsequently, W, which is defined as the geometric mean of two positive definite matrices, exists.

Second, we reduce the theorem to case where $B = I$. For this, define the linear map

$$\tilde{\mathcal{A}} : \mathbb{S}^r \to \mathbb{R}^m, \quad Z \mapsto \mathcal{A}(B^{1/2} Z B^{1/2}).$$

Noting that

$$\tilde{\mathcal{A}}(I) = \mathcal{A}(B) \text{ and } \tilde{\mathcal{A}}^\top(Z) = B^{1/2}\mathcal{A}^\top(Z)B^{1/2},$$

it follows that

$$\begin{aligned} W &= [\mathcal{A}^\top \mathcal{A}](B)\#B^{-1} = B^{-1}\#[\mathcal{A}^\top \mathcal{A}](B) \\ &= B^{-1/2}(B^{1/2}[\mathcal{A}^\top \mathcal{A}](B)B^{1/2})^{1/2}B^{-1/2} \\ &= B^{-1/2}([\tilde{\mathcal{A}}^\top \tilde{\mathcal{A}}](I))^{1/2}B^{-1/2}. \end{aligned}$$

Furthermore, setting

$$\tilde{Z} = B^{-1/2} Z B^{-1/2},$$

we get that

$$\begin{aligned} \langle Z, WZW\rangle &= \operatorname{tr}(ZWZW) \\ &= \operatorname{tr}\Big(ZB^{-1/2}([\tilde{\mathcal{A}}^\top \tilde{\mathcal{A}}](I))^{1/2}B^{-1/2}ZB^{-1/2}([\tilde{\mathcal{A}}^\top \tilde{\mathcal{A}}](I))^{1/2}B^{-1/2}\Big) \\ &= \operatorname{tr}\Big(B^{-1/2}ZB^{-1/2}([\tilde{\mathcal{A}}^\top \tilde{\mathcal{A}}](I))^{1/2}B^{-1/2}ZB^{-1/2}([\tilde{\mathcal{A}}^\top \tilde{\mathcal{A}}](I))^{1/2}\Big) \\ &= \langle \tilde{Z}, ([\tilde{\mathcal{A}}^\top \tilde{\mathcal{A}}](I))^{1/2}\tilde{Z}([\tilde{\mathcal{A}}^\top \tilde{\mathcal{A}}](I))^{1/2}\rangle. \end{aligned}$$

Similarly

$$\begin{aligned} \langle Z, [\mathcal{A}^\top \mathcal{A}](Z)\rangle &= \langle B^{1/2}B^{-1/2}ZB^{-1/2}B^{1/2}, [\mathcal{A}^\top \mathcal{A}](B^{1/2}B^{-1/2}ZB^{-1/2}B^{1/2})\rangle \\ &= \langle \tilde{Z}, B^{1/2}[\mathcal{A}^\top \mathcal{A}](B^{1/2}\tilde{Z}B^{1/2})B^{1/2}\rangle \\ &= \langle \tilde{Z}, [\tilde{\mathcal{A}}^\top \tilde{\mathcal{A}}](\tilde{Z})\rangle. \end{aligned}$$

To conclude the proof it remains to prove the theorem statement in the special case where $B = I$. Note that when $B = I$ we have

$$W = [\mathcal{A}^\top \mathcal{A}](I)\#I^{-1} = [\mathcal{A}^\top \mathcal{A}](I)\#I = I\#[\mathcal{A}^\top \mathcal{A}](I) = ([\mathcal{A}^\top \mathcal{A}](I))^{1/2},$$

where in the second to last equality we used that the geometric mean is symmetric in its two arguments and the least equality follows by the definition of the geometric mean (21).

