Mathematical Programming (2019) 176:279–310
https://doi.org/10.1007/s10107-019-01370-7

FULL LENGTH PAPER

Series B

Computation of the maximum likelihood estimator in low-rank factor analysis

Koulik Khamaru[1] · **Rahul Mazumder**[2]

Received: 18 January 2018 / Accepted: 2 February 2019 / Published online: 2 March 2019

Abstract

Factor analysis is a classical multivariate dimensionality reduction technique popularly used in statistics, econometrics and data science. Estimation for factor analysis is often carried out via the maximum likelihood principle, which seeks to maximize the Gaussian likelihood under the assumption that the positive definite covariance matrix can be decomposed as the sum of a low-rank positive semidefinite matrix and a diagonal matrix with nonnegative entries. This leads to a challenging rank constrained nonconvex optimization problem, for which very few reliable computational algorithms are available. We reformulate the low-rank maximum likelihood factor analysis task as a nonlinear nonsmooth semidefinite optimization problem, study various structural properties of this reformulation; and propose fast and scalable algorithms based on difference of convex optimization. Our approach has computational guarantees, gracefully scales to large problems, is applicable to situations where the sample covariance matrix is rank deficient and adapts to variants of the maximum likelihood problem with additional constraints on the model parameters. Our numerical experiments validate the usefulness of our approach over existing state-of-the-art approaches for maximum likelihood factor analysis.

Keywords Low-rank constraint · Maximum likelihood factor analysis · Difference of convex optimization · Semidefinite optimization · Spectral functions · Sparsity

Mathematics Subject Classification 90-08 · 90C22 · 90C30

✉ Rahul Mazumder
rahulmaz@mit.edu

Koulik Khamaru
koulik@berkeley.edu

[1] Department of Statistics, University of California Berkeley, Berkeley, USA

[2] MIT Sloan School of Management, Operations Research Center and Center for Statistics, Massachusetts Institute of Technology, Cambridge, USA

1 Introduction

Factor analysis (FA) [2,7,26], a generalization of principal component analysis, is a classical dimensionality reduction technique for multivariate data that was introduced more than a hundred years ago. FA is popularly used to understand the correlation structure among a collection of observed random variables, in terms of a smaller number of common factors. In a typical FA model, we assume that the (mean centered) observed random vector $\mathbf{x} \in \Re^{p\times 1}$ may be expressed in the form: $\mathbf{x} = \mathbf{L}\mathbf{f} + \mathbf{u}$, where, $\mathbf{L} := ((\ell_{ij})) \in \Re^{p\times r}$ is a matrix of factor loadings, $\mathbf{f} \in \Re^{r\times 1}$ is a random vector of scores and $\mathbf{u} \in \Re^{p\times 1}$ is a vector of uncorrelated variables. We assume that $\mathbf{f}$ and $\mathbf{u}$ have zero means, are uncorrelated; and without loss of generality, we set the covariance matrix of $\mathbf{f}$ as the identity matrix (of size $r \times r$) $\mathrm{Cov}(\mathbf{f}) = \mathbf{I}$. This leads to the following decomposition:

$$\mathrm{Cov}(\mathbf{x}) := \mathbf{\Sigma} = \mathbf{L}\mathbf{L}^\top + \mathbf{\Psi}, \tag{1}$$

where, $\mathrm{Cov}(\mathbf{u}) := \mathbf{\Psi} = \mathrm{diag}(\psi_1, \ldots, \psi_p)$ is a diagonal matrix with entries $\{\psi_i\}_1^p$. Decomposition (1) suggests that the population covariance matrix $\mathbf{\Sigma} := ((\sigma_{ij}))$, can be written as the sum of a low-rank positive semidefinite (PSD) matrix and a diagonal matrix $\mathbf{\Psi}$ with nonnegative entries. In particular, this implies that the variance of x_i (i.e., $\mathrm{Var}(x_i)$) can be decomposed as $\mathrm{Var}(x_i) = \sigma_{ii} = \sum_{k=1}^r \ell_{ik}^2 + \psi_i$. In this decomposition, $\sum_{k=1}^r \ell_{ik}^2$ represents the (part of the) variance of x_i which is shared with other variables via the common factors (this is called the *communality*); and ψ_i represents the (part of the) variance of x_i that is not shared with the other variables (this is known as *specific* or *unique* variance).

One of the most popular FA estimation methods is the Gaussian maximum likelihood (ML) procedure [2,7,26]. Given n multivariate samples $\mathbf{x}_i, i = 1, \ldots, n$, assumed to be mean-centered, the task is to minimize the negative log-likelihood with respect to the parameter $\mathbf{\Sigma}$ that is of the form (1). This leads to the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \mathcal{L}(\mathbf{\Sigma}) := \; -\log\det(\mathbf{\Sigma}^{-1}) + \mathrm{tr}(\mathbf{\Sigma}^{-1}\mathbf{S}) \\ \text{s.t.} \quad & \mathbf{\Sigma} = \mathbf{\Psi} + \mathbf{L}\mathbf{L}^\top \\ & \mathbf{\Psi} = \mathrm{diag}(\psi_1, \ldots, \psi_p) \succeq \epsilon\mathbf{I}, \end{aligned} \tag{2}$$

where, $\mathbf{S} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^\top$ is the sample covariance matrix, $\mathbf{I}$ is the identity matrix (of size $p \times p$); $\mathbf{\Psi} \in \Re^{p\times p}$, $\mathbf{L} \in \Re^{p\times r}$ and $\mathbf{\Sigma}$ are the optimization variables and the notation $\mathbf{A} \succeq \mathbf{B}$ means that $\mathbf{A} - \mathbf{B}$ is PSD. Here, ϵ is a small positive constant specified a-priori, that satisfies $\epsilon < \min_{i=1,\ldots,p} s_{ii}$. To gain further intuition regarding this choice, note that if n is sufficiently large, then $s_{ii} \approx \sigma_{ii} = \psi_i + \sum_{k=1}^r \ell_{ik}^2 \geq \psi_i$, for all i. In addition, a lower bound on ψ_is ensures that Problem (2) is bounded below.[1]

[1] Indeed, $\mathbf{\Psi} \succeq \epsilon\mathbf{I}$ implies that $\mathbf{\Sigma} \succeq \epsilon\mathbf{I} \succ \mathbf{0}$. Thus, $-\log\det(\mathbf{\Sigma}^{-1}) \geq p\log(\epsilon)$ and $\mathrm{tr}\left(\mathbf{\Sigma}^{-1}\mathbf{S}\right) \geq 0$ which shows that Problem (2) is bounded below. Note that Problem (2) with $\epsilon = 0$ need not have a finite solution, i.e., the ML solution need not exist. Note that if for some i, we have $\psi_i \to \infty$ then $\mathcal{L}(\mathbf{\Sigma}) \to \infty$, a similar argument applies if $\mathbf{L}\mathbf{L}^\top$ becomes unbounded. Thus the infimum of Problem (2) is attained when $\epsilon > 0$.

We note that Problem (2) can be rewritten using a new variable $\mathbf{\Theta} = \mathbf{L}\mathbf{L}^\top \in \Re^{p\times p}$ as:

$$\begin{aligned} \text{minimize} \quad & \mathcal{L}(\mathbf{\Sigma}) \\ \text{s.t.} \quad & \mathbf{\Sigma} = \mathbf{\Psi} + \mathbf{\Theta} \\ & \text{rank}(\mathbf{\Theta}) \leq r \\ & \mathbf{\Theta} \succeq \mathbf{0}, \\ & \mathbf{\Psi} = \text{diag}(\psi_1, \ldots, \psi_p) \succeq \epsilon \mathbf{I}, \end{aligned} \tag{3}$$

where, the optimization variables are $\mathbf{\Theta}$, $\mathbf{\Psi}$, $\mathbf{\Sigma}$. Due to the presence of the rank constraint in formulation (3), in what follows, we will often refer to Problem (2) as a rank constrained optimization problem. Observe that Problem (2) is nonconvex since (a) the objective function $\mathcal{L}(\mathbf{\Sigma})$ is *not* convex in $\mathbf{\Sigma}$ [11] and (b) the equality constraint $\mathbf{\Sigma} = \mathbf{\Psi} + \mathbf{L}\mathbf{L}^\top$ is nonconvex.

Related work There is a significant body of work in FA, a concept that originated more than a hundred years ago [36]. Here, we present a selective overview that is relevant for this work—important contributions in FA have been nicely documented in [2,5,7,26]. Despite being a problem of fundamental importance in statistical estimation, not much is known about the computational properties of FA. Many popular off-the-shelf implementations for ML factor analysis (e.g., the routinely used algorithms available in Matlab and R) are quite unstable.[2] These algorithms are based on rather ad hoc computational methods, and often lead to negative variance estimates which are highly problematic from a statistical inference viewpoint. This is perhaps not surprising, given that the basic problem underlying ML factor analysis is a difficult (nonconvex) optimization problem and there has been limited work in developing mathematical optimization based algorithms for this problem. We note that it is also difficult to generalize existing algorithms to address variants of the ML factor analysis problem, with the inclusion of additional problem-specific constraints. Anderson and Lawley and Maxwell [2,23] present a nice overview of classical algorithms used for ML factor analysis. Some of the current state-of-the-art computational approaches for ML factor analysis are based on the seminal contribution of [19]. This approach assumes that $\mathbf{S}$ is of full rank and $\mathbf{L}$ has rank exactly equal to r. It is based on an ad hoc gradient descent based algorithm on an objective function that is locally differentiable. Recently, the authors of [31] provide necessary and sufficient conditions for existence of a solution of Problem (2) with $\epsilon = 0$, however, they do not discuss any computational algorithms. Another popular approach for ML factor analysis is based on the EM algorithm [9,33]. Some publicly available implementations of the EM-type methods apply to cases where $\mathbf{S}$ need not be of full rank.

Not all methods in FA are based on the ML framework. In other approaches, one seeks to estimate a matrix $\mathbf{\Sigma}$ of the form $\mathbf{\Sigma} = \mathbf{\Psi} + \mathbf{L}\mathbf{L}^\top$, which is close to the sample covariance matrix $\mathbf{S}$, in terms of the Frobenius norm or some weighted variant of this norm. Some popular methods in the literature are the minimum residual FA, principal axis method, principal component method, minimum trace FA, among others—see [7,8,35] for more description on these approaches. Fairly recently [8], propose a spatial

[2] We have observed this in our experiments and they are reported in our section on numerical experiments.

branch and bound method for minimizing the squared Frobenius norm $\|\mathbf{S} - \mathbf{\Sigma}\|_F^2$ where, $\mathbf{\Sigma}$ is of the form (1) with an additional constraint $\mathbf{S} - \mathbf{\Sigma} \succeq \mathbf{0}$. Saunderson et al. [34] studies the noiseless decomposition for the FA problem, using nuclear norm relaxations of the rank constraint on $\mathbf{\Theta} := \mathbf{L}\mathbf{L}^\top$. The aforementioned line of work is different from the ML criterion in FA, as the data-fidelity measure is different. In this paper, our focus is on the computational properties of the ML problem (2).

Contributions The main contributions of this paper can be summarized as follows:

1. We propose a new computational framework for the task of (Gaussian) maximum likelihood estimation in factor analysis—a problem central to classical and modern multivariate statistical learning. The associated optimization problem is challenging: it is given by the minimization of a nonconvex function subject to a rank constraint and additional semidefinite constraints. We reformulate the problem as the minimization of a nonsmooth nonconvex function containing eigenvalues of a positive semidefinite matrix, subject to (simple) convexity constraints.
2. Using (convexity) properties of spectral functions, we show that the objective function can be expressed as a difference of convex functions; and is hence amenable to computational techniques in difference of convex optimization. The computational bottleneck of our algorithm is a low-rank singular value decomposition (SVD) of a $p \times p$ matrix, that needs to be performed for every iteration. Exploiting problem structure, we show that this can be computed with cost $O(\min\{n, p\}^2 \max\{n, p\})$. When compared to other publicly available implementations, an important advantage of our proposal is that it applies to the case where the sample covariance matrix is rank deficient.
3. We explore computational guarantees of our proposed algorithm in terms of reaching a first order stationary point. We demonstrate that on a series of numerical examples (including both real and synthetic datasets), our method significantly outperforms commonly used approaches for ML factor analysis in terms of (i) reduced computation time, (ii) obtaining a solution with better objective value; and (iii) superior numerical stability of the algorithm. To our knowledge, our proposal is one of the most scalable computational mathematical optimization approaches for ML factor analysis. Our approach also generalizes to instances of ML factor analysis where, $\mathbf{\Psi}$ is not necessarily diagonal.

Notation For a real symmetric matrix $\mathbf{A}_{p\times p}$, we will denote its eigenvalues by $\lambda_i(\mathbf{A}), i = 1, \ldots, p$ with $\lambda_i(\mathbf{A}) \geq \lambda_{i+1}(\mathbf{A})$ for all i; we will use the shorthand notation $\boldsymbol{\lambda}(\mathbf{A})$ to denote the vector of eigenvalues of $\mathbf{A}$. For a real postive semidefinite (PSD) matrix $\mathbf{A}$, we use $\mathbf{A}^{\frac{1}{2}}$ to denote its symmetric square root. If $\mathbf{A}$ is invertible and PSD, we use $\mathbf{A}^{-\frac{1}{2}}$ to denote the square root of $\mathbf{A}^{-1}$. Given a nonnegative integer m, we denote $1, \ldots, m$ by $[m]$. For a vector $\mathbf{x} \in \Re^d$ and scalars α, γ (with $\alpha \leq \gamma$), we define $[\alpha, \gamma]^d := \{\mathbf{x} : \alpha \leq x_i \leq \gamma, i \in [d]\}$; similarly, we define $(\alpha, \gamma]^d := \{\mathbf{x} : \alpha < x_i \leq \gamma, i \in [d]\}$. For any vector $\mathbf{a} \in \Re^p$, we use $\mathrm{diag}(\mathbf{a})$ to denote a $p \times p$ diagonal matrix with its diagonal entries being the coordinates of $\mathbf{a}$; whereas, for a $p \times p$ square matrix $\mathbf{A}$, $\mathrm{diag}(\mathbf{A})$ denotes the diagonal matrix with diagonal entries same as the diagonal entries of $\mathbf{A}$. For any matrix $\mathbf{A} := ((a_{ij})) \in \Re^{p\times p}$ and nonnegative integers $0 = p_0 < p_1 < \cdots < p_m = p$ (we will use the notation

$\mathbf{p}$ to denote the vector $(p_i)_{i=0}^m$) and $b \geq 1$, we use $\text{Blkdiag}_{\mathbf{p}}(\mathbf{A})$ and $\text{Banded}_b(\mathbf{A})$ to denote the following two $p \times p$ matrices:

$$\left[\text{Blkdiag}_{\mathbf{p}}(\mathbf{A})\right]_{ij} = \begin{cases} a_{ij} & \text{if } p_{k-1} < i, j \leq p_k \text{ for some } k \in [m] \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

$$[\text{Banded}_b(\mathbf{A})]_{ij} = \begin{cases} a_{ij} & \text{if } |i-j| \leq b, i \in [p] \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

We let $\mathbf{I}$ denote the identity matrix and $\mathbf{1}$ a vector of all ones (with dimension determined from the context). For a vector $\mathbf{a}$, we use the notation $\mathbf{a} \geq \mathbf{0}$ to denote componentwise inequality; for a matrix $\mathbf{A}$, we use the notation $\mathbf{A} \succeq \mathbf{0}$ (or $\succ \mathbf{0}$) to denote that the matrix $\mathbf{A}$ is positive semidefinite (respectively, positive definite). We will assume that all diagonal entries of the sample covariance matrix $\mathbf{S}$ are strictly greater than zero.

2 Methodology

We state a simple result (the proof of which is omitted) regarding the eigenvalues of the product of two matrices—a property that is used throughout the paper.

Proposition 1 *For any two real symmetric matrices* $\mathbf{A}, \mathbf{B} \in \Re^{p \times p}$, *we have* $\boldsymbol{\lambda}(\mathbf{AB}) = \boldsymbol{\lambda}(\mathbf{BA})$.

