An Alternating Manifold Proximal Gradient Method for Sparse PCA and Sparse CCA

Shixiang Chen∗ Shiqian Ma† Lingzhou Xue‡ Hui Zou§

March 27, 2019

Abstract

Sparse principal component analysis (PCA) and sparse canonical correlation analysis (CCA) are two essential techniques from high-dimensional statistics and machine learning for analyzing large-scale data. Both problems can be formulated as an optimization problem with nonsmooth objective and nonconvex constraints. Since non-smoothness and nonconvexity bring numerical difficulties, most algorithms suggested in the literature either solve some relaxations or are heuristic and lack convergence guarantees. In this paper, we propose a new alternating manifold proximal gradient method to solve these two high-dimensional problems and provide a unified convergence analysis. Numerical experiment results are reported to demonstrate the advantages of our algorithm.

1 Introduction

Principal Component Analysis (PCA), invented by Pearson [37], is widely used in dimension reduction. Let $X = [X_1, \ldots, X_p] \in \mathbb{R}^{n \times p}$ be a given data matrix whose column means are all 0. Assume the singular value decomposition (SVD) of $X$ is $X = UDV^\top$, then it is known that $Z = UD$ are the principal components (PCs) and the columns of $V$ are the corresponding loadings of the PCs. In other words, the first PC can be defined as $Z_1 = \sum_{j=1}^p \alpha_{1j} X_j$ with $\alpha_1 = (\alpha_{11}, \ldots, \alpha_{1p})^\top$ maximizing the variance of $Z_1$, i.e.,

$$\alpha_1 = \arg \max_\alpha \alpha^\top \hat{\Sigma} \alpha, \quad \text{s.t.} \|\alpha\|_2 = 1,$$

where $\hat{\Sigma} = (X^\top X)/(n - 1)$ is the sample covariance matrix. The rest PCs are defined as

$$\alpha_{k+1} = \arg \max_\alpha \alpha^\top \hat{\Sigma} \alpha, \quad \text{s.t.} \|\alpha\|_2 = 1, \alpha^\top \alpha_l = 0, \forall 1 \leq l \leq k.$$

Canonical correlation analysis (CCA), introduced by Hotelling [23], is another widely used tool, which explores the relation between two sets of variables. For random variables $x \in \mathbb{R}^p$ and $y \in \mathbb{R}^q$, CCA seeks linear combinations of $x$ and $y$ such that the resulting values are mostly correlated. That is, it targets to solve the following optimization problem:

$$\max_{u \in \mathbb{R}^p, v \in \mathbb{R}^q} \frac{u^\top \Sigma_{xy} v}{\sqrt{u^\top \Sigma_x u} \sqrt{v^\top \Sigma_y v}}, \quad (1.1)$$
where $\Sigma_x$ and $\Sigma_y$ are covariance of $x$ and $y$ respectively, $\Sigma_{xy}$ is their covariance matrix, and $u \in \mathbb{R}^p$ and $v \in \mathbb{R}^q$ are the first canonical vectors. It can be shown that solving (1.1) corresponds to computing the SVD of $\Sigma_x^{-1/2}\Sigma_{xy}\Sigma_y^{-1/2}$. In practice, given two centered data sets $X \in \mathbb{R}^{n \times p}, Y \in \mathbb{R}^{n \times q}$ with joint covariance matrix

$$
\begin{pmatrix}
\Sigma_x & \Sigma_{xy} \\
\Sigma_{yx} & \Sigma_y
\end{pmatrix},
$$

CCA seeks the coefficients $u, v$ such that the correlation of $Xu$ and $Yv$ is maximized. The classical CCA [23] can be formulated as

$$
\begin{align*}
\max_{u \in \mathbb{R}^p, v \in \mathbb{R}^q} & \quad u^\top X^\top Y v \\
\text{s.t.} & \quad u^\top X^\top X u = 1, \quad v^\top Y^\top Y v = 1,
\end{align*}
$$

(1.2)

where $X^\top Y, X^\top X, Y^\top Y$ are used to estimated the true parameters $\Sigma_{xy}, \Sigma_x, \Sigma_y$ after scaling.