Furthermore, by the definition of the map $\mathcal{A}$ we have that

$$[\mathcal{A}^\top \mathcal{A}](Z) = \sum_{k=1}^m \langle A_k, Z\rangle A_k,$$

which in particular implies that

$$W = ([\mathcal{A}^\top \mathcal{A}](I))^{1/2} = \left(\sum_k \operatorname{tr}(A_k) A_k\right)^{1/2}.$$

Thus, in the case where $B = I$, the inequality we need to prove (cf. Equation (1)) specializes to:

$$\operatorname{tr}\left(Z \left(\sum_k \operatorname{tr}(A_k) A_k\right)^{1/2} Z \left(\sum_k \operatorname{tr}(A_k) A_k\right)^{1/2}\right) \geq \sum_k \operatorname{tr}(A_k Z)^2, \text{ for all } Z \in \mathbb{S}^r. \tag{29}$$

Given a concave function $f(x)$ that is also positively homogenous with degree one in x (that is, $f(\lambda x) = \lambda f(x)$ for all $\lambda \geq 0$), one has

$$f(x_1 + \ldots + x_k) \geq f(x_1) + \ldots + f(x_k).$$

A special case of the Lieb's concavity theorem tells us that the function $f(X) = \operatorname{tr}(Z X^{1/2} Z X^{1/2})$ is concave for all symmetric matrices Z. It is clear that $f(X)$ is positively homogenous with degree one. Hence

$$\operatorname{tr}\left(Z (\sum_k \operatorname{tr}(A_k) A_k)^{1/2} Z (\sum_k (\operatorname{tr}(A_k) A_k)^{1/2}\right) \geq \sum_k \operatorname{tr}(A_k) \operatorname{tr}\Big(Z A_k^{1/2} Z A_k^{1/2}\Big), \tag{30}$$

for any family of positive definite matrices $A_1, \ldots, A_m$ and any fixed symmetric matrix Z. Based on (30), to prove (29) it remains to show that

$$\sum_k \operatorname{tr}(A_k) \operatorname{tr}\Big(Z A_k^{1/2} Z A_k^{1/2}\Big) \geq \sum_k \operatorname{tr}(A_k Z)^2. \tag{31}$$

We prove inequality (31) term-by-term, i.e., we show that

$$\operatorname{tr}(A_k) \operatorname{tr}\Big(Z A_k^{1/2} Z A_k^{1/2}\Big) = \operatorname{tr}(A_k) \operatorname{tr}\Big((Z A_k^{1/2})^2\Big) \geq \operatorname{tr}(A_k Z)^2. \tag{32}$$

To do this, we note the following inequality for all $X, Y \in \mathbb{S}^n$

$$\operatorname{tr}(XY)^2 \leq \operatorname{tr}\big(X^2\big) \cdot \operatorname{tr}\big(Y^2\big).$$

This follows from the Cauchy-Schwarz inequality applied to the trace-inner product over the space of symmetric matrices. Subsequently, one has

$$\operatorname{tr}(A_k Z)^2 = \operatorname{tr}\Big(A_k^{1/2} (A_k^{1/4} Z A_k^{1/4})\Big)^2 \leq \operatorname{tr}(A_k) \operatorname{tr}\Big((A_k^{1/4} Z A_k^{1/4})^2\Big),$$

which is exactly (32). □

8 Fixed-points are KKT

In this section we give the proof of Theorem 2.

Lemma 4. *Let $\mathcal{A}$ be the linear map $\mathcal{A}(Z) = (\operatorname{tr}(A_1 Z), \ldots, \operatorname{tr}(A_m Z))^\top$, where A_i are positive definite matrices, and let x be a vector with positive entries. Suppose B is a positive definite matrix satisfying $B = W(\mathcal{A}^\top x)W$, where $W = ([\mathcal{A}^\top \mathcal{A}](B))^{-1} \# B$. Then $\mathcal{A}^\top x = [\mathcal{A}^\top \mathcal{A}](B)$.*