We present below a simple corollary of the above, that is used widely in the paper:

Corollary 1 *For any PSD matrix* $\boldsymbol{\Phi}$, *we have:*

$$\boldsymbol{\lambda}\left(\boldsymbol{\Phi}^{\frac{1}{2}}\mathbf{S}\boldsymbol{\Phi}^{\frac{1}{2}}\right) = \boldsymbol{\lambda}(\mathbf{S}\boldsymbol{\Phi}) = \boldsymbol{\lambda}(\boldsymbol{\Phi}\mathbf{S}) = \boldsymbol{\lambda}\left(\mathbf{S}^{\frac{1}{2}}\boldsymbol{\Phi}\mathbf{S}^{\frac{1}{2}}\right). \tag{6}$$

2.1 Reformulations

In this section, we present a reformulation of the rank constrained optimization Problem (2) to one that does not involve an explicit rank constraint: Proposition 2 presents a reformulation of Problem (2) as an optimization problem that only involves $\boldsymbol{\Psi}$. The resulting problem (7) is amenable to efficient optimization techniques based on difference of convex optimization. Proposition 3 provides bounds on an optimal solution of Problem (7). Recall that in Problem (2), we assume $\epsilon < \min_{i \in [p]} s_{ii}$.

Proposition 2 *(***a***) Problem* (2) *is equivalent to:*

$$\begin{aligned} \text{minimize} \quad & \left\{ \log\det(\boldsymbol{\Psi}) + \text{tr}\left(\mathbf{S}^*\right) + \sum_{i=1}^{r} \left(\log(\max\{1, \lambda_i^*\}) - \max\{1, \lambda_i^*\} + 1\right) \right\} \\ \text{s.t.} \quad & \boldsymbol{\Psi} = \text{diag}(\psi_1, \ldots, \psi_p) \succeq \epsilon \mathbf{I}, \end{aligned} \tag{7}$$

where, $\lambda_1^* \geq \lambda_2^* \geq \cdots \geq \lambda_p^*$ *denote the eigenvalues of* $\mathbf{S}^* := \boldsymbol{\Psi}^{-\frac{1}{2}}\mathbf{S}\boldsymbol{\Psi}^{-\frac{1}{2}}$; *and the optimization variables are* $\boldsymbol{\Psi}, \mathbf{S}^*$ *and* $\{\lambda_i^*\}_{i \geq 1}$.

(**b**) *Suppose* $\hat{\boldsymbol{\Psi}}$ *is a minimizer of Problem* (7) *and* $\hat{\mathbf{L}} = \hat{\boldsymbol{\Psi}}^{\frac{1}{2}} \left[\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2, \ldots, \hat{\mathbf{z}}_r\right]$ *where,* $\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2, \ldots, \hat{\mathbf{z}}_r$ *are eigenvectors of* $\hat{\boldsymbol{\Psi}}^{-\frac{1}{2}} \mathbf{S} \hat{\boldsymbol{\Psi}}^{-\frac{1}{2}}$ *corresponding to its top* r *eigenvalues* $\hat{\lambda}_i^*, i \in [r]$ *with* $\|\hat{\mathbf{z}}_i\|^2 = \max\left\{1, \hat{\lambda}_i^*\right\} - 1$ *for all* $i \in [r]$. *Then* $(\hat{\boldsymbol{\Psi}}, \hat{\mathbf{L}})$ *is a minimizer of Problem* (2).

Proof Part (a): We first minimize Problem (2) with respect to $\mathbf{L}$ for a fixed $\boldsymbol{\Psi}$. A simple application of the Sherman Woodbury formula with some rearrangement gives:

$$\begin{aligned}
\boldsymbol{\Sigma}^{-1} &= \left(\boldsymbol{\Psi} + \mathbf{L}\mathbf{L}^\top\right)^{-1} \\
&= \boldsymbol{\Psi}^{-1} - \boldsymbol{\Psi}^{-1}\mathbf{L}\left(\mathbf{I} + \mathbf{L}^\top\boldsymbol{\Psi}^{-1}\mathbf{L}\right)^{-1}\mathbf{L}^\top\boldsymbol{\Psi}^{-1} \\
&= \boldsymbol{\Psi}^{-1} - \boldsymbol{\Psi}^{-\frac{1}{2}}\boldsymbol{\Psi}^{-\frac{1}{2}}\mathbf{L}(\mathbf{I} + \mathbf{L}^\top\boldsymbol{\Psi}^{-\frac{1}{2}}\boldsymbol{\Psi}^{-\frac{1}{2}}\mathbf{L})^{-1}\mathbf{L}^\top\boldsymbol{\Psi}^{-\frac{1}{2}}\boldsymbol{\Psi}^{-\frac{1}{2}}. \qquad (8)
\end{aligned}$$

Writing $\mathbf{L}^* = \boldsymbol{\Psi}^{-\frac{1}{2}}\mathbf{L}$ in the last line of display (8), we get:

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{\Psi}^{-1} - \boldsymbol{\Psi}^{-\frac{1}{2}}\mathbf{L}^*(\mathbf{I} + (\mathbf{L}^*)^\top\mathbf{L}^*)^{-1}(\mathbf{L}^*)^\top\boldsymbol{\Psi}^{-\frac{1}{2}}.$$

The above implies that:

$$\begin{aligned}
\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}) &= \mathrm{tr}(\boldsymbol{\Psi}^{-1}\mathbf{S}) - \mathrm{tr}\left(\boldsymbol{\Psi}^{-\frac{1}{2}}\mathbf{L}^*\left(\mathbf{I} + (\mathbf{L}^*)^\top\mathbf{L}^*\right)^{-1}(\mathbf{L}^*)^\top\boldsymbol{\Psi}^{-\frac{1}{2}}\mathbf{S}\right) \\
&= \mathrm{tr}(\boldsymbol{\Psi}^{-\frac{1}{2}}\mathbf{S}\boldsymbol{\Psi}^{-\frac{1}{2}}) - \mathrm{tr}\left((\mathbf{L}^*)^\top\boldsymbol{\Psi}^{-\frac{1}{2}}\mathbf{S}\boldsymbol{\Psi}^{-\frac{1}{2}}\mathbf{L}^*\left(\mathbf{I} + (\mathbf{L}^*)^\top\mathbf{L}^*\right)^{-1}\right) \qquad (9) \\
&= \mathrm{tr}(\mathbf{S}^*) - \mathrm{tr}\left((\mathbf{L}^*)^\top\mathbf{S}^*\mathbf{L}^*\left(\mathbf{I} + (\mathbf{L}^*)^\top\mathbf{L}^*\right)^{-1}\right) \text{ (Using, } \mathbf{S}^* = \boldsymbol{\Psi}^{-\frac{1}{2}}\mathbf{S}\boldsymbol{\Psi}^{-\frac{1}{2}})
\end{aligned}$$

where, in the second and third lines of display (9) we (repeatedly) used the fact that $\mathrm{tr}(\mathbf{AB}) = \mathrm{tr}(\mathbf{BA})$. In addition, note that:

$$\begin{aligned}
-\log\det(\boldsymbol{\Sigma}^{-1}) = \log\det(\boldsymbol{\Sigma}) &= \log\det(\boldsymbol{\Psi} + \mathbf{L}\mathbf{L}^\top) \\
&= \log\det(\boldsymbol{\Psi}) + \log\det\left(\mathbf{I} + \mathbf{L}^{*\top}\mathbf{L}^*\right). \qquad (10)
\end{aligned}$$

We denote the objective in Problem (2) by $h(\boldsymbol{\Psi}, \mathbf{L}) := \log\det(\mathbf{L}\mathbf{L}^\top + \boldsymbol{\Psi}) + \mathrm{tr}((\mathbf{L}\mathbf{L}^\top + \boldsymbol{\Psi})^{-1}\mathbf{S})$. Using (9), (10) in the objective function of Problem (2), we get the following equivalent reformulation of Problem (2):

$$\begin{aligned}
\text{minimize} \quad & \log\det(\boldsymbol{\Psi}) + \log\det\left(\mathbf{I} + \mathbf{L}^{*\top}\mathbf{L}^*\right) + \mathrm{tr}\left(\mathbf{S}^*\right) \\
& \qquad - \mathrm{tr}\left(\mathbf{L}^{*\top}\mathbf{S}^*\mathbf{L}^*\left(\mathbf{I} + \mathbf{L}^{*\top}\mathbf{L}^*\right)^{-1}\right) \\
\text{s.t.} \quad & \boldsymbol{\Psi} = \mathrm{diag}(\psi_1, \ldots, \psi_p) \succeq \epsilon\mathbf{I}, \qquad (11)
\end{aligned}$$

where, recall that we use the notation: $\mathbf{L}^* = \boldsymbol{\Psi}^{-\frac{1}{2}}\mathbf{L}$ and $\mathbf{S}^* = \boldsymbol{\Psi}^{-\frac{1}{2}}\mathbf{S}\boldsymbol{\Psi}^{-\frac{1}{2}}$. The optimization variables in Problem (11) are $\boldsymbol{\Psi}, \mathbf{L}$ (and consequently, $\mathbf{L}^*, \mathbf{S}^*$). Note that $h(\boldsymbol{\Psi}, \mathbf{L}) = h(\boldsymbol{\Psi}, \mathbf{LU})$ for any orthogonal matrix $\mathbf{U}$. So we can substitute $\mathbf{L}$ by $\mathbf{LU}$ in Problem (11). We choose $\mathbf{U}$ such that the columns of $\mathbf{L}^*$ are orthogonal or zero vectors. Note that the partial derivative of $h(\boldsymbol{\Psi}, \mathbf{L}) = \log\det(\mathbf{LL}^\top + \boldsymbol{\Psi}) + \mathrm{tr}((\mathbf{LL}^\top + \boldsymbol{\Psi})^{-1}\mathbf{S})$ w.r.t. $\mathbf{L}$ is given by:

$$\frac{\partial h(\boldsymbol{\Psi}, \mathbf{L})}{\partial \mathbf{L}} = 2\boldsymbol{\Sigma}^{-1}(\boldsymbol{\Sigma} - \mathbf{S})\boldsymbol{\Sigma}^{-1}\mathbf{L}. \tag{12}$$

Using $\boldsymbol{\Sigma} = \boldsymbol{\Psi} + \mathbf{LL}^\top \succ \mathbf{0}$, we note that $\partial h(\boldsymbol{\Psi}, \mathbf{L})/\partial \mathbf{L} = 0$ iff $\mathbf{L} = \mathbf{S}(\boldsymbol{\Psi} + \mathbf{LL}^\top)^{-1}\mathbf{L}$. Algebraic manipulations (using (8)) show that this condition is equivalent to the following (see Sect. A.2):

$$\mathbf{S}^*\mathbf{L}^* = \mathbf{L}^*\left(\mathbf{I} + \mathbf{L}^{*\top}\mathbf{L}^*\right). \tag{13}$$

Note that when $\psi_i < s_{ii}$ for all $i \in [p]$ (such a ψ_i is feasible for Problem (2) as $\epsilon < \min_{i\in[p]} s_{ii}$), the diagonal entries of $\mathbf{S}^*$ satisfy $s^*_{ii} = s_{ii}/\psi_i > 1$ for all i. Since $\lambda_1(\mathbf{S}^*) \geq \mathrm{tr}(\mathbf{S}^*)/p > 1$—the largest eigenvalue of $\mathbf{S}^*$ is larger than one. Since we choose the columns of $\mathbf{L}^*$ to be pairwise orthogonal or zero vectors, it follows that $\mathbf{I} + \mathbf{L}^{*\top}\mathbf{L}^*$ is a diagonal matrix with every diagonal entry greater than or equal to one. This means that (13) is a collection of eigenvector equations for the matrix $\mathbf{S}^*$. Since $\lambda_1(\mathbf{S}^*) > 1$, it follows that the system (13) has at least one nonzero solution in $\mathbf{L}^*$.

Let us denote the columns of $\mathbf{L}^*$ by $\mathbf{z}_i, i \in [r]$. The part of the function $h(\boldsymbol{\Psi}, \mathbf{L})$ in display (11), that depends upon $\mathbf{L}^*$ is given by:

$$g(\mathbf{L}^*) := \sum_{i=1}^{r}\left(\log(1 + \mathbf{z}_i^\top\mathbf{z}_i) - \frac{\mathbf{z}_i^\top\mathbf{S}^*\mathbf{z}_i}{1 + \mathbf{z}_i^\top\mathbf{z}_i}\right). \tag{14}$$

Since $\mathbf{z}_i$s are pairwise orthogonal or zero vectors, it follows from equation (13) that

$$\mathbf{S}^*\mathbf{z}_i = \beta_i\mathbf{z}_i \quad \text{and} \quad \beta_i = 1 + \mathbf{z}_i^\top\mathbf{z}_i, \quad i \in [r]. \tag{15}$$

Note that in the above equation, either $\beta_i = 1$ with $\mathbf{z}_i = 0$ or $\beta_i > 1$ and β_i equals some eigenvalue of $\mathbf{S}^*$ with eigenvector $\mathbf{z}_i$—thus (14) becomes

$$g(\mathbf{L}^*) = \sum_{i=1}^{r}(\log(\beta_i) - \beta_i + 1). \tag{16}$$

Note that $\beta \mapsto \log(\beta) - \beta + 1$ is strictly decreasing for all $\beta \geq 1$. So it is easy to see that (16) is minimized for $\beta_i = \max\{1, \lambda^*_i\}$ for $i \in [r]$ where, $\lambda^*_1 \geq \cdots \geq \lambda^*_r$ are the top r eigenvalues of $\mathbf{S}^*$. The optimal choice of $\mathbf{z}_i$ is given by: $\mathbf{z}_i = 0$ when $\beta_i = 1$; when $\beta_i > 1$, $\mathbf{z}_i$ is an eigenvector of $\mathbf{S}^*$ with eigenvalue λ^*_i and we have that $\mathbf{z}_i^\top\mathbf{z}_i = \max\{1, \lambda^*_i\} - 1$.

Finally, we note that

$$\min_{\boldsymbol{\Psi}\succeq\epsilon\mathbf{I},\mathbf{L}} h(\boldsymbol{\Psi}, \mathbf{L}) = \min_{\boldsymbol{\Psi}\succeq\epsilon\mathbf{I}}\left\{\min_{\mathbf{L}} h(\boldsymbol{\Psi}, \mathbf{L})\right\}. \tag{17}$$

Substituting the value of $\mathbf{L}$ that minimizes the inner minimization problem above (in the right hand side of the above display), into the objective function $h(\mathbf{\Psi}, \mathbf{L})$, we obtain formulation (7).

Part (b): The proof of this part is a consequence of the proof of Part (a). □

The method of minimizing the objective function w.r.t. $\mathbf{L}$ with $\mathbf{\Psi}$ held fixed, is inspired by the classical work of [19,21,22]—this line of work however, assumes $\mathbf{S}$ to be of full rank. Since we do not assume $\mathbf{S}$ to have full rank, our derivation is different. Robertson and Symons [31] investigate the existence of ML solutions for a general $\mathbf{S}$—however, no computational algorithms are presented. Note that the expression (7) does not appear in [31]. Formulation (7) plays a key role in developing algorithms for Problem (2), a main focus of this paper.

Proposition 3 shows that any solution of Problem (7) is bounded above. Recall that we assume $\epsilon < \min_{i\in[p]} s_{ii}$.

Proposition 3 *Let $\hat{\mathbf{\Psi}}$ be a solution of Problem (7). Then we have $s_{ii} \geq \hat{\psi}_i \geq \epsilon$ for all $i \in [p]$.*

Proof Consider $\mathbf{\Psi} \succeq \epsilon \mathbf{I}$ that is feasible for Problem (7).