However, PCA and CCA perform poorly and often lead to wrong findings when modeling with high-dimensional data. For example, when the dimension is proportional to the sample size such that $\lim_{n \to \infty} p/n = \gamma \in (0, 1)$ and the largest eigenvalue $\lambda_1 \leq \sqrt{\gamma}$, the leading sample principal eigenvector could be asymptotically orthogonal to the leading population principal eigenvector almost surely [3, 34, 36]. Sparse PCA and Sparse CCA are proposed as the more interpretable and reliable dimension reduction and feature extraction techniques for high-dimensional data. In what follows, we provide a brief overview of their methodological developments respectively.

Sparse PCA seeks sparse basis (loadings) of the subspace spanned by the data so that the obtained leading PCs are easier to interpret. Jolliffe et al. [25] proposed the SCoTLASS procedure by imposing $\ell_1$ norm on the loading vectors, which can be formulated as the following optimization problem for given data $X \in \mathbb{R}^{n \times p}$:

$$
\begin{align*}
\min_{A \in \mathbb{R}^{p \times r}} & \quad -\text{Tr}(A^\top X^\top X A) + \mu \|A\|_1 \\
\text{s.t.} & \quad A^\top A = I_r,
\end{align*}
$$

(1.3)

where $\text{Tr}(Z)$ denotes the trace of matrix $Z$, $\mu > 0$ is a weighting parameter, $\|A\|_1 = \sum_{i,j} |A_{ij}|$, and $I_r$ denotes the $r \times r$ identity matrix. Note that the original SCoTLASS model in [25] uses an $\ell_1$ constraint $\|A\|_1 \leq t$ instead of penalizing $\|A\|_1$ in the objective. The SCoTLASS model (1.3) is numerically very challenging. Algorithms for solving it have been very limited. As a result, a new formulation of Sparse PCA has been proposed by Zou et al. [60], and it has been the main focus in the literature on this topic. In [60], Zou et al. formulate Sparse PCA problem as the following ridge regression problem plus a lasso penalty:

$$
\begin{align*}
\min_{A \in \mathbb{R}^{p \times r}, B \in \mathbb{R}^{p \times r}} & \quad H(A, B) + \mu \sum_{j=1}^r \|B_j\|_2^2 + \sum_{j=1}^r \mu_{1,j} \|B_j\|_1 \\
\text{s.t.} & \quad A^\top A = I_r,
\end{align*}
$$

(1.4)

where

$$
H(A, B) := \sum_{i=1}^n \|x_i - AB^\top x_i\|_2^2,
$$

(1.5)

$x_i$ denotes the transpose of the $i$-th row vector of $X$, $B_j$ is the $j$-th column vector of $B$, and $\mu > 0$ and $\mu_{1,j} > 0$ are weighting parameters. However, it should be noted that (1.4) is indeed still numerically challenging. The combination of a nonsmooth objective and a manifold constraint makes the problem very difficult to solve. Zou et al. [60] proposed to solve it using an alternating minimization algorithm (AMA), which updates $A$ and $B$ alternatingly with the other variable fixed as the current iterate. A typical iteration of AMA is as follows

$$
\begin{align*}
A^{k+1} & := \arg\min_{A \in \mathbb{R}^{p \times r}} H(A, B^k), \text{s.t.}, A^\top A = I_r, \\
B^{k+1} & := \arg\min_{B \in \mathbb{R}^{p \times r}} H(A^{k+1}, B) + \mu \sum_{j=1}^r \|B_j\|_2^2 + \sum_{j=1}^r \mu_{1,j} \|B_j\|_1.
\end{align*}
$$

(1.6)
The $A$-subproblem in (1.6) is known as a Procrustes rotation problem and has a closed-form solution given by an SVD. The $B$-subproblem in (1.6) is a linear regression problem with an elastic-net regularizer, and it can be solved by many existing solvers such as elastic net [58], coordinate descent [18], and FISTA [3]. However, there is no convergence guarantee of AMA (1.6). Recently, some new algorithms are proposed in the literature that can solve (1.4) with guarantees of convergence to a stationary point. We will give a summary of some representative ones in the next section.