Proof. Since B is positive definite, it follows that $[\mathcal{A}^\top \mathcal{A}](B) = \sum_{i=1}^m \langle A_i, B \rangle A_i$ is also positive definite, and thus $W = ([\mathcal{A}^\top \mathcal{A}](B))^{-1} \# B$ is positive definite. We then have that

$$\mathcal{A}^\top x = W^{-1} B W^{-1} = (B^{-1} \# [\mathcal{A}^\top \mathcal{A}](B)) B (B^{-1} \# [\mathcal{A}^\top \mathcal{A}](B)) = [\mathcal{A}^\top \mathcal{A}](B),$$

where the last equality follows by the unicity property of the matrix geometric mean (17). □

9 Comparison of MMU with Fast Projected Gradient Method

We briefly describe a numerical experiment comparing our MMU algorithm with earlier work. We specifically focus on the Fast Projected Gradient Method (FPGM) proposed by Vandaele et al. [37].

In this experiment, we generate a matrix of dimensions 20×20 where each entry is of the form $\mathrm{tr}(A_i B_j)$, where A_i and B_j are $r \times r$-dimensional matrices with $r = 4$ drawn from the Wishart distribution. (To sample from the Wishart matrix, we draw a random matrix G_i where each entry is from the standard normal distribution, and we compute the product $A_i = G_i G_i^T$.) We supply as input the rank $r = 4$ into the MMU algorithm and the FPGM and we run both algorithms over 50 iterations. We repeat these steps over 5 different random initializations for both methods.

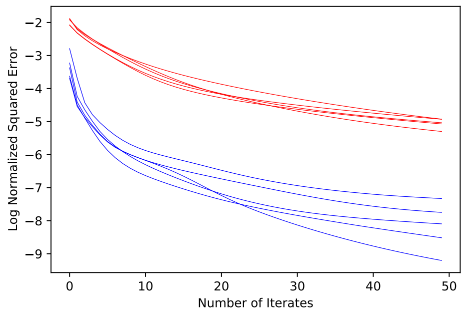

Figure 6: Comparison between MMU and FPGM method.

In Figure 6, we compare the log squared error loss using both methods. As the errors using both methods converge to zero, we conclude that both methods compute factorizations that are near optimal. Our results suggest that, for a fixed number of iterations, the FPGM computes factorizations with a much smaller error compared to the MMU algorithm. Nevertheless, these results do also suggest that the asymptotic rates achieved by both methods are comparable.

Our results are consistent with similar numerical experiments that compare the Multiplicative Update algorithm for NMF proposed by Lee and Seung with other methods; namely that the Multiplicative Update algorithm is not the most competitive (see [12], for example, and the references therein). One point we emphasize is that the MMU algorithm we present is the vanilla version – it would be interesting to explore accelerated variants of the MMU algorithm in a similar fashion that the FPGM is an accelerated of Projected Gradient Methods [12, 13].

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default TODO to [Yes], [No], or [NA]. You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes]See Section 6.
- Did you include the license to the code and datasets? [No]The code and the data are proprietary.
- Did you include the license to the code and datasets? [NA]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]

(b) Did you describe the limitations of your work? [Yes]
(c) Did you discuss any potential negative societal impacts of your work? [NA]
(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] The proof of Theorem 1 is in the main body. It relies on Lemma 3, which is stated and proved in the Appendix. The proof of Theorem 2 is found in the Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The essential instructions are provided in the numerical experimental section. The code will be uploaded to an online repository at a later date.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] An important feature of the MMU (our) algorithm we propose is that it does *not* require a tuning parameter. In the numerical experiment on the CBCL Face dataset, the choice of parameters were such that the number of degrees of freedom in all three methods are similary (either 27 or 28).
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] The PSD factorization task is non-convex so the performance of any algorithm is dependent on initialization. We describe the error bars (indirectly) in our plot in Figure 1.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] The experiments were conducted on a laptop, and we briefly state the specifications in the mainbody.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We used the CBCL Face Dataset [1].
   (b) Did you mention the license of the assets? [No] We do not mention the license because it appears to be customized. Nevertheless the license permits use of the dataset for academic research, and can be found on the cited URL.
   (c) Did you include any new assets either in the supplemental material or as a URL? [NA]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [NA]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [NA]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [NA]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [NA]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [NA]