Setting (12) to zero and with some simplification we get (see (59) in Sect. A.2)

$$\mathbf{L} = \mathbf{S}\mathbf{\Psi}^{-1}\mathbf{L}\left(\mathbf{I} + \mathbf{L}^\top \mathbf{\Psi}^{-1}\mathbf{L}\right)^{-1}. \tag{18}$$

Since (13) has a nontrivial solution in $\mathbf{L}^*$, it follows that (18) has a nontrivial solution in $\mathbf{L}$. Moreover, using (8) in place of $\mathbf{\Sigma}^{-1}$, the expression $\mathbf{S}\mathbf{\Sigma}^{-1}$ simplifies as:

$$\begin{aligned}
\mathbf{S}\mathbf{\Sigma}^{-1} &= \mathbf{S}\mathbf{\Psi}^{-1} - \left(\mathbf{S}\mathbf{\Psi}^{-1}\mathbf{L}\left(\mathbf{I} + \mathbf{L}^\top \mathbf{\Psi}^{-1}\mathbf{L}\right)^{-1}\right)\left(\mathbf{L}^\top \mathbf{\Psi}^{-1}\right) \\
&= \mathbf{S}\mathbf{\Psi}^{-1} - \mathbf{L}\mathbf{L}^\top \mathbf{\Psi}^{-1} \qquad \text{(Using expression of } \mathbf{L} \text{ from rhs of (18))} \\
&= \mathbf{S}\mathbf{\Psi}^{-1} - (\mathbf{\Sigma} - \mathbf{\Psi})\,\mathbf{\Psi}^{-1} \qquad \text{(Since, } \mathbf{\Sigma} = \mathbf{L}\mathbf{L}^\top + \mathbf{\Psi}\text{)} \\
&= \mathbf{S}\mathbf{\Psi}^{-1} - \mathbf{\Sigma}\mathbf{\Psi}^{-1} + \mathbf{I}. \qquad (19)
\end{aligned}$$

Note that we have the following expression for $\mathbf{\Sigma}^{-1}(\mathbf{\Sigma} - \mathbf{S})\,\mathbf{\Sigma}^{-1}$:

$$\begin{aligned}
\mathbf{\Sigma}^{-1}(\mathbf{\Sigma} - \mathbf{S})\,\mathbf{\Sigma}^{-1} &= \mathbf{\Sigma}^{-1} - \mathbf{\Sigma}^{-1}(\mathbf{S}\mathbf{\Sigma}^{-1}) \\
&= \mathbf{\Sigma}^{-1} - \mathbf{\Sigma}^{-1}\left(\mathbf{S}\mathbf{\Psi}^{-1} - \mathbf{\Sigma}\mathbf{\Psi}^{-1} + \mathbf{I}\right) \\
&= -\left(\mathbf{\Sigma}^{-1}\mathbf{S}\right)\mathbf{\Psi}^{-1} + \mathbf{\Psi}^{-1} \\
&= -\left(\mathbf{\Psi}^{-1}\mathbf{S} - \mathbf{\Psi}^{-1}\mathbf{\Sigma} + \mathbf{I}\right)\mathbf{\Psi}^{-1} + \mathbf{\Psi}^{-1} \\
&= \mathbf{\Psi}^{-1}(\mathbf{\Sigma} - \mathbf{S})\,\mathbf{\Psi}^{-1}. \qquad (20)
\end{aligned}$$

where, the second line follows by using the expression for $\mathbf{S}\mathbf{\Sigma}^{-1}$ from (19); and the fourth line follows by using the same expression for $\mathbf{\Sigma}^{-1}\mathbf{S} = (\mathbf{S}\mathbf{\Sigma}^{-1})^\top$.

Since $\partial h(\mathbf{L}, \boldsymbol{\Psi})/\partial \boldsymbol{\psi} = \mathrm{diag}(\boldsymbol{\Sigma}^{-1}(\boldsymbol{\Sigma} - \mathbf{S})\boldsymbol{\Sigma}^{-1})$, using the last line of display (20), the expression for the ith entry of $\partial h(\mathbf{L}, \boldsymbol{\Psi})/\partial \boldsymbol{\psi}$ is given by:

$$\frac{\partial h(\mathbf{L}, \boldsymbol{\Psi})}{\partial \psi_i} = \frac{(\sigma_{ii} - s_{ii})}{\psi_i^2}. \tag{21}$$

We consider two cases, depending upon whether an optimal solution $\hat{\psi}_i$ satisfies: $\hat{\psi}_i > \epsilon$ or $\hat{\psi}_i = \epsilon$. If $\hat{\psi}_i > \epsilon$, then $\partial h(\mathbf{L}, \boldsymbol{\Psi})/\partial \psi_i = 0$: hence $\sigma_{ii} = s_{ii}$ which implies that $s_{ii} \geq \hat{\psi}_i > \epsilon$ (since, $\sigma_{ii} = \sum_{k=1}^{r} \ell_{ik}^2 + \psi_i$). Otherwise, if $\hat{\psi}_i = \epsilon$ then $\partial h(\mathbf{L}, \boldsymbol{\Psi})/\partial \psi_i \leq 0$: this leads to $s_{ii} \geq \hat{\psi}_i = \epsilon$. This completes the proof. □

Corollary 2 presents another equivalent representation of Problem (7) by a simple change of variables $\boldsymbol{\Phi} := \boldsymbol{\Psi}^{-1}$. Below, for notational convenience, we use the shorthand $\boldsymbol{\Phi} := \mathrm{diag}\left(\phi_1, \ldots, \phi_p\right)$ and $\boldsymbol{\phi} = \left(\phi_1, \ldots, \phi_p\right)$.

Corollary 2 *Problem (7) is equivalent to the following optimization problem in $\boldsymbol{\phi}$:*

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{\phi}) := \sum_{i=1}^{p} (-\log \phi_i + s_{ii}\phi_i) \\ & + \sum_{i=1}^{r} \left(\log\left(\max\{1, \lambda_i^*\}\right) - \max\{1, \lambda_i^*\} + 1\right) \\ \text{s.t.} \quad & \mathbf{0} \prec \boldsymbol{\Phi} = \mathrm{diag}(\phi_1, \ldots, \phi_p) \preceq \frac{1}{\epsilon}\mathbf{I}, \end{aligned} \tag{22}$$

where, $\lambda_i^*, i \in [r]$ are the top r eigenvalues of $\mathbf{S}^* = \boldsymbol{\Phi}^{\frac{1}{2}}\mathbf{S}\boldsymbol{\Phi}^{\frac{1}{2}}$. If $\hat{\boldsymbol{\Psi}}$ is a solution to Problem (7), then $\hat{\boldsymbol{\Psi}} = \hat{\boldsymbol{\Phi}}^{-1}$ where, $\hat{\boldsymbol{\Phi}}$ is a solution to Problem (22).

Remark 1 Problem (7) (and Problem (22)) is a minimization problem in $\boldsymbol{\Psi}$ (respectively, $\boldsymbol{\Phi}$), unlike the rank constrained Problem (2) with variables $\mathbf{L}$ and $\boldsymbol{\Psi}$. Note that Problem (22) is nonconvex due to the nonconvex objective function, though the constraints are convex. Corollary 3 shows that the objective function $f(\boldsymbol{\phi})$ appearing in Problem (22) is neither convex nor concave, but it can be written as a difference of two simple convex functions.

2.2 Expressing Problem (22) as a difference of convex functions

Here we show via Propositions 4 and 5 that the objective function in Problem (22) can be written as a difference of two convex functions (Corollary 3). This observation makes it possible to use algorithms based on difference of convex optimization, to get good solutions to Problem (22). Proposition 6 shows that when $\mathbf{S}$ is of full rank, the objective function in Problem (22) can be expressed purely in terms of the eigenvalues of $\mathbf{S}^*$.

Let $y_{(1)} \geq \cdots \geq y_{(p)}$ be an ordering of $\{y_i\}_1^p \in [0, \infty)^p$ and define:

$$H_r(\mathbf{y}) := \sum_{i=1}^{r} \left(\log\left(\max\{1, y_{(i)}\}\right) - \max\{1, y_{(i)}\} + 1\right). \tag{23}$$

The following proposition shows that $\mathbf{y} \mapsto H_r(\mathbf{y})$ is concave on $\mathbf{y} \geq \mathbf{0}$.

Proposition 4 *For any $r \in [p]$, the function $H_r(\mathbf{y})$ as defined in (23), is concave on $\mathbf{y} \geq \mathbf{0}$.*

Proof We first establish that $H_r(\mathbf{y})$ admits the following representation:

$$\begin{aligned} H_r(\mathbf{y}) = \min_{\mathbf{w}} \widetilde{H}(\mathbf{w}; \mathbf{y}) &:= \sum_{i=1}^{p} w_i \left(\log\left(\max\{1, y_i\}\right) - \max\{1, y_i\} + 1\right) \\ \text{s.t.} \quad \sum_{i=1}^{p} w_i &= r, \quad 0 \leq w_i \leq 1, i \in [p], \end{aligned} \tag{24}$$

where, the objective function is the linear functional $\mathbf{w} \mapsto \widetilde{H}(\mathbf{w}; \mathbf{y})$. To see why this is true, note that the scalar function $y \mapsto \log(\max\{1, y\}) - \max\{1, y\} + 1$ is decreasing on $y \geq 0$. Hence the sum $\sum_{i=1}^{p} w_i \left(\log\left(\max\{1, y_i\}\right) - \max\{1, y_i\} + 1\right)$ will be minimized for a choice: $w_i = 1$ whenever y_i is one of the top r elements among $y_1, \ldots, y_p$; and $w_i = 0$ for all other choices of $i \in [p]$. This justifies representation (24).

For any $\Re \ni y \geq 0$, note that $y \mapsto \log(\max\{1, y\}) - \max\{1, y\} + 1$ is *concave*. Hence, for every fixed $\Re^p \ni \mathbf{w} \geq \mathbf{0}$, the function

$$(y_1, \ldots, y_p) \mapsto \sum_{i=1}^{p} w_i \left(\log\left(\max\{1, y_i\}\right) - \max\{1, y_i\} + 1\right)$$

is concave on $\mathbf{y} \geq \mathbf{0}$. Since the point-wise infimum of a family of concave functions is concave [11], $\mathbf{y} \mapsto H_r(\mathbf{y})$ is concave on $\mathbf{y} \geq \mathbf{0}$. □

Proposition 5 *For any $r \in [p]$, the function*

$$\boldsymbol{\phi} \mapsto h(\boldsymbol{\phi}) := \sum_{i=1}^{r} \left(\log\left(\max\{1, \lambda_i^*\}\right) - \max\{1, \lambda_i^*\} + 1\right) \tag{25}$$

is concave on $\boldsymbol{\phi} \geq \mathbf{0}$; where, $\{\lambda_i^*\}_1^p$ are the eigenvalues of $\mathbf{S}^*$.

Proof Note $\boldsymbol{\lambda}(\mathbf{S}^*) = \boldsymbol{\lambda}\left(\mathbf{S}^{\frac{1}{2}} \boldsymbol{\Phi} \mathbf{S}^{\frac{1}{2}}\right)$ (cf Corollary 1). By a classical result due to Davis and Lewis [13,25], the following mapping

$$\mathbf{S}^{\frac{1}{2}} \boldsymbol{\Phi} \mathbf{S}^{\frac{1}{2}} \mapsto \sum_{i=1}^{r} \left(\log\left(\max\{1, \lambda_i^*\}\right) - \max\{1, \lambda_i^*\} + 1\right) \tag{26}$$

is concave in $\mathbf{S}^{\frac{1}{2}}\boldsymbol{\Phi}\mathbf{S}^{\frac{1}{2}}$ if and only if the function (23) is symmetric[3] and concave in $\mathbf{y}$ on $\mathbf{y} \geq \mathbf{0}$. It is easy to see that the function in (23) is symmetric; and concavity follows from Proposition 4. So we conclude that the map in (26) is concave in $\mathbf{S}^{\frac{1}{2}}\boldsymbol{\Phi}\mathbf{S}^{\frac{1}{2}}$. The linearity of the map $\boldsymbol{\phi} \mapsto \mathbf{S}^{\frac{1}{2}}\boldsymbol{\Phi}\mathbf{S}^{\frac{1}{2}}$ implies that $h(\boldsymbol{\phi})$ is concave in $\boldsymbol{\phi}$ on $\boldsymbol{\phi} \geq \mathbf{0}$. This completes the proof of the proposition. □

Corollary 3 *For any $\boldsymbol{\phi} > \mathbf{0}$, $f(\boldsymbol{\phi})$ can be written as the difference of two convex functions, $f_i(\boldsymbol{\phi}), i = 1, 2$ that is: $f(\boldsymbol{\phi}) = f_1(\boldsymbol{\phi}) - f_2(\boldsymbol{\phi})$, where,*

$$f_1(\boldsymbol{\phi}) = \sum_{i=1}^{p} (-\log \phi_i + s_{ii}\phi_i) \quad \textit{and}$$

$$f_2(\boldsymbol{\phi}) = -\sum_{i=1}^{r} \left(\log(\max\{1, \lambda_i^*\}) - \max\{1, \lambda_i^*\} + 1\right).$$

Proof The convexity of $f_1(\boldsymbol{\phi})$ is easy to see. Proposition 5 implies that $f_2(\boldsymbol{\phi})$ is convex. □

When $\mathbf{S}$ is full rank i.e., $\mathbf{S} \succ \mathbf{0}$, then Problem (22) can be rewritten purely as a function of the eigenvalues $\{\lambda_i^*\}_{i\geq 1}$—this is established in Proposition 6. Such a representation does not seem to be available when $\mathbf{S}$ is rank deficient.

Proposition 6 *If $\mathbf{S} \succ \mathbf{0}$ then Problem (22) is equivalent to the following problem:*

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{p} \left(-\log \lambda_i^* + \lambda_i^*\right) + \sum_{i=1}^{r} \left(\log(\max\{1, \lambda_i^*\}) - \max\{1, \lambda_i^*\} + 1\right) \\ \text{s.t.} \quad & \mathbf{0} \prec \boldsymbol{\Phi} = \operatorname{diag}(\phi_1, \ldots, \phi_p) \preceq \tfrac{1}{\epsilon}\mathbf{I}, \end{aligned} \tag{27}$$

where, $\{\lambda_i^*\}_1^p$ are the eigenvalues of $\boldsymbol{\Phi}^{\frac{1}{2}}\mathbf{S}\boldsymbol{\Phi}^{\frac{1}{2}}$.

Proof Problem (22) is equivalent to minimizing $\bar{f}(\boldsymbol{\phi}) := -\log\det(\mathbf{S}) + f(\boldsymbol{\phi})$ over $\mathbf{0} \prec \boldsymbol{\Phi} \preceq \frac{1}{\epsilon}\mathbf{I}$. The function $\bar{f}(\boldsymbol{\phi})$ can be expressed as:

$$\begin{aligned} \bar{f}(\boldsymbol{\phi}) &= \sum_{i=1}^{p} (-\log \phi_i + s_{ii}\phi_i) - \log\det(\mathbf{S}) \\ &\quad + \sum_{i=1}^{r} \left(\log\left(\max\{1, \lambda_i^*\}\right) - \max\{1, \lambda_i^*\} + 1\right) \qquad (28) \\ &= -\log\det(\mathbf{S}^*) + \operatorname{tr}(\mathbf{S}^*) + \sum_{i=1}^{r} \left(\log\left(\max\{1, \lambda_i^*\}\right) - \max\{1, \lambda_i^*\} + 1\right) \qquad (29) \\ &= \sum_{i=1}^{p} \left(-\log \lambda_i^* + \lambda_i^*\right) + \sum_{i=1}^{r} \left(\log\left(\max\{1, \lambda_i^*\}\right) - \max\{1, \lambda_i^*\} + 1\right) \qquad (30) \end{aligned}$$

[3] A function $g(y_1, \ldots, y_p) : \Re^p \to \Re$ is said to be symmetric in its arguments if, for any permutation π of the indices $\{1, \ldots, p\}$, we have $g(y_1, \ldots, y_p) = g(y_{\pi(1)}, \ldots, y_{\pi(p)})$.

where, line (29) follows from (28) by observing that

$$\begin{aligned}\sum_{i=1}^{p}(-\log(\phi_i)+s_{ii}\phi_i)-\log\det(\mathbf{S}) &= -\log\det(\mathbf{\Phi})+\operatorname{tr}(\mathbf{S\Phi})-\log\det(\mathbf{S})\\ &= -\log\det(\mathbf{S\Phi})+\operatorname{tr}(\mathbf{S\Phi})\\ &= -\log\det(\mathbf{S}^*)+\operatorname{tr}(\mathbf{S}^*),\end{aligned} \tag{31}$$

where, $\mathbf{S}^* = \mathbf{\Phi}^{\frac{1}{2}}\mathbf{S}\mathbf{\Phi}^{\frac{1}{2}}$; and the last equality in (31) made use of Corollary 1. Moreover, as $\mathbf{S}$ is of full rank and $\mathbf{\Phi} \succ \mathbf{0}$, all the eigenvalues $\lambda_i^* > 0$, $i \in [p]$. This completes the proof. □

Proposition 6 provides an interesting alternative characterization of the formulation presented in Corollary 3—this helps us gain additional understanding of the optimization problem for ML factor analysis when $\mathbf{S} \succ \mathbf{0}$.