We need to point out that there are other ways to formulate Sparse PCA such as the ones in [13, 14, 15, 26, 29, 30, 33, 40, 47, 50, 53]. We refer interested readers to the recent survey paper [59] for more details on these works on Sparse PCA. In this paper, we focus on the formulation of (1.4) to estimate multiple principal components, which is a manifold optimization problem with nonsmooth objective function.

Sparse CCA [20, 35, 49, 50] is proposed to improve the interpretability of CCA, which can be formulated as

$$\begin{align*}
\min_{u \in \mathbb{R}^p, v \in \mathbb{R}^q} & -u^\top X^\top Yv + f(u) + g(v) \\
\text{s.t.} & \quad u^\top X^\top Xu = 1, \quad v^\top Y^\top Yv = 1,
\end{align*}$$

(1.7)

where $X \in \mathbb{R}^{n \times p}$, $Y \in \mathbb{R}^{n \times q}$, $f$ and $g$ are regularization terms promoting the sparsity of $u$ and $v$, and common choices for them include the $\ell_1$ norm for sparsity and the $\ell_{2,1}$ norm for group sparsity. When multiple canonical vectors are needed, one can consider the matrix counterpart of (1.7) which can be formulated as

$$\begin{align*}
\min_{A \in \mathbb{R}^{p \times r}, B \in \mathbb{R}^{q \times r}} & -\text{Tr}(A^\top X^\top YB) + f(A) + g(B) \\
\text{s.t.} & \quad A^\top X^\top XA = I_r, \quad B^\top Y^\top YB = I_r,
\end{align*}$$

(1.8)

where $r$ is the number of canonical vectors needed. From now on, we call (1.7) the single Sparse CCA model and (1.8) the multiple Sparse CCA model. Moreover, motivated by [10], in this paper we choose $f$ and $g$ to be the $\ell_{2,1}$ norm to promote the group sparsity of $A$ and $B$ in (1.8). Specifically, we choose $f(A) = \tau_1 \|A\|_{2,1}$, and $g(B) = \tau_2 \|B\|_{2,1}$, where the $\ell_{2,1}$ norm is defined as $\|A\|_{2,1} = \sum_{j=1}^p \|A_j\|_2$, and $A_j$ denotes the $j$-th row vector of matrix $A$, and $\tau_1 > 0$ and $\tau_2 > 0$ are weighting parameters. In this case, the multiple Sparse CCA (1.8) reduces to

$$\begin{align*}
\min_{A \in \mathbb{R}^{p \times r}, B \in \mathbb{R}^{q \times r}} & -\text{Tr}(A^\top X^\top YB) + \tau_1 \|A\|_{2,1} + \tau_2 \|B\|_{2,1} \\
\text{s.t.} & \quad A^\top X^\top XA = I_r, \quad B^\top Y^\top YB = I_r.
\end{align*}$$

(1.9)

Note that when $r = 1$, i.e., when the matrix reduces to a vector, the $\ell_{2,1}$ norm becomes the $\ell_1$ norm of the vector. That is, for vector $u \in \mathbb{R}^p$, $\|u\|_{2,1} = \|u\|_1$, and in this case, the vector Sparse CCA (1.7) reduces to

$$\begin{align*}
\min_{u \in \mathbb{R}^p, v \in \mathbb{R}^q} & -u^\top X^\top Yv + \tau_1 \|u\|_1 + \tau_2 \|v\|_1 \\
\text{s.t.} & \quad u^\top X^\top Xu = 1, \quad v^\top Y^\top Yv = 1.
\end{align*}$$

(1.10)

Note that both (1.9) and (1.10) are manifold optimization problems with nonsmooth objectives. Here we assume that both $X^\top X$ and $Y^\top Y$ are positive definite, and we will discuss later the modifications when they are not positive definite.