2.3 Algorithm based on difference of convex optimization

Problem (22) is a nonconvex optimization problem with semidefinite constraints and obtaining a global minimum for a general r is quite challenging. We thus focus on developing efficient computational procedures for obtaining good (feasible) solutions to Problem (22). By Corollary 3, Problem (22) is equivalent to the following nonconvex optimization problem:

$$\text{minimize} \quad f(\boldsymbol{\phi}) = f_1(\boldsymbol{\phi}) - f_2(\boldsymbol{\phi}) \ \text{ s.t. } \boldsymbol{\phi} \in \mathcal{C} := \left\{\boldsymbol{\phi} : \tfrac{1}{\epsilon} \geq \phi_i > 0, i \in [p]\right\}. \tag{32}$$

We use a sequential linearization procedure: at every iteration, we linearize the function $f_2(\boldsymbol{\phi})$ (leaving $f_1(\cdot)$ as is) and solve the resultant convex problem. This is an instance of the well-known framework used for optimization of difference of convex problems [18,30,37]. In the machine learning community, these methods are popularly referred to as the convex concave procedure [38,39]. These algorithms have gained significant traction in the wider optimization community due to their pervasive use in practice—some excellent recent works on this topic include [1,14,27,29] (see also references therein).

Let us formally describe the algorithm. If $\boldsymbol{\phi}^{(k)} \in \mathcal{C}$ denotes the value of $\boldsymbol{\phi}$ at the kth iteration, we linearize $f_2(\boldsymbol{\phi})$ at $\boldsymbol{\phi}^{(k)}$ with $f_1(\boldsymbol{\phi})$ unchanged; and obtain a convex approximation of $f(\boldsymbol{\phi})$, denoted by $F(\boldsymbol{\phi}; \boldsymbol{\phi}^{(k)})$:

$$f(\boldsymbol{\phi}) \approx f_1(\boldsymbol{\phi}) - \left(f_2\left(\boldsymbol{\phi}^{(k)}\right) + \left\langle \nabla_k, \boldsymbol{\phi} - \boldsymbol{\phi}^{(k)}\right\rangle\right) := F(\boldsymbol{\phi}; \boldsymbol{\phi}^{(k)}), \tag{33}$$

where, ∇_k is a subgradient of $f_2(\boldsymbol{\phi})$ at $\boldsymbol{\phi}^{(k)}$ (see Sect. 2.3.1 for details). We note that $F(\boldsymbol{\phi}; \boldsymbol{\phi}^{(k)})$ is an upper bound to $f(\boldsymbol{\phi})$ for any $\boldsymbol{\phi}^{(k)}$. We compute $\boldsymbol{\phi}^{(k+1)}$ as:

$$\boldsymbol{\phi}^{(k+1)} \in \underset{\frac{1}{\epsilon} \geq \phi_i > 0, i \in [p]}{\arg\min} \quad F(\boldsymbol{\phi}; \boldsymbol{\phi}^{(k)})$$

$$= \underset{\frac{1}{\epsilon} \geq \phi_i > 0, i \in [p]}{\arg\min} \quad \sum_{i=1}^{p} \left(-\log \phi_i + s_{ii}\phi_i - \nabla_{k,i}\phi_i \right), \tag{34}$$

where, $\nabla_{k,i}$ is the ith coordinate of $\boldsymbol{\nabla}_k \in \Re^p$. The ith entry of $\boldsymbol{\phi}^{(k+1)}$ is given by:

$$\phi_i^{(k+1)} = \min\left\{ \frac{1}{s_{ii} - \nabla_{k,i}}, \frac{1}{\epsilon} \right\} \quad \text{for} \quad i \in [p].$$

The updates continue till some stopping criterion is satisfied. This can be in terms of the relative change in the successive objective values:[4] $f(\boldsymbol{\phi}^{(k)}) - f(\boldsymbol{\phi}^{(k+1)}) < \eta |f(\boldsymbol{\phi}^{(k+1)})|$ or relative change in successive iterate values: $\|\boldsymbol{\phi}^{(k+1)} - \boldsymbol{\phi}^{(k)}\|_2 < \eta \|\boldsymbol{\phi}^{(k)}\|_2$; where, $\eta > 0$ denotes a pre-specified tolerance level. We summarize the algorithm below.

Algorithm 1: An algorithm for Problem (32).
Initialize with $\boldsymbol{\phi}^{(1)} \in \mathcal{C}$ and update $\boldsymbol{\phi}^{(k)}$ using (34) until some stopping criterion like $f(\boldsymbol{\phi}^{(k)}) - f(\boldsymbol{\phi}^{(k+1)}) < \eta |f(\boldsymbol{\phi}^{(k+1)})|$ is met.

2.3.1 Computing subgradients

Here, we study the computation of (sub)gradients [32] of the functions $f_1(\boldsymbol{\phi})$ and $f_2(\boldsymbol{\phi})$. Note that $f_1(\boldsymbol{\phi})$ is differentiable, hence its subgradient is the same as its gradient. However, the convex spectral function[5] $f_2(\boldsymbol{\phi})$ is not differentiable. A subgradient of $f_2(\boldsymbol{\phi})$ can be computed following the work of [24] on differentiability of spectral functions of Hermitian matrices. To this end, consider the representation of $H_r(\mathbf{y})$ in (24) and define the function $g(y) = \log(\max\{1, y\}) - \max\{1, y\} + 1$ on $y \geq 0$. Let us denote $\widetilde{H}_r(\mathbf{y}) = -H_r(\mathbf{y})$ and note that $\widetilde{H}_r(\mathbf{y})$ is a convex function in $\mathbf{y}$. If $\partial \widetilde{H}_r(\mathbf{y})$ is a subgradient of $\widetilde{H}_r(\mathbf{y})$, then it can be computed by using Danskin's theorem:

$$\partial \widetilde{H}_r(\mathbf{y}) = -\sum_{i=1}^{p} \hat{w}_i \nabla g(y_i),$$

where, $\hat{\mathbf{w}}$ is a minimizer of the inner optimization task in Problem (24); and $\nabla g(y_i) \in \Re^p$ is the gradient of $g(y_i)$, with ith coordinate given by $\nabla_i g(y_i) = \min\{0, \frac{1}{y_i} - 1\}$, and $\nabla_j g(y_i) = 0$ for all $j \neq i$. The function $\widetilde{H}_r(\mathbf{y})$ is differentiable at $\mathbf{y}$ iff $\hat{\mathbf{w}}$ is unique.[6] The set of all subgradients of $\widetilde{H}_r(\mathbf{y})$ is given by

$$\text{Conv}\left(\left\{ -\sum_{i=1}^{p} \hat{w}_i \nabla g(y_i) \ : \ \hat{\mathbf{w}} \text{ is a minimizer of Problem (24), i.e., } H_r(\mathbf{y}) = \widetilde{H}(\hat{\mathbf{w}}; \mathbf{y}) \right\} \right).$$

[4] We note that the objective values are decreasing $f(\boldsymbol{\phi}^{(k+1)}) \leq f(\boldsymbol{\phi}^{(k)})$ for all k (See Proposition 7).

[5] We call $f_2(\boldsymbol{\phi})$ a spectral function as it is a function of the eigenvalues (or spectral values) $\{\lambda_i^*\}_1^p$.

[6] We note that *non-differentiability* occurs if $g(y_{(r+1)}) = g(y_{(r)})$.

Let us consider a matrix $\mathbf{A} \succeq \mathbf{0}$, with eigen decomposition $\mathbf{A} = \mathbf{V}\text{diag}(\boldsymbol{\lambda})\mathbf{V}^\top$; and consequently consider the spectral convex function $\tilde{g}_r(\mathbf{A}) := -\sum_{i=1}^r g(\lambda_i)$. Using properties of subgradients of spectral functions [24], we have that a subgradient of $\mathbf{A} \mapsto \tilde{g}_r(\mathbf{A})$ is given by:

$$\partial \tilde{g}_r(\mathbf{A}) = \mathbf{V}\text{diag}(\partial \widetilde{H}_r(\boldsymbol{\lambda}))\mathbf{V}^\top \tag{35}$$

where, $\partial \widetilde{H}_r(\boldsymbol{\lambda})$ is a subgradient of $\boldsymbol{\lambda} \mapsto \widetilde{H}_r(\boldsymbol{\lambda})$.

Let $\boldsymbol{\Phi}^{\frac{1}{2}}\mathbf{S}\boldsymbol{\Phi}^{\frac{1}{2}} = \mathbf{U}\text{diag}(\lambda_1^*, ..., \lambda_p^*)\mathbf{U}^\top$ be the eigen decomposition of $\boldsymbol{\Phi}^{\frac{1}{2}}\mathbf{S}\boldsymbol{\Phi}^{\frac{1}{2}}$. By the chain rule, $\partial f_2(\boldsymbol{\phi})$ is given by

$$\partial f_2(\boldsymbol{\phi}) = \text{diag}\left(\boldsymbol{\Phi}^{-\frac{1}{2}}\mathbf{U}\mathbf{D}_1\mathbf{U}^\top\boldsymbol{\Phi}^{\frac{1}{2}}\mathbf{S}\right), \tag{36}$$

where, $\mathbf{D}_1 = \text{diag}\left(\delta_1, ..., \delta_p\right)$ with

$$\delta_i = \begin{cases} \max\left\{0, 1 - \frac{1}{\lambda_i^*}\right\} & \text{if } 1 \le i \le r \\ 0 & \text{otherwise.} \end{cases} \tag{37}$$

2.4 Computational guarantees for Algorithm 1

We present herein, computational guarantees for Algorithm 1 in terms of: the number of iterations required to deliver an approximate first order stationary point and asymptotic convergence to a first order stationary point. Towards this end, we recall certain standard definitions of first order stationary conditions for Problem (32) (see for example, [30]). We say that $\tilde{\boldsymbol{\phi}} \in \mathcal{C}$ is a first order stationary point of Problem (32), if the following condition holds:

$$\begin{aligned} \tilde{\boldsymbol{\phi}} \in \arg\min_{\boldsymbol{\phi}} \quad & F(\boldsymbol{\phi}; \tilde{\boldsymbol{\phi}}) = f_1(\boldsymbol{\phi}) - \langle \partial f_2(\tilde{\boldsymbol{\phi}}), \boldsymbol{\phi} - \tilde{\boldsymbol{\phi}} \rangle \\ \text{s.t.} \quad & \boldsymbol{\phi} \in \mathcal{C} = \left\{\boldsymbol{\phi} : \tfrac{1}{\epsilon}\mathbf{1} \ge \boldsymbol{\phi} > \mathbf{0}\right\}, \end{aligned} \tag{38}$$

for some choice of a subgradient $\partial f_2(\tilde{\boldsymbol{\phi}})$. From standard optimality conditions of convex functions [10,32], the above condition is equivalent to saying that

$$\partial f_2(\tilde{\boldsymbol{\phi}}) \in \nabla f_1(\tilde{\boldsymbol{\phi}}) + \mathcal{N}(\tilde{\boldsymbol{\phi}}; \mathcal{C}), \tag{39}$$

where, $\mathcal{N}(\tilde{\boldsymbol{\phi}}; \mathcal{C})$ is the normal cone to the convex set $\mathcal{C}$ at the point $\tilde{\boldsymbol{\phi}}$. Recall that $\mathcal{N}(\tilde{\boldsymbol{\phi}}; \mathcal{C})$ is the convex cone of all vectors $\mathbf{d} \in \Re^p$ such that $\langle \mathbf{d}, \boldsymbol{\phi} - \tilde{\boldsymbol{\phi}} \rangle \le 0$ for all $\boldsymbol{\phi} \in \mathcal{C}$. In (39), the right hand side denotes the standard Minkwoski sum of a vector $(\nabla f_1(\tilde{\boldsymbol{\phi}}))$ and a set $(\mathcal{N}(\tilde{\boldsymbol{\phi}}; \mathcal{C}))$.

Proposition 7 shows that the sequence $\{\boldsymbol{\phi}^{(k)}\}_k$ leads to a decreasing sequence of objective values where, the amount of decrease is lower bounded by the squared norm of successive difference of the iterates $\{\boldsymbol{\phi}^{(k)}\}$.

Proposition 7 *Let $\boldsymbol{\phi}^{(k)}$ be a sequence generated via Algorithm 1. Then, there exists $\rho \geq \epsilon^2$ such that for every $k \geq 1$:*

$$f\left(\boldsymbol{\phi}^{(k)}\right) - f\left(\boldsymbol{\phi}^{(k+1)}\right) \geq \frac{\rho}{2}\|\boldsymbol{\phi}^{(k+1)} - \boldsymbol{\phi}^{(k)}\|^2. \tag{40}$$

Proof From convexity of $f_2(\boldsymbol{\phi})$ we have that

$$f_2\left(\boldsymbol{\phi}^{(k+1)}\right) \geq f_2\left(\boldsymbol{\phi}^{(k)}\right) + \left\langle \partial f_2\left(\boldsymbol{\phi}^{(k)}\right), \boldsymbol{\phi}^{(k+1)} - \boldsymbol{\phi}^{(k)} \right\rangle \tag{41}$$

where, $\partial f_2(\boldsymbol{\phi})$ is a subgradient of $f_2(\boldsymbol{\phi})$. Note that the function $f_1(\boldsymbol{\phi})$ is separable across the coordinates and $\nabla^2 f_1(\boldsymbol{\phi}) \succeq \epsilon^2 \mathbf{I}$ for all $\boldsymbol{\phi} \in (0, \frac{1}{\epsilon}]^p$. If we denote ρ $(\geq \epsilon^2)$ to be a coefficient of strong convexity for the function $f_1(\boldsymbol{\phi})$ on $(0, \frac{1}{\epsilon}]^p$ then:

$$f_1\left(\boldsymbol{\phi}^{(k)}\right) \geq f_1\left(\boldsymbol{\phi}^{(k+1)}\right) + \left\langle \boldsymbol{\phi}^{(k)} - \boldsymbol{\phi}^{(k+1)}, \nabla f_1\left(\boldsymbol{\phi}^{(k+1)}\right) \right\rangle + \frac{\rho}{2}\|\boldsymbol{\phi}^{(k+1)} - \boldsymbol{\phi}^{(k)}\|^2, \tag{42}$$

where, $\nabla f_1(\boldsymbol{\phi})$ is the derivative of $f_1(\boldsymbol{\phi})$. By standard optimality conditions [10] of Problem (34), we have:

$$\min\left\{ \left\langle \nabla F(\boldsymbol{\phi}^{(k+1)}; \boldsymbol{\phi}^{(k)}), \boldsymbol{\phi} - \boldsymbol{\phi}^{(k+1)} \right\rangle : \tfrac{1}{\epsilon}\mathbf{1} \geq \boldsymbol{\phi} > \mathbf{0} \right\} \geq 0, \tag{43}$$

where, $\nabla F(\boldsymbol{\phi}^{(k+1)}; \boldsymbol{\phi}^{(k)}) = \nabla f_1(\boldsymbol{\phi}^{(k+1)}) - \partial f_2(\boldsymbol{\phi}^{(k)})$ is the derivative of $\boldsymbol{\phi} \mapsto F(\boldsymbol{\phi}; \boldsymbol{\phi}^{(k)})$ evaluated at $\boldsymbol{\phi}^{(k+1)}$. Adding (42) and (41), and rearranging terms we get:

$$\begin{aligned} f_1\left(\boldsymbol{\phi}^{(k+1)}\right) - f_2\left(\boldsymbol{\phi}^{(k+1)}\right) &\leq f_1\left(\boldsymbol{\phi}^{(k)}\right) - f_2\left(\boldsymbol{\phi}^{(k)}\right) - \frac{\rho}{2}\|\boldsymbol{\phi}^{(k+1)} - \boldsymbol{\phi}^{(k)}\|^2 \\ &\quad + \underbrace{\left\langle \boldsymbol{\phi}^{(k+1)} - \boldsymbol{\phi}^{(k)}, \nabla f_1\left(\boldsymbol{\phi}^{(k+1)}\right) - \partial f_2\left(\boldsymbol{\phi}^{(k)}\right) \right\rangle}_{\leq 0} \\ &\leq f_1\left(\boldsymbol{\phi}^{(k)}\right) - f_2\left(\boldsymbol{\phi}^{(k)}\right) - \frac{\rho}{2}\|\boldsymbol{\phi}^{(k+1)} - \boldsymbol{\phi}^{(k)}\|^2. \end{aligned}$$

Here, the last line follows by setting $\phi = \phi^{(k)}$ in (43). □

The above proposition says that $f(\boldsymbol{\phi}^{(k)})$ is a decreasing sequence—being bounded below, it converges to $\hat{f}$, say. By the definition of a first order stationary point (38), the quantity $\|\boldsymbol{\phi}^{(k+1)} - \boldsymbol{\phi}^{(k)}\|$ dictates the proximity of $\boldsymbol{\phi}^{(k)}$ to a first order stationary point and an approximate first order stationary point. We have the following proposition, formalizing the rate at which the sequence $\boldsymbol{\phi}^{(k)}$ approaches a first order stationary point.