Manifold optimization recently draws a lot of research attention because of its success in a variety of important applications, including low-rank matrix completion [8, 46], phase retrieval [5, 43], phase synchronization [7, 28], blind deconvolution [24], and dictionary learning [12, 42]. Most existing algorithms for solving manifold optimization problems rely on the smoothness of the objective, see the recent monograph by Absil et al. [1]. Studies on manifold optimization problems with nonsmooth objective such as (1.4), (1.9), and (1.10) have been very limited. This

---

1R package available from https://cran.r-project.org/web/packages/elasticnet/
2R package available from https://cran.r-project.org/web/packages/glmnet/
urges us to study efficient algorithms that solve manifold optimization problems with nonsmooth objective, and this is the main focus of this paper.

The rest of this paper is organized as follows. We review existing methods for Sparse PCA and Sparse CCA in Section 2. We propose a unified alternating manifold proximal gradient method with provable convergence guarantees for solving both Sparse PCA and Sparse CCA in Section 3. The numerical performance is demonstrated in Section 4. We provide preliminaries on manifold optimization and details of the global convergence analysis of our proposed method in the Appendix.

2 Existing Methods

Before proceeding, we review existing methods for solving Sparse PCA (1.4) in Section 2.1 and for solving Sparse CCA (1.7) and (1.8) in Section 2.2.

2.1 Solving Sparse PCA

For Sparse PCA (1.4), other than the AMA algorithm suggested in the original paper [60], there exist some other efficient algorithms for solving this problem. We now give a brief review of these works. We first introduce two powerful optimization algorithms for solving nonconvex problems: proximal alternating minimization (PAM) algorithm [2] and proximal alternating linearization method (PALM) [6]. Surprisingly, it seems that these two methods have not been used to solve (1.4) yet. We now briefly describe how these two methods can be used to solve (1.4). PAM for (1.4) solves the following two subproblems in each iteration:

\[
A_{k+1} := \arg\min_A H(A, B_k) + \frac{1}{2t_1} \|A - A_k\|_F^2, \text{s.t.}, A^T A = I_r, \\
B_{k+1} := \arg\min_B H(A_{k+1}, B) + \mu \sum_{j=1}^r \|B_j\|_2^2 + \sum_{j=1}^r \mu_{1,j} \|B_j\|_1 + \frac{1}{2t_2} \|B - B_k\|_F^2,
\]

where \(t_1 > 0, t_2 > 0\) are stepsizes. Note that in each subproblem, PAM minimizes the objective function with respect to one variable by fixing the other, and a proximal term is added for the purpose of convergence guarantee. It is shown in [2] that the sequence of PAM converges to a critical point of (1.4) under the assumption that the objective function satisfies the Kurdyka-Lojasiewicz (KL) inequality\(^3\). We need to point out that the only difference between PAM (2.1) and the AMA (1.6) is the proximal terms, which together with the KL inequality helps establish the convergence result. Note that the \(A\)-subproblem in (2.1) corresponds to the reduced rank procrustes rotation and can be solved by an SVD. The \(B\)-subproblem in (2.1) is a Lasso type problem and can be solved efficiently by first-order methods such as FISTA or block coordinate descent. A better algorithm that avoids iterative solver for the subproblem is PALM, which linearizes the quadratic functions in the subproblems of (2.1). A typical iteration of PALM is:

\[
A_{k+1} := \arg\min_A \langle \nabla_A H(A_k, B_k), A \rangle + \frac{1}{2t_1} \|A - A_k\|_F^2, \text{s.t.}, A^T A = I_r, \\
B_{k+1} := \arg\min_B \langle \nabla_B H(A_{k+1}, B_k), B \rangle + \frac{1}{2t_2} \|B - B_k\|_F^2 + \mu \sum_{j=1}^r \|B_j\|_2^2 + \sum_{j=1}^r \mu_{1,j} \|B_j\|_1,
\]

where \(\nabla_A H\) and \(\nabla_B H\) denote the gradient of \(H\) with respect to \(A\) and \(B\), respectively. The two subproblems in (2.2) are easier to solve than the ones in (2.1) because they both admit closed-form

\(^3\)Without KL inequality, only subsequence convergence is obtained.
solutions. In particular, the solution of the $A$-subproblem in (2.2) corresponds to the projection onto the orthogonality constraint, which is given by an SVD; the solution of the $B$-subproblem in (2.2) is given by the $\ell_1$ soft-thresholding operation. It is shown in [6] that the sequence of PALM converges to a critical point of (1.4) under the assumption that the objective function satisfies the Kurdyka-Łojasiewicz inequality. Recently, Erichson et al. [16] proposed a projected gradient method based on variable projection (VP) for solving (1.4). Though the motivation of this algorithm is different, it can be viewed as a variant of PAM and PALM. Roughly speaking, the VP algorithm combines the $A$-subproblem (without the proximal term) in (2.1) and the $B$-subproblem in (2.2). That is, it updates the iterates as follows:

$$A_{k+1} := \arg\min_A H(A, B_k), \text{ s.t. } A^\top A = I_r,$$

$$B_{k+1} := \arg\min_B \langle \nabla_B H(A_{k+1}, B_k), B \rangle + \frac{1}{2\ell_2} \|B - B_k\|_F^2 + \mu \sum_{j=1}^r \|B_j\|_2^2 + \mu \sum_{j=1}^r \mu_1,j \|B_j\|_1.$$  \hspace{1cm} (2.3)

Note that the difference of PALM (2.2) and VP (2.3) lies in the $A$-subproblem. The $A$-subproblem linearizes the quadratic function $H(A, B_k)$ in (2.2) but not in (2.3). This does not affect much the performance of the algorithms because in this specific problem the $A$-subproblems correspond to an SVD in both algorithms. It is shown in [16] that VP (2.3) converges to a stationary point of (1.4). Another recent work that can solve (1.4) is the ManPG (manifold proximal gradient method) algorithm proposed by Chen et al. [11]. We will discuss it in more details later as it is closely related to the algorithm we propose in this paper. For other algorithms for solving Sparse PCA, we refer the interested readers to the recent survey paper [59] for more details.

### 2.2 Solving Sparse CCA

Chen et al. [10] proposed a CAPIT (standing for canonical correlation analysis via precision adjusted iterative thresholding) algorithm for solving the single Sparse CCA (1.10). The CAPIT algorithm alternates between an iterative thresholding step and a power method step, to deal with the sparsity regularization and orthogonality constraints respectively. The CoLaR (standing for Convex program with group-Lasso Refinement) method proposed by Gao et al. [19] targets to solve the multiple Sparse CCA (1.8). CoLaR is a two-stage algorithm. In the first stage, a convex relaxation of (1.8) based on the matrix lifting technique is solved. In the second stage, the solution obtained from the first stage is refined by solving a group Lasso type problem. In [49], Wiesel et al. proposed a greedy approach for solving (1.1) with cardinality constraints on $u$ and $v$. There is no convergence guarantee of this greedy approach due to the challenges posed by the combinatorial nature of the cardinality function. Recently, Suo et al. [44] proposed an alternating minimization algorithm (AMA) for solving the single Sparse CCA (1.10), which solves two subproblems in each iteration by solving (1.10) with respect to $u$ (resp. $v$) with $v$ (resp. $u$) fixed as $v^k$ (resp. $u^k$). The subproblems were then solved by a linearized ADMM (alternating direction method of multipliers) algorithm. We need to point out that none of these algorithms for Sparse CCA has a convergence guarantee. There exist some other methods for Sparse CCA (see, e.g., [20, 50]), but we omit their details here because they are not directly related to (1.7) and (1.8). We also point out that the PAM, PALM and VP algorithms discussed in Section 2.1 do not apply to Sparse CCA (1.7) and (1.8) because they all result in complicated subproblems. For instance, to apply PALM to (1.7), one needs to compute the proximal mapping of $f(u) + \iota(u^\top X^\top X u = 1)$, which does not admit a closed-form solution and is thus computationally expensive, where $\iota(\cdot)$ denotes the indicator function.
3 A Unified A-ManPG Algorithm

In this section, we give a unified treatment for solving Sparse PCA (1.4) and Sparse CCA (1.7) and (1.8), and introduce our alternating manifold proximal gradient algorithm (A-ManPG) for solving them. We first note that both Sparse PCA (1.4) and Sparse CCA (1.7) and (1.8) are special cases of the following problem:

$$
\min F(A, B) := H(A, B) + f(A) + g(B), \quad \text{s.t. } A \in \mathcal{M}_1, B \in \mathcal{M}_2,
$$