Proposition 8 *The sequence $f(\boldsymbol{\phi}^{(k)})$ is decreasing and converges to $\hat{f}$. The finite time convergence rate is given by:*

$$\min_{1 \leq k \leq \mathcal{K}} \rho\|\boldsymbol{\phi}^{(k+1)} - \boldsymbol{\phi}^{(k)}\|^2 \leq \frac{2}{\mathcal{K}}\left(f(\boldsymbol{\phi}^{(1)}) - \hat{f}\right). \tag{44}$$

Proof The proof uses (40). If $\Delta_k := \frac{\rho}{2}\|\boldsymbol{\phi}^{(k+1)} - \boldsymbol{\phi}^{(k)}\|^2$ then:

$$\mathcal{K} \min_{1\le k\le \mathcal{K}} \Delta_k \le \sum_{i=1}^{\mathcal{K}} \Delta_k \le \sum_{i=1}^{\mathcal{K}} \left\{ f\left(\boldsymbol{\phi}^{(k)}\right) - f\left(\boldsymbol{\phi}^{(k+1)}\right)\right\} \le \quad f\left(\boldsymbol{\phi}^{(1)}\right) - \hat{f}, \quad (45)$$

where, the second inequality uses (40); and the final inequality used the fact that $f(\boldsymbol{\phi}^{(k)}) \downarrow \hat{f}$. The result (44) follows from combining the left and right parts of the inequality (45). □

The above proposition states that for any tolerance $\delta > 0$, there is an integer $\mathcal{K} = O(\frac{1}{\delta})$ such that for some $k \in [\mathcal{K}]$, the following holds: $\|\boldsymbol{\phi}^{(k+1)} - \boldsymbol{\phi}^{(k)}\|^2 \le \delta$. Since the optimization problem is nonconvex, $\hat{f}$ may depend upon the initialization $\boldsymbol{\phi}^{(1)}$. The following proposition shows that all limit points of the sequence $\{\boldsymbol{\phi}^{(k)}\}_k$ are first order stationary points.

Proposition 9 *Any limit point of the sequence $\boldsymbol{\phi}^{(k)}$ is a first order stationary point for Problem (32).*

Proof The proof is deferred to Appendix A.1. □

2.5 Computational cost of Algorithm 1

We discuss the computational cost of Algorithm 1 and techniques for computational scalability to large problems. We consider different cases, depending upon the relative sizes of n, p.

When $n > p$: The main computational cost of Algorithm 1 stems from computing a subgradient of $f_2(\boldsymbol{\phi})$ which requires a low-rank eigen decomposition of $\mathbf{S}^*$. When p is small relative to n, it is convenient to form and work with the $p \times p$ matrix $\mathbf{S}^*$. Creating $\mathbf{S}$ from $\mathbf{X}$ costs $O(n^2 p)$ (this operation can be done once offline). Computing $\mathbf{S}^*$ from $\mathbf{S}$ costs $O(p^2)$. A direct low-rank eigen-decomposition of $\mathbf{S}^*$ using dense matrix factorization methods [16] costs $O(p^3)$. This approach applies to problems with p up to a few thousand ($p \approx 3000$, for example). Note that this cost is of the same order as obtaining the unrestricted maximum likelihood estimate of $\boldsymbol{\Sigma}^{-1}$, which is given by $\mathbf{S}^{-1}$ (assuming $\mathbf{S}$ is invertible).

When $p \gg n$: In several applications of interest, n is smaller than p—a situation that occurs commonly in the modern highdimensional regime (in genomics applications for example, n is often a few hundred and p is in the order of tens of thousands). In such cases, obtaining a rank r eigen decomposition of $\mathbf{S}^*$ will cost $O(n^2 p)$, which is linear in p if $n \ll p$. This follows by observing that $\mathbf{S} = \frac{1}{n}\mathbf{X}^\top\mathbf{X}$ (where, $\mathbf{X}$ is mean-centered) and an eigen decomposition of $\mathbf{S}^*$, i.e., $\boldsymbol{\Phi}^{\frac{1}{2}}\mathbf{S}\boldsymbol{\Phi}^{\frac{1}{2}} = (\frac{1}{\sqrt{n}}\mathbf{X}\boldsymbol{\Phi}^{\frac{1}{2}})^\top(\frac{1}{\sqrt{n}}\mathbf{X}\boldsymbol{\Phi}^{\frac{1}{2}})$ can be obtained via a SVD of the $n \times p$ matrix $\frac{1}{\sqrt{n}}\mathbf{X}\boldsymbol{\Phi}^{\frac{1}{2}}$—this SVD costs $O(n^2 p)$.

In addition, there are certain costs associated with matrix multiplications. Indeed, a careful book-keeping allows us to operate with matrices that are of low-rank—we never need to create or form a dense $p \times p$ matrix. This is beneficial from a memory

standpoint and enables us to scale up the computations to instances where, p is of the order of tens of thousands (as long as n is sufficiently small). First of all, note that the computation of $\mathbf{X}\boldsymbol{\Phi}^{\frac{1}{2}}$ costs $O(np)$. In addition, one needs to compute the diagonal entries of $T := \boldsymbol{\Phi}^{-\frac{1}{2}}\mathbf{U}\mathbf{D}_1\mathbf{U}^\top\boldsymbol{\Phi}^{\frac{1}{2}}\mathbf{S}$ as in (36). Note that T is a $p \times p$ matrix; however, its diagonal entries can be computed without explicitly creating the matrix T. This follows by observing that T can be written as the product of two low-rank matrices: $T = T_1T_2$, where, $T_1 = \boldsymbol{\Phi}^{-\frac{1}{2}}\mathbf{U}\mathbf{D}_1$ and $T_2 = \mathbf{U}^\top\boldsymbol{\Phi}^{\frac{1}{2}}\mathbf{S}$. Note that: $T_1 \in \Re^{p\times r}$, $T_2 \in \Re^{r\times p}$ and $r < n$. Hence, one can compute the diagonal entries of T with a cost $O(pr)$. It is important to note that T_2 has to be computed with care, since we do not want to create/store the matrix $\mathbf{S}$—to this end, observe that $T_2 = T_{21}\mathbf{X}$ with $T_{21} = \frac{1}{n}\mathbf{U}^\top(\boldsymbol{\Phi}^{\frac{1}{2}}\mathbf{X}^\top) \in \Re^{r\times n}$ and $\mathbf{X} \in \Re^{n\times p}$—we can thus compute T_2 without forming $\mathbf{S}$.

When both p, n are large: When both n, p are large, the direct SVD factorization methods outlined above, will become computationally expensive. We will need to resort to approximate schemes for large scale low-rank SVD decompositions. Approximate rank r eigen decompositions can be computed using techniques in [12]; or methods based on the Lanczos method [20] or block power iterations [16]. This will cost $O(p^2r)$, which can be significantly smaller than $O(p^3)$ for $r \ll p \approx n$.

2.6 Obtaining solutions to Problem (2) when $\epsilon \approx 0$

The conventional version of the ML factor model optimization problem is given by:

$$\begin{aligned} \text{minimize} \quad & -\log\det(\boldsymbol{\Sigma}^{-1}) + \operatorname{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}) \\ \text{s.t.} \quad & \boldsymbol{\Sigma} = \boldsymbol{\Psi} + \mathbf{L}\mathbf{L}^\top, \boldsymbol{\Psi} = \operatorname{diag}(\psi_1, \ldots, \psi_p) \succ \mathbf{0}, \end{aligned} \tag{46}$$

which may be interpreted as a limiting version of Problem (2) with $\epsilon \to 0+$. We note that there are technical difficulties with Problem (46) as it may be unbounded from below and hence a ML estimator need not exist. [31] discusses necessary and sufficient conditions for boundedness of Problem (46). Problem (46) is bounded below under the following conditions: (a) If $\mathbf{S}$ has full rank or (b) If $\mathbf{S}$ is rank deficient, then $r < s - 1$ where, s denotes the number of nonzero coordinates in the sparsest nonzero vector (i.e., a nonzero vector with the maximum number of coordinates that are zero) in the null space of $\mathbf{S}$. Unfortunately, computing the sparsest nonzero vector in a subspace is a combinatorially difficult problem. However, note that if the sample covariance matrix corresponds to that of a continuous random variable, then Problem (46) is bounded below with probability one. Even if Problem (46) is bounded below, the minimum may not be attained—the infimum may correspond to some coordinates of $\boldsymbol{\psi}$ being set to zero. These are known as Heywood cases [31]—they are infamously known to create numerical difficulty from a computational viewpoint and may also lead to misleading statistical inference.

If $\hat{\boldsymbol{\Psi}}_\epsilon$ is a solution to Problem (2), then a limiting value of $\hat{\boldsymbol{\Psi}}_\epsilon$ as $\epsilon \downarrow 0+$ will give a solution to Problem (46) (provided a limit exists). This inspires a simple continuation scheme: we consider a sequence of ϵ-values that converge to a small number ϵ' (e.g.,

$\epsilon \downarrow \epsilon'$ with $\epsilon' := 10^{-8}$, say); and use Algorithm 1 with warm-start continuation. Suppose, there is a subset of indices $\mathbf{M} \subset \{1, \ldots, p\}$ such that, $\hat{\psi}_i = \epsilon'$ for all $i \in \mathbf{M}$, and $\hat{\psi}_i > \epsilon'$ for all $i \notin \mathbf{M}$. We can obtain a good solution to Problem (46) by fixing $\psi_i = \epsilon'$ for all $i \in \mathbf{M}$ and optimizing over the remaining $\psi_i, i \notin \mathbf{M}$ values. This can be performed by a simple modification to Algorithm 1: in (34), we update *only* the ϕ_i values (recall that $\phi_i = \psi_i^{-1}$ for all i) corresponding to $i \notin \mathbf{M}$ and set the remaining ϕ_i values to $1/\epsilon'$.

2.7 Ridge regularization

Instead of considering a direct lower bound on ψ_i as in Problem (2), we can also consider a ridge regularized version of Problem (46) which leads to the following variant of Problem (22):

$$\text{minimize} \quad f(\boldsymbol{\phi}) + \sum_{i=1}^{p} \gamma \phi_i^2 \quad \text{s.t.} \quad \boldsymbol{\phi} = \text{diag}(\phi_1, \ldots, \phi_p) \succ \mathbf{0}, \tag{47}$$

for some regularization parameter $\gamma > 0$. Algorithm 1 can be adapted to Problem (47) by changing update (34) as:

$$\boldsymbol{\phi}^{(k+1)} = \underset{\boldsymbol{\phi} > \mathbf{0}}{\arg\min} \sum_{i \in [p]} \left\{ -\log \phi_i + s_{ii}\phi_i + \gamma \phi_i^2 - \nabla_{k,i}\phi_i \right\}, \tag{48}$$

where, $\nabla_{k,i}$'s can be computed as in Sect. 2.3.1. The ith coordinate of $\boldsymbol{\phi}^{(k+1)}$ is given by:

$$\phi_i^{(k+1)} = \frac{1}{4\gamma} \left(\nabla_{k,i} - s_{ii} + \sqrt{(s_{ii} - \nabla_{k,i})^2 + 8\gamma} \right), \quad i \in [p]. \tag{49}$$

Note that $s_{ii} - \nabla_{k,i}$ is the ith diagonal entry of the matrix $\mathbf{S} - \boldsymbol{\Phi}^{-\frac{1}{2}} \mathbf{U} \mathbf{D}_1 \mathbf{U}^\top \boldsymbol{\Phi}^{\frac{1}{2}} \mathbf{S}$, which can be rearranged as:

$$\boldsymbol{\Phi}^{-\frac{1}{2}} \left(\boldsymbol{\Phi}^{\frac{1}{2}} \mathbf{S} \boldsymbol{\Phi}^{\frac{1}{2}} - \mathbf{U} \mathbf{D}_1 \mathbf{U}^\top \boldsymbol{\Phi}^{\frac{1}{2}} \mathbf{S} \boldsymbol{\Phi}^{\frac{1}{2}} \right) \boldsymbol{\Phi}^{-\frac{1}{2}} = \boldsymbol{\Phi}^{-\frac{1}{2}} \left(\mathbf{U} \widetilde{\mathbf{D}} \mathbf{U}^\top \right) \boldsymbol{\Phi}^{-\frac{1}{2}} \tag{50}$$

where, $\widetilde{\mathbf{D}}$ is a diagonal matrix with diagonal entries given by

$$\widetilde{d}_{ii} = \begin{cases} 1 & \text{if } 1 \le i \le r \text{ and } \lambda_i^* \ge 1 \\ \lambda_i^* & \text{otherwise.} \end{cases}$$

This implies that the matrix in (50) is positive semidefinite; and in particular, the diagonal entries are nonnegative: $s_{ii} - \nabla_{k,i} \ge 0$ for all $i \in [p]$. This implies from (49) that: $\psi_i^{(k+1)} = \frac{1}{\phi_i^{(k+1)}} \ge \sqrt{2\gamma}$. Hence ridge regularization keeps the estimated ψ_i values bounded away from zero. A continuation scheme similar to that described in Sect. 2.6 can be used with $\gamma \downarrow \gamma'$ (with $\gamma' = 10^{-8}$, say) to get a good solution to Problem (46).

3 Computational experiments

We present computational experiments demonstrating the performance of Algorithm 1, which we name FACTMLE. We compare its performance versus other popular approaches for ML factor analysis on synthetic and real data examples. All computations were done in Matlab on a standard Mac desktop with 32 GB RAM.

3.1 Comparison across different methods

Competing methods: We compared our proposed method: Algorithm 1 (FACTMLE) with the following leading algorithms for ML factor analysis:

1. `Factoran`: This is the widely used, native implementation of ML factor analysis in Matlab; and this code is based on the seminal work of [19].
2. `Emfact`: This is a popular EM algorithm based method presented in [4].
3. `Fa`: This is an EM algorithm based technique for factor analysis based on [9].[7]

Of the above three methods, `Factoran` and `Emfact` apply only when $n > p$—thus we restrict our attention to `Fa` as the only competitor of FACTMLE for examples where $n < p$. We do not include the method of [8] in our comparisons, since it optimizes a different criterion (not the maximum likelihood objective). In terms of scalability considerations, the method of [8] is less scalable (since it requires performing the eigen-decomposition of an unstructured $p \times p$ matrix) than FACTMLE. The standard implementation of `Factoran` uses correlation matrices; hence, we perform a post-processing of the results obtained from `Factoran` to facilitate comparison with other algorithms.

Synthetic data generation: We generated (the true) $\mathbf{L}^0 \in \Re^{p \times r_0}$ (with $r_0 \ll p$) with entries drawn iid from $N(\mu, \sigma^2)$. For examples with $n > p$, shown in Fig. 1, we set $\mu = 10$ and $\sigma^2 = 1$. The (true) unique variances $\mathbf{\Psi}^0 = \text{diag}(\psi_1^0, \psi_2^0, \ldots, \psi_p^0)$ were generated independently from an exponential distribution with mean 10. Once $\mathbf{\Psi}^0$ and $\mathbf{L}^0$ were generated, we created $\mathbf{\Sigma}^0 = \mathbf{\Psi}^0 + \mathbf{L}^0(\mathbf{L}^0)^\top$. We generated $\mathbf{X}_{n \times p}$ as n independent samples from a multivariate Gaussian distribution with mean $\mathbf{0}$ and covariance $\mathbf{\Sigma}^0$. We performed ten replications of the experiments.

Performance measures: All algorithms are compared in terms of the quality of solution obtained (i.e., the objective value): we consider the negative log-likelihood $\mathcal{L}(\hat{\mathbf{\Sigma}}) := -\log\det(\hat{\mathbf{\Sigma}}^{-1}) + \text{tr}(\hat{\mathbf{\Sigma}}^{-1}\mathbf{S})$ where, $\hat{\mathbf{\Sigma}} = \hat{\mathbf{\Psi}} + \hat{\mathbf{L}}\hat{\mathbf{L}}^\top$, assuming that the estimates are feasible.

In addition, we consider the run-times of the different algorithms. This may depend upon the different convergence criteria employed by the different algorithms (see below); and the quality of solution obtained. We note that the quality of solutions (in terms of objective value) can be different across the algorithms since the optimization problem is nonconvex. Thus the run-time of an algorithm is interpreted along with the quality of the available solution.

[7] This function is available as a part of Matlab's PRML toolbox https://www.mathworks.com/matlabcentral/fileexchange/55883-probabilistic-pca-and-factor-analysis?focused=6047050&tab=function.

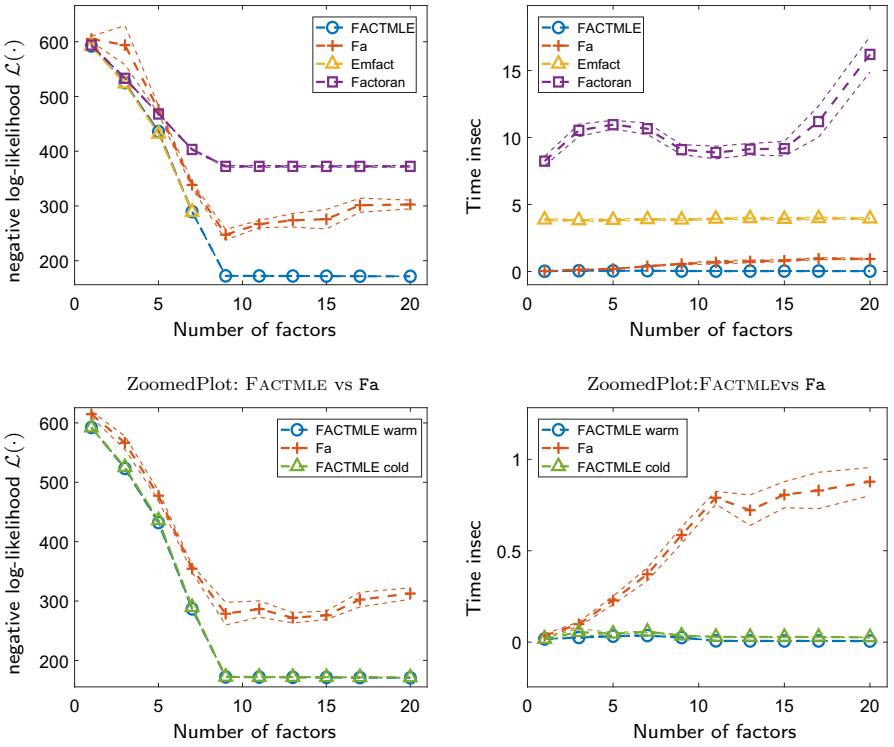

Fig. 1 Figure showing performances of different ML methods for factor analysis for synthetic data with $(n, p) = (2200, 200)$. [Top panel] we compare the performance of `Fa`, FACTMLE, `Factoran` and `Emfact` in terms of negative log-likelihood and run-time. We see that our method FACTMLE outperforms all the other competing methods, both in terms of run-time and negative log-likelihood value (for different values of r shown along the horizontal axes). In some cases, `Emfact` produces negative estimates of $\mathbf{\Phi}$—in those cases, we have not plotted the result obtained from `Emfact`. [Bottom panel] shows a zoomed-in version of `Fa` (the best competitor of FACTMLE) versus FACTMLE with warm-start continuation across different r values (denoted by "warm") and without warm-start (this is denoted by "cold'). We see that the performance of FACTMLE (with or without warm-start) is better than that of `Fa`. FACTMLE timings are found to improve marginally in the presence of warm-starts. Results are averaged over ten replications and the bands represent point-wise one standard error bars (the error bars for FACTMLE are quite narrow, and seem to overlap with the average line)

Finally, we note that Algorithm 1 can readily incorporate warm-starts—they may be useful if one desires a sequence of solutions to Problem (2) for different values of $r = 1, 2, \ldots$. Other algorithms like: `Fa`, `Factoran` and `Emfact` do not allow for warm-start specification—so we used their default initialization strategy.

Summary of comparisons and observations: We first consider a synthetic dataset with $n > p$. Figure 1 shows the performances of different methods for synthetic data generated as above, with $p = 200$, $n = 2200$, $r_0 = 8$, and different choices of r.

For FACTMLE, the tolerance level η (for convergence based on objective value difference, as explained in Sect. 2.3) was set to 10^{-4}; we set the maximum number of iterations to be 1000. For all other algorithms (`Fa`, `Factoran` and `Emfact`)

we choose their default convergence criteria with maximum number of iterations set to 1000. Figure 1 suggests that the performance of FACTMLE, measured in terms of the objective value, is significantly better than all the other algorithms—thereby suggesting that it does well in the task it was set to accomplish. The performance of `Emfact`, in terms of negative log-likelihood, is comparable to FACTMLE when r is small. However, `Emfact` often encounters numerical difficulties (especially when r is large) and produces negative estimates of $\boldsymbol{\Phi}$—this violates the condition $\boldsymbol{\Phi} \succ \mathbf{0}$ and is highly undesirable. This suggests that one should be cautious while using `Emfact` in practice. An attractive feature of FACTMLE is the timings. The number of iterations required by `Fa`, `Emfact` and `Factoran` (to converge) is usually much larger than that of FACTMLE. For example, for $p = 200$, $r = 6$ the (average) number of iterations for `Fa`, `Emfact`,`Factoran` were around 1000 (i.e., the maximal iteration limit), but for FACTMLE it was less than twenty. Note that in addition to a slow convergence speed, the competing algorithms seem to get stuck in suboptimal local solutions. We observe that the differences among FACTMLE and its competitors are more pronounced for larger values of r. In terms of timings, FACTMLE is a clear winner. The performance `Fa` seems to be better than `Factoran` and `Emfact`; and seems to be the only competitor to FACTMLE.

We compare the performance of `Fa` with FACTMLE for two different types of initializations: with warm start and with random initialization—see Fig. 1 (bottom panel). We observe that for both types of initializations, FACTMLE outperforms `Fa` in terms of the quality of solution obtained and also run-times. The timings of FACTMLE with warm start is found to be slightly better than FACTMLE with cold start (as expected). In addition to the examples reported here, we also took some other values of (n, p, r); but the results were found to be quite similar—hence we do not report them here.

3.1.1 Further comparisons with `Fa`

We systematically observed that among `Fa`, `Factoran` and `Emfact`; `Fa` emerged as a winner (see also Fig. 1) in terms of numerical stability, quality of solutions delivered and run-times. We also note that `Fa` is the only method among the three which applies for both the cases: $n > p$ and $n \leq p$. Hence, we perform a more detailed comparison between `Fa` and our proposed method: FACTMLE. We consider six additional datasets (see Table 1) in our experiments: three synthetic and three real—this includes both the cases $n > p$ and $p > n$. For all the numerical examples in this section, we used the same convergence criterion for both `Fa` and FACTMLE. We ran both `Fa` and FACTMLE for a maximum of 2000 iterations and tolerance threshold of $\eta = 10^{-8}$ (in terms of relative change in successive objective values). Let $f_k(\texttt{alg})$ denote the objective value (negative log-likelihood) for method $\texttt{alg} \in \{\texttt{Fa}, \text{FACTMLE}\}$ at iteration k. We obtain the best objective value across all the methods and set it to f_*. We then study the first time at which an algorithm "`alg`" reaches a tolerance level of "Tol": $(f_k(\texttt{alg}) - f_*)/|f_*| \leq \text{Tol}$.

We consider the following datasets in our experiments.

Example a (Simulated Data), $(p > n)$: We consider 3 synthetic datasets where, the number of covariates p is larger than the number of samples n. We took: $(n, p) =$

Table 1 Performance of FACTMLE and Fa for different real and synthetic datasets as described in the text

Tol	Time (in s) Fa	Time (in s) FACTMLE	Tol	Time (in s) Fa	Time (in s) FACTMLE
Synthetic data ($n = 500$, $p = 5000$)			Synthetic Data ($n = 150$, $p = 10^4$)		
10^{-2}	>*83.142 (4.174)	4.497 (0.041)	10^{-2}	>*48.489 (4.605)	1.680 (0.027)
10^{-3}	>*107.713 (2.148)	7.702 (2.508)	10^{-3}	>*90.445 (1.601)	1.920 (0.048)
10^{-4}	>*112.145 (1.157)	18.490 (5.706)	10^{-4}	>*93.004 (0.774)	4.393 (0.592)
10^{-5}	>*112.146 (1.157)	29.369 (6.154)	10^{-5}	>*93.004 (0.773)	6.662 (0.764)
Synthetic data ($n = 50$, $p = 10^4$)			Phoneme Data ($n = 4509$, $p = 256$)		
10^{-2}	>*43.794 (1.160)	0.558 (0.097)	10^{-2}	3.038 (0.385)	0.125 (0.006)
10^{-3}	>*50.798 (0.712)	0.882 (0.228)	10^{-3}	>*23.902 (1.624)	0.187 (0.007)
10^{-4}	>*52.669 (0.362)	1.901 (0.375)	10^{-4}	>*29.635 (1.430)	0.433 (0.008)
10^{-5}	>*52.959 (0.296)	3.798 (0.496)	10^{-5}	>*37.035 (1.245)	0.679 (0.008)
ZipCode data ($n = 1858$, $p = 249$)			Cancer data ($n = 144$, $p = 16063$)		
10^{-2}	1.558 (0.060)	0.111 (0.007)	10^{-2}	1.055 (0.016)	3.816 (0.014)
10^{-3}	5.822 (0.099)	0.171 (0.008)	10^{-3}	12.021 (0.099)	5.346 (0.040)
10^{-4}	>*8.673 (0.100)	0.445 (0.010)	10^{-4}	>*81.979 (0.355)	8.516 (0.050)
10^{-5}	>*10.803 (0.099)	0.648 (0.012)	10^{-5}	>*94.586 (0.357)	12.431 (0.060)

We show the times (s) taken by different algorithms to compute the entire path of solutions for different values of r, as specified in the text. In all the above examples, a symbol ">*" means that the corresponding algorithm did not converge to the specified tolerance level for multiple values of r (for multiple replications). This may be due to the algorithm being stuck in a suboptimal local solution (compared to a better solution obtained by FACTMLE) and/or due to slow convergence behavior. For additional details (e.g., the choice of problem parameters) see the main text. FACTMLE is seen to be a clear winner across all instances. The results are averaged over 10 replications with standard errors in parenthesis

$(500, 5000)$, $(n, p) = (50, 10^4)$ and $(n, p) = (150, 10^4)$. The data was simulated as per the setup mentioned in Sect. 3.1, with the entries ϕ_i^0 drawn iid from an exponential distribution with mean 1. We set $r_0 = 5$. We considered a sequence of solutions for the ML factor analysis problem for 15 equi-spaced values of $r \in [1, 18]$.

Example b (Phoneme Data), $(n > p)$: The data were extracted from the TIMIT database, a widely used resource for research in speech recognition. The data was downloaded from the companion website[8] of the textbook [17]—it consists of 4509 log-periodograms of length 256–here, $(n, p) = (4509, 256)$. We considered a sequence of solutions for the ML factor analysis problem for 18 equi-spaced values of $r \in [1, 27]$. The sample covariance matrix $\mathbf{S}$ was poorly conditioned. The condition number of $\mathbf{S}$ was $\sim 3.9 \times 10^3$.

Example c (ZipCode Data), $(n > p)$: This is the well-known ZipCode dataset[9] which was generated by scanning normalized handwritten digits by the U.S. Postal Service.

[8] Available at: https://web.stanford.edu/~hastie/ElemStatLearn/datasets/phoneme.data.

[9] Available at https://web.stanford.edu/~hastie/ElemStatLearn/datasets/zip.test.gz.

These are 16×16 grayscale images corresponding to digits (0–9) that are normalized/deslanted. The images were vectorized, and we created a data matrix comprising of digits 0 and 6. The data matrix $\mathbf{X}_{n \times p}$ had dimensions $(n, p) = (1858, 249)$. We considered a sequence of solutions for the ML factor analysis problem for 15 equi-spaced values of $r \in [1, 17]$. The condition number of $\mathbf{S}$ here was $\sim 3.8 \times 10^{17}$.

Example d (Cancer Data), $(p > n)$: This is a highdimensional microarray dataset available from[10] the companion website of [17]. It consists of gene expression measurements for $p = 16063$ genes from $n = 144$ individuals. We considered a sequence of solutions for the ML factor analysis problem for 15 equi-spaced values of $r \in [1, 22]$. The condition number of $\mathbf{S}$ here was $\sim 10^{20}$.

For the synthetic dataset (Example a), we took $\epsilon = 10^{-3}$ (in Problem (2)). For Examples b,c,d, the condition number of $\mathbf{S}$ was quite high—we took $\epsilon = 10^{-10}$ and Algorithm 1 was provided with this choice of ϵ. For Algorithm 1, we did not use the continuation strategy as described in Sect. 2.6. Algorithm 1 for $r = 1$ was initialized by drawing entries from a uniform [0, 1] distribution; and we used warm-start continuation to compute solutions for the other r values. For `Fa` we used its default initialization scheme since it does not allow for warm-starts.

For all cases with $n < p$, the low-rank SVD step in Algorithm 1 was performed according to the description given in Sect. 2.5 (for the case $p \gg n$). For $n > p$, we used the low-rank SVD method of Sect. 2.5 (for the case $n > p$).

Table 1 shows the results for all the four examples. We observe that FACTMLE clearly works extremely well in terms of obtaining a good objective value in much smaller run-times, compared to `Fa`. What is most important however, is that FACTMLE is numerically robust—in fact, much more stable and reliable than `Fa` which often encounters problems with convergence. In many cases (across multiple replications and multiple values of r) `Fa` is found to be trapped in poor fixed points with suboptimal objective values. These are referred to by the moniker "$>$*" in Table 1. For the synthetic datasets, the problem with convergence occurs even for small values of r (less than 5), and the issue is aggravated whenever r becomes larger than 5. Recall that the true rank r_0 in the underlying model is 5; and in our experiments, the largest value of r we took was 18. For the real datasets, the nonconvergence of `Fa` commonly occurs across both small and large values of $r \geq 1$ (recall that the largest values of r for the Zipcode, Phoneme and Cancer datasets were 17, 27 and 22 respectively). We observe problems with convergence as soon as the precision level becomes higher (denoted by the first column 'Tol' in Table 1)—indicating that `Fa` rarely gets solutions as accurate as those available from FACTMLE. We note that all the real datasets have a sample covariance matrix with a very large condition number, making these problems computationally challenging—however, we expect to often encounter datasets of this form in real-life settings—thus, having a good robust algorithm is of paramount importance. In this respect, FACTMLE seems to have a clear edge over the classical method `Fa`.

[10] Data available at http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/14cancer.info.