(3.1)

where $H(A, B)$ is a smooth function of $A, B$ with a Lipschitz continuous gradient, $f(\cdot)$ and $g(\cdot)$ are lower semi-continuous convex functions with relatively easy proximal mappings, and $\mathcal{M}_1, \mathcal{M}_2$ are two sub-manifolds embedded in the Euclidean space. The Sparse PCA (1.4) is in the form of (3.1) with $H(A, B) = \sum_{i=1}^{n} ||x_i - AB^\top x_i||_2^2$, $f(A) \equiv 0$, $g(B) = \mu \sum_{j=1}^{r} ||B_j||_2^2 + \sum_{j=1}^{r} \mu_{1,j} ||B_j||_1$, $\mathcal{M}_1 = \{A \mid A^\top A = I_r\}$ (the Stiefel manifold) and $\mathcal{M}_2 = \mathbb{R}^{p \times r}$. The single Sparse CCA (1.7) is in the form of (3.1) with $H(u, v) = -u^\top X^\top Y v$, $\mathcal{M}_1 = \{u \mid u^\top X^\top X u = 1\}$, $\mathcal{M}_2 = \{v \mid v^\top Y^\top Y v = 1\}$. The multiple Sparse CCA (1.8) is in the form of (3.1) with $H(A, B) = -\text{Tr}(A^\top X^\top Y B)$, $\mathcal{M}_1 = \{A \mid A^\top X^\top X A = I_r\}$, $\mathcal{M}_2 = \{B \mid B^\top Y^\top Y B = I_r\}$. Note that here in Sparse CCA (1.7) and (1.8) we assumed that both $X^\top X$ and $Y^\top Y$ are positive definite to guarantee that $\mathcal{M}_1$ and $\mathcal{M}_2$ are sub-manifolds. If they are not positive definite, we can always add a small perturbation to make them so. These manifolds used in Sparse CCA (1.7) and (1.8) are generalized Stiefel manifolds.

The ManPG algorithm proposed by Chen et al. [11] can be applied to solve (3.1). In each iteration, ManPG linearizes $H(A, B)$ and solves the following convex subproblem:

$$
\min_{D^A, D^B} \left\langle \begin{pmatrix} \nabla_{AH}(A_k, B_k) \\ \nabla_{BH}(A_k, B_k) \end{pmatrix}, \begin{pmatrix} D^A \\ D^B \end{pmatrix} \right\rangle + \frac{1}{2t_1} ||D^A||_F^2 + \frac{1}{2t_2} ||D^B||_F^2 + f(A_k + D^A) + g(B_k + D^B),
$$

s.t. $D^A \in T_{A_k, \mathcal{M}_1}$, $D^B \in T_{B_k, \mathcal{M}_2}$,

(3.2)

where $t_1 < 1/L_A$, $t_2 < 1/L_B$ and $L_A$ and $L_B$ are Lipschitz constants of $\nabla_A H(A, B_k)$ and $\nabla_B H(A_k, B)$ on tangent spaces $T_{A_k, \mathcal{M}_1}$ and $T_{B_k, \mathcal{M}_2}$, respectively. The ManPG algorithm (3.2) can be viewed as a Jacobi-type algorithm in this case, as it computes $D^A$ and $D^B$ in parallel. We found from our numerical experiments that the algorithm converges faster if $D^A$ and $D^B$ are computed in a Gauss-Seidel manner. This leads to the following updating scheme, which is the basis of our alternating manifold proximal gradient (A-ManPG) algorithm:

$$
D^A_k := \arg\min_{D^A} \left\langle \nabla_{AH}(A_k, B_k), D^A \right\rangle + f(A_k + D^A) + \frac{1}{2t_1} ||D^A||_F^2, \quad \text{s.t. } D^A \in T_{A_k, \mathcal{M}_1},
$$