4 Generalizing beyond a diagonal $\boldsymbol{\Phi}$

Here we discuss how our proposed framework can be adapted to handle more general convex constraints (not necessarily diagonal) on $\boldsymbol{\Phi}$. Towards this end, we have the following remark:

Remark 2 For any $r \in [p]$ and any $\boldsymbol{\Phi} \succeq \mathbf{0}$ (not necessarily diagonal) the function

$$\boldsymbol{\Phi} \mapsto F_2(\boldsymbol{\Phi}) := \sum_{i=1}^{r} \left(\log\left(\max\{1, \lambda_i^*\}\right) - \max\{1, \lambda_i^*\} + 1 \right)$$

is concave in $\boldsymbol{\Phi} \succeq \mathbf{0}$; where, $\{\lambda_i^*\}_1^p$ are the eigenvalues of $\mathbf{S}^{\frac{1}{2}} \boldsymbol{\Phi} \mathbf{S}^{\frac{1}{2}}$. This follows via an argument similar to that used in Proposition 5.

When $\boldsymbol{\Phi} \succ \mathbf{0}$ lies in a convex set $\mathtt{X}$, Problem (32) gets modified to:

$$\text{minimize} \quad H(\boldsymbol{\Phi}) := F_1(\boldsymbol{\Phi}) - F_2(\boldsymbol{\Phi}) \quad \text{s.t.} \quad \boldsymbol{\Phi} \succ \mathbf{0}, \boldsymbol{\Phi} \in \mathtt{X} \tag{51}$$

where, $F_1(\boldsymbol{\Phi}) := -\log\det(\boldsymbol{\Phi}) + \mathrm{tr}(\boldsymbol{\Phi}\mathbf{S})$ and $F_2(\boldsymbol{\Phi})$ is as defined above (Remark 2). Note that $F_1(\boldsymbol{\Phi})$ and $F_2(\boldsymbol{\Phi})$ are convex in $\boldsymbol{\Phi}$. In a FA model, $\boldsymbol{\Phi} \in \mathtt{X}$ may encode structure beyond a simple diagonal matrix. Note that $\boldsymbol{\Phi}$ is the precision matrix (inverse covariance matrix) of the error vector $\mathbf{u}$ (cf display (1)) and a structure on $\boldsymbol{\Phi}$ is a statement regarding the conditional (in)dependence structure [17] in $\mathbf{u}$ (assuming $\mathbf{u}$ follows a Gaussian distribution). For example, $\boldsymbol{\Phi}$ can be block-diagonal, banded or have entries with a small ℓ_1-norm.[11]

The difference of convex optimization procedure described in Sect. 2.3 can also be applied to Problem (51). To this end, note that step (34) in Algorithm 1 gets modified to:

$$\boldsymbol{\Phi}^{(k+1)} \in \arg\min \left\{ F_1(\boldsymbol{\Phi}) - \langle \partial F_2(\boldsymbol{\Phi}^{(k)}), \boldsymbol{\Phi} \rangle : \boldsymbol{\Phi} \in \mathtt{X}, \boldsymbol{\Phi} \succ \mathbf{0} \right\}. \tag{52}$$

Clearly, the efficiency of this procedure depends upon how easily subproblem (52) can be solved—this depends upon $\mathtt{X}$. In general, we expect the cost of solving (52) and hence the cost of obtaining a good solution to Problem (51) to be higher than the case where $\boldsymbol{\Phi}$ is diagonal. If $\boldsymbol{\Phi}$ is banded, block-diagonal, or has small ℓ_1-norm; $\mathtt{X}$ can be described by a simple polyhedral set. Note also that the log-det barrier appearing in $F_1(\boldsymbol{\Phi})$ will implicitly enforce $\boldsymbol{\Phi} \succ \mathbf{0}$.

Below, we illustrate two special cases:

- $\boldsymbol{\Phi}$ is block-diagonal (i.e., of the form (4) for a-priori specified block structure)
- $\boldsymbol{\Phi}$ is banded (i.e., of the form (5) with a-priori specified bandwidth b).

[11] Note that additional assumptions may be needed to ensure a unique decomposition of $\boldsymbol{\Sigma}$ into its components $\boldsymbol{\Psi}$ and $\boldsymbol{\Theta} = \mathbf{L}\mathbf{L}^\top$. However, the optimization task is well defined even in the absence of such identifiability constraints.

For the above cases, a subgradient $\partial F_2(\mathbf{\Phi})$ can be calculated as follows:

$$\partial F_2(\mathbf{\Phi}) = \begin{cases} \text{Blkdiag}_{\mathbf{p}}\left(\mathbf{S}^{\frac{1}{2}}\mathbf{V}\mathbf{D}_1\mathbf{V}^\top\mathbf{S}^{\frac{1}{2}}\right) & \text{if } \mathbf{\Phi} \text{ is block-diagonal} \\ \text{Banded}_b\left(\mathbf{S}^{\frac{1}{2}}\mathbf{V}\mathbf{D}_1\mathbf{V}^\top\mathbf{S}^{\frac{1}{2}}\right) & \text{if } \mathbf{\Phi} \text{ is b-banded,} \end{cases}$$

where, $\mathbf{S}^{\frac{1}{2}}\mathbf{\Phi}\mathbf{S}^{\frac{1}{2}} = \mathbf{V}\text{diag}(\lambda_1^*, \ldots, \lambda_p^*)\mathbf{V}^\top$ is an eigen-decomposition; and the diagonal matrix $\mathbf{D}_1$ is given by equation (37). Note that when the matrix $\mathbf{\Phi}$ is block-diagonal, the convex optimization problem (52) has a closed form solution. Concretely, we have

$$\mathbf{\Phi}^{(k+1)} = \left[\text{Blkdiag}_{\mathbf{p}}(\mathbf{S}) - \partial F_2(\mathbf{\Phi}^{(k)})\right]^{-1}. \tag{53}$$

Notice that this inverse can be calculated efficiently by utilizing the block-diagonal structure of the matrix. For instance, if the matrix $\text{Blkdiag}_{\mathbf{p}}(\mathbf{S}) - \partial F_2(\mathbf{\Phi}^{(k)})$ is a block-diagonal matrix with m-blocks of equal sizes, the total cost of the inversion in equation (53) will be $O(p^3/m^2)$, whereas the cost of inverting a dense $p \times p$ matrix is $O(p^3)$.

If $\mathbf{\Phi}$ is banded, Problem (52) will involve solving a semidefinite optimization problem where $\mathbf{\Phi}$ is of the form (5). This problem can be solved with off-the-shelf solvers (e.g, we can use standard modeling tools like `cvx`, the SCS solver [28], etc) for small/moderate scale problems. We used our own implementation of a gradient descent algorithm to solve Problem (52). Our algorithm is a special instance of the deterministic algorithm presented in [3] for the graphical lasso problem [15]. Concretely, let $G(\mathbf{\Phi})$ denote the objective function of Problem (52). Note that the gradient of $G(\mathbf{\Phi})$ is locally Lipschitz: if $\mathbf{\Phi} \approx \mathbf{\Phi}'$ (and both are positive definite) then the following holds $\|\nabla G(\mathbf{\Phi}) - \nabla G(\mathbf{\Phi}')\| \leq L\|\mathbf{\Phi} - \mathbf{\Phi}'\|$ (where, the norm is the Frobenius norm) for $L \approx \lambda_1(\mathbf{\Phi}^{-2})$. At every iteration t (of the gradient descent method) we compute L based on the previous estimate of $\mathbf{\Phi}$. We use a step-size $\alpha = \tau/L$ where, we take the largest value of $\tau \in \{0.5^m : m = 1, 2, \ldots\}$ to ensure a decrease in the value of the objective function.

Another regularization scheme on $\mathbf{\Phi}$ might be to constrain the ℓ_1-norm of its entries. This can be imposed by adding an ℓ_1-norm penalty on the elements of $\mathbf{\Phi}$, in the objective function (51). In this case, Problem (52) will be given by the graphical lasso [6,15] problem[12] and can be solved via standard solvers [3,15].

We note that the framework outlined above, will not apply if the structure on $\mathbf{\Phi}$ admits a complicated, nonconvex description—this may arise for example, if one imposes a banded structure on $\mathbf{\Psi}$—this will lead to a nonconvex description of $\mathbf{\Phi}$.

4.1 Computational experiments

We present some computational experiments for the cases when $\mathbf{\Phi}$ is not diagonal. In our first example, we consider the case when $\mathbf{\Phi} \succ \mathbf{0}$ is block-diagonal; in the second

[12] This encourages a conditional independence structure among the variables in $\mathbf{u}$ (assuming $\mathbf{u}$ follows a multivariate normal distribution).

example, we consider $\mathbf{\Phi} \succ \mathbf{0}$ to be tri-diagonal (i.e., a banded matrix with $b = 1$ or a 1-banded matrix). Finally, we present run-time comparisons across $\mathbf{\Phi}$ being diagonal, block-diagonal or tri-diagonal.

Synthetic data generation: Our data generation scheme is similar to Sect. 3.1. We generated (the true) $\mathbf{L}^0 \in \Re^{p \times r_0}$ (with $r_0 = 8$ for tri-diagonal case, and $r_0 = 10$ for block-diagonal case) with entries drawn iid from $N(1, 1)$. For the block-diagonal case, the diagonal blocks of the matrix $\mathbf{\Phi}^0$ were generated as follows: we first generate square matrices $\mathbf{A}_i, i = 1, 2$ with entries drawn iid from an exponential distribution with mean 1. We next generate square matrices $\mathbf{\Phi}_i^0 = \mathbf{A}_i \mathbf{A}_i^\top$ and then create the (true) block-diagonal matrix $\mathbf{\Phi}^0$ with diagonal blocks $\mathbf{\Phi}_1^0$ and $\mathbf{\Phi}_2^0$. For the tri-diagonal example, we first simulate a vector $\boldsymbol{b} = (b_1, \ldots, b_{p-1})$ with entries taken to be the absolute value of an independent Gaussian distribution with mean 0 and standard deviation 5. We then define the tri-diagonal matrix $\mathbf{\Phi}^0$ as follows:

$$\mathbf{\Phi}^0 = \begin{bmatrix} a_1 & b_1 & \ldots & 0 \\ b_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & b_{p-1} \\ 0 & \ldots & b_{p-1} & a_p \end{bmatrix}$$

where, the vector $\boldsymbol{a} = (a_1, a_2, \ldots, a_p)$ is simulated in such a way that matrix $\mathbf{\Phi}^0$ is diagonally dominant—thereby ensuring that the matrix $\mathbf{\Phi}^0$ is PSD and tri-diagonal. Concretely, in our simulation, the scalar a_i satisfies $a_i = b_{i-1} + b_i + v_i$, where $b_0 = b_p = 0$ and each v_i is set to be the absolute value of an independent Gaussian distribution with mean 0 and standard deviation 5. Finally, we define the (true) covariance matrix $\mathbf{\Sigma}^0 = (\mathbf{\Phi}^0)^{-1} + \mathbf{L}^0 \mathbf{L}^{0\top}$ and generate the data matrix $\mathbf{X}_{n \times p}$ with rows corresponding to n independent samples from a multivariate Gaussian distribution with mean $\mathbf{0}$ and covariance $\mathbf{\Sigma}^0$. For the block-diagonal case, we consider $(n, p) = (2000, 40)$ and in the tri-diagonal case we took $(n, p) = (10,000, 100)$.

Run-times for block-diagonal and tri-diagonal instances: We initialize our algorithms with a random diagonal matrix, where the diagonal entries are drawn form the uniform distribution in [0, 1]. We did not use warm-start continuation across different r values—we expect that warm-starts might speed-up the overall computation time. For our algorithm, we set the tolerance level η (for convergence based on objective value difference, as explained in Sect. 2.3) to 10^{-4}; and we set the maximum number of iterations to be 1000. For the banded case, the inner semidefinite optimization problem (52) was solved to a tolerance of 10^{-6} (based on relative difference in successive objective values).

The result for the block-diagonal example is shown in Table 2 and the banded case is shown in Table 3. We study the performance of our proposed algorithm based on two criteria: (a) the objective value and (b) the run-time of the algorithm. Note that due to the nonconvexity of Problem (51), the solution obtained will have a dependence on the initialization. In these examples, the dependence on initialization was found to be more pronounced than the case where $\mathbf{\Phi}$ was diagonal. Tables 2 and 3 show the best

Table 2 Table showing performance of our proposed algorithm when $\boldsymbol{\Phi}$ is block-diagonal (here, $(n, p) = (2000, 40)$)

Rank (r)	Reference objective	FACTMLE objective	Time (in secs)
2	1091.23	449.30 (0.00)	0.01 (0.00)
4	857.90	443.84 (0.17)	0.31 (0.02)
6	512.29	440.71 (0.12)	0.37 (0.02)
8	462.80	438.48 (0.14)	0.39 (0.02)

"Reference objective" corresponds to the objective value based on the true underlying generative model (as explained in the text). "FACTMLE objective" refers to the objective value obtained by our algorithm (best solution obtained across 50 initializations)—it is smaller than the reference objective—showing that our algorithm reaches a good solution, despite the highly nonconvex landscape of the optimization problem. Time (in secs) denotes the average time across the replications, no warm-start continuation is used across different r-values. Numbers within parenthesis denote the standard errors

Table 3 This table is similar to Table 2, but considers the case when $\boldsymbol{\Phi}$ is banded (here, $(n, p) = (10,000, 100)$)

Rank (r)	Reference objective	FACTMLE objective	Time (in s)
1	3777.57	117.11 (0.00)	0.64 (0.02)
3	2040.21	95.03 (0.18)	4.48 (0.17)
5	757.19	63.45 (0.07)	9.97 (0.31)
7	189.16	21.17 (0.09)	15.85 (0.55)

Once again, we see that our proposed algorithm does a good job in obtaining good solutions. The times are seen to increase with r—note that for every r, we use a random diagonal initialization for our algorithm, as described in the text (no warm-start continuation is used across different r values)

Table 4 Table showing run-time comparisons to obtain a solution to Problem (51), when $\boldsymbol{\Phi}$ is specified to be diagonal, block-diagonal or tri-diagonal—the algorithm run-times are denoted by Diag, Blk-diag and Tri-diag, respectively

p	Time ratio (example 1)		Time ratio (example 2)		Time ratio (example 3)	
	Blk-diag / Diag	Tri-diag / Diag	Blk-diag / Diag	Tri-diag / Diag	Blk-diag / Diag	Tri-diag / Diag
100	10.47 (0.39)	662.33 (39.05)	7.41 (0.28)	149.02 (12.11)	1.41 (0.06)	1419.31 (20.98)
300	5.47 (0.18)	1017.70 (55.55)	7.90 (0.11)	389.42 (17.97)	1.31 (0.01)	1379.07 (9.86)
500	4.48 (0.05)	1362.04 (59.22)	8.64 (0.10)	642.78 (15.66)	1.94 (0.02)	1940.98 (23.97)

The cells display the run-time ratios averaged across replications (with standard errors within parenthesis). For further details see text

objective value obtained among 50 replications (for every value of r), and we also mention the standard deviation across the objective values obtained. We are not aware of any publicly available implementation for Problem (51). Hence, for reference, we show the value of the negative log-likelihood based on the true underlying model. Note that when $r_0 > r$ the matrix $\boldsymbol{\Sigma}^0$ is not a feasible solution for Problem (51). We compute the objective value (i.e., the negative log-likelihood) at $\tilde{\boldsymbol{\Sigma}} = \boldsymbol{\Psi}^0 + \tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top$, where $\tilde{\mathbf{L}}$ is the best r rank approximation of $\mathbf{L}^0$. Observe that in both the examples (Tables 2 and 3),

our algorithm demonstrates good performance both in terms of total run-time and the negative log-likelihood (objective value). The time taken for the banded case is longer than the block-diagonal case partly because the value of p is larger.

Timing comparisons (diagonal versus non-diagonal): We compare algorithm run-times for different structures on $\mathbf{\Phi}$ (diagonal, non-diagonal) on synthetic datasets (as above) where, the underlying $\mathbf{\Phi}^0$ is either diagonal, block diagonal or tri-diagonal. We study the examples:

- example 1: we let $\mathbf{\Phi}^0$ be block-diagonal (with 2 blocks of equal sizes), as discussed earlier in Sect. 4.1.
- example 2: we set $\mathbf{\Phi}^0$ to be a diagonal matrix with diagonal entries set to those of the matrix used in example 1.
- example 3: we let $\mathbf{\Phi}^0$ be a tri-diagonal matrix, as discussed earlier in Sect. 4.1.