(3.3)

$$
D^B_k := \arg\min_{D^B} \left\langle \nabla_{BH}(A_{k+1}, B_k), D^B \right\rangle + g(B_k + D^B) + \frac{1}{2t_2} ||D^B||_F^2, \quad \text{s.t. } D^B \in T_{B_k, \mathcal{M}_2},
$$

where $A_{k+1}$ is obtained via a retraction operation (see Algorithm [1]), $t_1 < 1/L_A$, $t_2 < 1/L_B$ and $L_A$ and $L_B$ are Lipschitz constants of $\nabla_A H(A, B_k)$ and $\nabla_B H(A_{k+1}, B)$ on tangent spaces $T_{A_k, \mathcal{M}_1}$ and $T_{B_k, \mathcal{M}_2}$, respectively. The Gauss-Seidel type algorithm A-ManPG usually performs much better than the Jacobi-type algorithm ManPG, because the Lipschitz constants are smaller and thus larger step sizes are allowed. The details of the A-ManPG algorithm are described in Algorithm [1].

Remark 3.1. Note that the iterates $A_k$ and $B_k$ are kept on the manifolds through the retraction operations $R_A$ and $R_B$. There exist many choices for the retraction operations, and in Algorithm...
Algorithm 1 Alternating Manifold Proximal Gradient Method (A-ManPG)

1: Input: Initial point \((A_0, B_0)\), parameters \(\delta \in (0, 1), \gamma \in (0, 1)\), step sizes \(t_1\) and \(t_2\).

2: for \(k = 0, 1, \ldots, \) do

3: Solve the \(A\)-subproblem in (3.3) to obtain \(D^A_k\).

4: Set \(\alpha_1 = 1\).

5: while \(F(R_{A_k}(\alpha_1 D^A_k), B_k) > F(A_k, B_k) - \delta \alpha_1 \|D^A_k\|_F^2\) do

6: \(\alpha_1 = \gamma \alpha_1\)

7: end while

8: Set \(A_{k+1} = R_{A_k}(\alpha_1 D^A_k)\).

9: Solve the \(B\)-subproblem in (3.3) to obtain \(D^B_k\).

10: Set \(\alpha_2 = 1\).

11: while \(F(A_{k+1}, R_{B_k}(\alpha_2 D^B_k)) > F(A_{k+1}, B_k) - \delta \alpha_2 \|D^B_k\|_F^2\) do

12: \(\alpha_2 = \gamma \alpha_2\)

13: end while

14: Set \(B_{k+1} = R_{B_k}(\alpha_2 D^B_k)\).

15: end for

We did not specify which ones to use. We discuss common retractions for Stiefel manifold and generalized Stiefel manifold in the Appendix. In our numerical experiments in Section 4, we chose polar decomposition as the retraction. Lines 4-7 and 9-13 in Algorithm 1 are backtracking line search procedures. These are necessary to guarantee that the objective function has a sufficient decrease in each iteration, which is needed for the convergence analysis (see the Appendix).

From Lemma C.5 (see Appendix), we know that \(D^A_k = 0\) and \(D^B_k = 0\) imply that \((A_k, B_k)\) is a stationary point for problem (3.1). As a result, we can define an \(\epsilon\)-stationary point of (3.1) as follows.

**Definition 3.2.** \((A_k, B_k)\) is called an \(\epsilon\)-stationary point of (3.1) if \(D^A_k\) and \(D^B_k\) returned by (3.3) satisfy \((\|D^A_k\|_F^2 + \|D^B_k\|_F^2) \leq \epsilon^2\).

We have the following convergence results for the A-ManPG algorithm (Algorithm 1).

**Theorem 3.3.** Any limit point of the sequence \(\{(A_k, B_k)\}\) generated by Algorithm 1 is a stationary point of problem (3.1). Furthermore, Algorithm 1 returns an \(\epsilon\)-stationary point \((A_k, B_k)\) in at most \((F(A_0, B_0) - F^*)/((\tilde{\beta}_1 + \tilde{\beta}_2)\epsilon^2)\) iterations, where \(F^*\) denotes a lower bound of the optimal value of (3.1), \(\tilde{\beta}_1 > 0\) and \(\tilde{\beta}_2 > 0\) are constants.