We set $r_0 = 10$, draw $n = 10,000$ samples (as above) and take different values of $p \in \{100, 200, 500\}$. For every example, we consider three versions of our algorithm: "FACTMLE- DIAG", "FACTMLE- BLK- DIAG" and "FACTMLE- TRI- DIAG"—they correspond to $\mathbf{\Phi}$ in Problem (51) being diagonal, block-diagonal (with 2 blocks) and tri-diagonal, respectively. This setup allows us to study the run-time of our algorithms under model misspecification in $\mathbf{\Phi}$, unlike the synthetic examples considered earlier. We study the times taken by the algorithms to obtain a stationary point of Problem (51). For every algorithm, we consider the total time[13] taken by it to compute a path of solutions for $r \in \{1, \ldots, 10\}$ using warm-start continuation across r. For $r = 1$, we initialize with a diagonal matrix with entries drawn independently from a uniform distribution on $[0, 1]$. We perform 15 replications for every problem. Other algorithm specifications are taken as before. Table 4 compares the algorithms: we present two ratios—(a) $\frac{\text{Blk-diag}}{\text{Diag}}$ denoting the ratio of the times taken by FACTMLE- BLK- DIAG and FACTMLE- DIAG; and (b) $\frac{\text{Tri-diag}}{\text{Diag}}$ denoting the ratio of the times taken by FACTMLE- TRI- DIAG and FACTMLE- DIAG. The average ratios and standard errors are shown in Table 4. FACTMLE- DIAG is found to take between 0.05-1.3 seconds (approx.), with the longest corresponding to example 3 ($p = 500$). Run-times of all algorithms are found to increase with p. The run-time for FACTMLE- TRI- DIAG is consistently higher than FACTMLE- BLK- DIAG and FACTMLE- DIAG—this is due to the inner semidefinite optimization problem for which no closed form solution exists (unlike the cases where $\mathbf{\Phi}$ is diagonal or block-diagonal). FACTMLE- BLK- DIAG and FACTMLE- DIAG seem to have comparable run-times under model-misspecification (example 3); in all other cases, FACTMLE- DIAG is found to be faster (approx. 4-11 times) than FACTMLE- BLK- DIAG.

Note that when p is of the order of a few thousands, FACTMLE- TRI- DIAG will become prohibitely expensive. FACTMLE- BLK- DIAG will also be expensive if the block-sizes become large. FACTMLE- DIAG can still be computationally feasible if n is small (cf Sects. 2.5 and 3).

[13] We declare convergence if the relative change in successive objective values is smaller than 10^{-4}.

5 Conclusions

In this paper, we present a new algorithmic framework for the well-known Gaussian ML factor analysis problem. This is a challenging rank constrained nonconvex optimization problem, for which very few reliable computational algorithms exist. A key ingredient of our approach is a reformulation, where we minimize a continuous (non-differentiable) nonconvex objective function including spectral functions, subject to simple convex constraints. We employ an algorithm based on difference of convex optimization and use structured SVD computations (involving the SVD of a $n \times p$ matrix) for scalability to large problems. Contrary to many standard FA algorithms, where $\mathbf{S}$ is assumed to be of full rank, our approach applies to settings where $\mathbf{S}$ is rank deficient—making it useful for highdimensional applications where, n can be much smaller than p. Our approach is found to perform better than existing algorithms for ML factor analysis, in terms of obtaining high-quality solutions in significantly smaller runtimes. Compared to existing algorithms, our approach appears to have a significant advantage when: (a) $\mathbf{S}$ is full rank and the number of factors is not too small (b) $\mathbf{S}$ is rank deficient or $\mathbf{S}$ is of full rank but poorly conditioned. In this paper, we do not discuss how to obtain dual bounds or certificates of global optimality (e.g., based on mixed integer optimization methods as in the work of [8])—this is an interesting direction for future research. We also discuss how to extend our framework to address the case where $\mathbf{\Phi}$ is not diagonal, but is allowed to lie in a simple convex set. The cost of this method depends upon how well we can solve Problem (52)—developing specialized solvers for this problem (that apply to more general structures on $\mathtt{X}$) is an important direction for future research.

Acknowledgements The authors will like to thank the Editors and the anonymous Reviewers for their helpful feedback and detailed comments that helped improve this manuscript. Rahul Mazumder was partially supported by ONR-N000141512342, ONR-N000141812298 (YIP) and NSF-IIS1718258. The authors will also like to thank Columbia University (Statistics department) for hosting Koulik Khamaru as a summer intern, when this work started.

A Appendix

Proposition 10 (See Sect. 6 in [10]) *Suppose the function $g : \mathbf{E} \mapsto (-\infty, \infty)$ is convex, and the point x lies in interior of dom(g) with $\mathbf{E} \subset \Re^m$. Let $x^r, x \in \mathbf{E}$ and v^r be a subgradient of g evaluated at x^r for $r \geq 1$. If $x^r \to x$ and $v^r \to v$ as $r \to \infty$, then v is a subgradient of g evaluated at x.*

A.1 Proof of Proposition 9

Note that the objective function $f(\boldsymbol{\phi})$ (see (22)) is unbounded above when $\phi_i \to 0$ for any $i \in [p]$—see also Proposition 3. This implies that there exists a $\alpha > 0$ such that $\boldsymbol{\phi}^{(k)} \in [\alpha, \frac{1}{\epsilon}]^p$ for all k. The boundedness of the sequence $\boldsymbol{\phi}^{(k)}$ implies the existence of a limit point of $\boldsymbol{\phi}^{(k)}$, say, $\boldsymbol{\phi}^*$. Let $\boldsymbol{\phi}^{(k_j)}$ be a subsequence such that $\boldsymbol{\phi}^{(k_j)} \to \boldsymbol{\phi}^*$.

Note that for every k, $\boldsymbol{\phi}^{(k+1)} \in \arg\min_{\boldsymbol{\phi}\in\mathcal{C}} F(\boldsymbol{\phi}; \boldsymbol{\phi}^{(k)})$ is equivalent to

$$\left\langle \nabla f_1(\boldsymbol{\phi}^{(k+1)}) - \partial f_2(\boldsymbol{\phi}^{(k)}), \boldsymbol{\phi} - \boldsymbol{\phi}^{(k+1)} \right\rangle \geq 0 \ \ \forall \boldsymbol{\phi} \in \mathcal{C}. \tag{54}$$

Now consider the sequence $\boldsymbol{\phi}^{(k_j)}$ as $k_j \to \infty$. Using the fact that $\boldsymbol{\phi}^{(k+1)} - \boldsymbol{\phi}^{(k)} \to \mathbf{0}$; it follows from the continuity of $\nabla f_1(\cdot)$ that: $\nabla f_1(\boldsymbol{\phi}^{(k_j+1)}) \to \nabla f_1(\boldsymbol{\phi}^*)$ as $k_j \to \infty$.

Note that $\partial f_2(\boldsymbol{\phi}^{(k_j)})$ (see (36)) is bounded as $\boldsymbol{\phi}^{(k_j)} \in [\alpha, \frac{1}{\epsilon}]^p$. Passing onto a further subsequence $\{k'_j\}$ if necessary, it follows that $\partial f_2(\boldsymbol{\phi}^{(k'_j)}) \to \vartheta$ (say). Using Proposition 10, we conclude that ϑ is a subgradient of f_2 evaluated at $\boldsymbol{\phi}^*$. As $k'_j \to \infty$, the above argument along with (54) implies that:

$$\left\langle \nabla f_1(\boldsymbol{\phi}^*) - \partial f_2(\boldsymbol{\phi}^*), \boldsymbol{\phi} - \boldsymbol{\phi}^* \right\rangle \geq 0 \ \ \forall \boldsymbol{\phi} \in \mathcal{C}, \tag{55}$$

where, $\partial f_2(\boldsymbol{\phi}^*)$ is a subgradient of f_2 evaluated at $\boldsymbol{\phi}^*$. (55) implies that $\boldsymbol{\phi}^*$ is a first order stationary point.

A.2 Representing $\partial h(\boldsymbol{\Psi}, \mathbf{L})/\partial \mathbf{L} = \mathbf{0}$

Note that $\partial h(\boldsymbol{\Psi}, \mathbf{L})/\partial \mathbf{L} = 0$ is equivalent to setting (12) to zero, which leads to $\mathbf{L} = \mathbf{S}\boldsymbol{\Sigma}^{-1}\mathbf{L}$. By applying Sherman Woodbury formula on $(\boldsymbol{\Psi} + \mathbf{L}\mathbf{L}^\top)^{-1}$, we have the following:

$$\begin{aligned}
\mathbf{L} &= \mathbf{S}\boldsymbol{\Sigma}^{-1}\mathbf{L} \\
&= \mathbf{S}\left(\boldsymbol{\Psi}^{-1} - \boldsymbol{\Psi}^{-1}\mathbf{L}\left(\mathbf{I} + \mathbf{L}^\top\boldsymbol{\Psi}^{-1}\mathbf{L}\right)^{-1}\mathbf{L}^\top\boldsymbol{\Psi}^{-1}\right)\mathbf{L} && (56)\\
&= \mathbf{S}\boldsymbol{\Psi}^{-1}\mathbf{L} - \mathbf{S}\boldsymbol{\Psi}^{-1}\mathbf{L}\left(\left(\mathbf{I} + \mathbf{L}^\top\boldsymbol{\Psi}^{-1}\mathbf{L}\right)^{-1}\mathbf{L}^\top\boldsymbol{\Psi}^{-1}\mathbf{L}\right) && (57)\\
&= \mathbf{S}\boldsymbol{\Psi}^{-1}\mathbf{L} - \mathbf{S}\boldsymbol{\Psi}^{-1}\mathbf{L}\left(\mathbf{I} - \left(\mathbf{I} + \mathbf{L}^\top\boldsymbol{\Psi}^{-1}\mathbf{L}\right)^{-1}\right) && (58)\\
&= \mathbf{S}\boldsymbol{\Psi}^{-1}\mathbf{L}\left(\mathbf{I} + \mathbf{L}^\top\boldsymbol{\Psi}^{-1}\mathbf{L}\right)^{-1}, && (59)
\end{aligned}$$

where, Eqn (56) follows from (8); Eqn (58) follows from (57) by using the observation that for a PSD matrix $\mathbf{B}$, we have the following identity: $(\mathbf{I}+\mathbf{B})^{-1}\mathbf{B} = \mathbf{I} - (\mathbf{I}+\mathbf{B})^{-1}$ (this can be verified by simple algebra).

Finally, we note that (13) follows by using the definition of $\mathbf{L}^*$ and $\mathbf{S}^*$ in (59) and doing some algebraic manipulations.

References

1. Ahn, M., Pang, J.-S., Xin, J.: Difference-of-convex learning: directional stationarity, optimality, and sparsity. SIAM J. Optim. **27**(3), 1637–1665 (2017)
2. Anderson, T.: An Introduction to Multivariate Statistical Analysis, 3rd edn. Wiley, New York (2003)

3. Atchadé, Y.F., Mazumder, R., Chen, J.: Scalable computation of regularized precision matrices via stochastic optimization (2015). arXiv preprint arXiv:1509.00426
4. Bai, J., Li, K.: Statistical analysis of factor models of high dimension. Ann. Stat. **40**(1), 436–465 (2012)
5. Bai, J., Ng, S.: Large dimensional factor analysis. Found. Trends Econom. **3**(2), 89–163 (2008)
6. Banerjee, O., El Ghaoui, L., d'Aspremont, A.: Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. J. Mach. Learn. Res. **9**, 485–516 (2008)
7. Bartholomew, D., Knott, M., Moustaki, I.: Latent Variable Models and Factor Analysis: A Unified Approach. Wiley, London (2011)
8. Bertsimas, D., Copenhaver, M.S., Mazumder, R.: Certifiably optimal low rank factor analysis. J. Mach. Learn. Res. **18**(29), 1–53 (2017)
9. Bishop, C.: Pattern Recognition and Machine Learning. Springer, New York (2006)
10. Borwein, J., Lewis, A.: Convex Analysis and Nonlinear Optimization. Springer, New York (2006)
11. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
12. Brand, M.: Fast low-rank modifications of the thin singular value decomposition. Linear Algebra Appl. **415**(1), 20–30 (2006)
13. Davis, C.: All convex invariant functions of hermitian matrices. Archiv der Mathematik **8**(4), 276–278 (1957)
14. Dinh, T.P., Le T., Hoai A.: Recent advances in dc programming and DCA. In: Transactions on Computational Intelligence XIII, pp. 1–37. Springer, New York (2014)
15. Friedman, J., Hastie, T., Tibshirani, R.: Sparse inverse covariance estimation with the graphical lasso. Biostatistics **9**, 432–441 (2007)
16. Golub, G., Van Loan, C.: Matrix Computations, vol. 3. JHU Press, Baltimore (2012)
17. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning, Second Edition: Data Mining, Inference, and Prediction (Springer Series in Statistics). Springer, New York (2009)
18. Hiriart-Urruty, J-B.: Generalized differentiability/duality and optimization for problems dealing with differences of convex functions. In: Convexity and duality in optimization, pp. 37–70. Springer, New York (1985)
19. Jöreskog, K.G.: Some contributions to maximum likelihood factor analysis. Psychometrika **32**(4), 443–482 (1967)
20. Larsen, R.M.: PROPACK-Software for large and sparse SVD calculations (2004). http://sun.stanford.edu/rmunk/PROPACK
21. Lawley, D.N.: the estimation of factor loadings by the method of maximum likelihood. Proc. R. Soc. Edinb. **60**(01), 64–82 (1940)
22. Lawley, D.N.: Some new results in maximum likelihood factor analysis. Proc. R. Soc. Edinb. **67**(01), 256–264 (1967)
23. Lawley, D.N., Maxwell, A.E.: Factor Analysis as a Statistical Method, 2nd edn. Butterworth, London (1971)
24. Lewis, A.: Derivatives of spectral functions. Math. Oper. Res. **21**(3), 576–588 (1996)
25. Lewis, A.S.: Convex analysis on the hermitian matrices. SIAM J. Optim. **6**, 164–177 (1996)
26. Mardia, K., Kent, J., Bibby, J.: Multivariate Analysis. Academic Press, London (1979)
27. Nouiehed, M., Pang, J.-S., Razaviyayn, M.: On the pervasiveness of difference-convexity in optimization and statistics (2017). arXiv preprint arXiv:1704.03535
28. O'Donoghue, B., Chu, E., Parikh, N., Boyd, S.: Conic optimization via operator splitting and homogeneous self-dual embedding. J. Optim. Theory Appl. **169**(3), 1042–1068 (2016)
29. Pang, J.-S., Razaviyayn, M., Alvarado, A.: Computing b-stationary points of nonsmooth dc programs. Math. Oper. Res. **42**(1), 95–118 (2016)
30. Pham Dinh, T., Ngai, H.V., Le Thi, H.A.: Convergence analysis of dc algorithm for dc programming with subanalytic data (2013) **(preprint)**
31. Robertson, D., Symons, J.: Maximum likelihood factor analysis with rank-deficient sample covariance matrices. J. Multiv. Anal. **98**(4), 813–828 (2007)
32. Rockafellar, R.T., Wets, R.J.-B.: Variational Analysis, vol. 317. Springer, New York (2009)
33. Rubin, D.B., Thayer, D.T.: Em algorithms for ml factor analysis. Psychometrika **47**(1), 69–76 (1982)
34. Saunderson, J., Chandrasekaran, V., Parrilo, P., Willsky, A.: Diagonal and low-rank matrix decompositions, correlation matrices, and ellipsoid fitting. SIAM J. Matrix Anal. Appl. **33**(4), 1395–1416 (2012)
35. Shapiro, A., Ten Berge, J.: Statistical inference of minimum rank factor analysis. Psychometrika **67**, 79–94 (2002)

36. Spearman, C.: "General Intelligence," objectively determined and measured. Am. J. Psychol. **15**, 201–293 (1904)
37. Tuy, H.: Dc optimization: theory, methods and algorithms. In: Handbook of Global Optimization, pp. 149–216. Springer, New York (1995)
38. Vangeepuram, S., Lanckriet, G.R.G.B.: On the convergence of the concave-convex procedure. In: Advances in Neural Information Processing Systems, (NIPS), vol. 22. MIT Press (2009)
39. Yuille, A., Rangarajan, A.: The concave-convex procedure (cccp). Neural Comput. **15**, 915–936 (2003)