**Proof.** The proof is given in the Appendix.

### 3.1 Semi-Smooth Newton Method for the Subproblems

The main computational effort in each iteration of Algorithm 1 is to solve the two subproblems in (3.3). For Stiefel manifold and generalized Stiefel manifold, the two subproblems in (3.3) are both equality-constrained convex problems, given that both \(f\) and \(g\) are convex functions. Note that if \(f\) (resp. \(g\)) vanishes, the \(A\)-subproblem (resp. \(B\)-subproblem) becomes the projection onto the tangent space of \(M_1\) (resp. \(M_2\)), which reduces to Riemannian gradient step and can be easily done. Here we discuss the general case where \(f\) and \(g\) do not vanish. In this case, we found that a regularized semi-smooth Newton (SSN) method [51] is very suitable for solving this kind of problems. The notion of semi-smoothness was originally introduced by Mifflin [32] for real-valued functions and...
extended to vector-valued mappings by Qi and Sun [39]. A pioneer work on the SSN method was due to Solodov and Svaiter [41], where the authors proposed a globally convergent Newton’s method by exploiting the structure of monotonicity, and local superlinear rate was established under the conditions that generalized Jacobian is semi-smooth and non-singular at the global optimal point. The convergence rate is extended in [57] to the setting where the generalized Jacobian is not necessarily non-singular. Recently, SSN has received significant attention due to its success in solving structured convex problems to high accuracy. In particular, it has been successfully applied to solving SDP [53, 56], Lasso [27], nearest correlation matrix estimation [38], clustering [48], sparse inverse covariance selection [52], and composite convex minimization [51].

We now describe how to apply the regularized SSN method in [51] to solve the subproblems in (3.3). For brevity, we only focus on the A-subproblem with \( M_1 = \{ A \mid A^\top X^\top X A = I_r \} \) and \( f(A) = \tau_1 \| A \|_{2,1} \) as used in (1.9). For the ease of notation, we denote \( t = t_1, D = D^A, M := X^\top X, h(A) := H(A, B_k) \). In this case, the A-subproblem in (3.3) reduces to

\[
D_k := \text{argmin}_D \langle \nabla h(A_k), D \rangle + f(A_k + D) + \frac{1}{2t} \| D \|^2_F, \quad \text{s.t.} \quad D^\top MA_k + A_k^\top MD = 0. \tag{3.4}
\]

By associating a Lagrange multiplier \( \Lambda \) to the linear equality constraint, the Lagrangian function of (3.4) can be written as

\[
\mathcal{L}(D; \Lambda) = \langle \nabla h(A_k), D \rangle + \frac{1}{2t} \| D \|^2_F + f(A_k + D) - \langle D^\top MA_k + A_k^\top MD, \Lambda \rangle, \tag{3.5}
\]

and the Karush-Kuhn-Tucker (KKT) system of (3.4) is given by

\[
0 \in \partial_D \mathcal{L}(D; \Lambda), \quad D^\top MA_k + A_k^\top MD = 0. \tag{3.6}
\]

The first condition in (3.6) implies that \( D \) can be computed by

\[
D(\Lambda) = \text{prox}_f(B(\Lambda)) - A_k, \quad \text{with} \quad B(\Lambda) = A_k - t(\nabla h(A_k) - 2MA_k \Lambda), \tag{3.7}
\]

where \( \text{prox}_f(A) \) denotes the proximal mapping of function \( f \) at point \( A \). By substituting (3.7) into the second condition in (3.6), we obtain that \( \Lambda \) satisfies

\[
E(\Lambda) := D(\Lambda)^\top MA_k + A_k^\top MD(\Lambda) = 0, \tag{3.8}
\]

and thus the problem reduces to finding a root of a function \( E \). Since \( E \) is a monotone operator (see [11]) and the proximal mapping of the \( \ell_2 \) norm is semi-smooth [4], we can apply SSN to find the zero of \( E \). The SSN method requires to compute the generalized Jacobian of \( E \), and in the following we show how to compute it. We first derive the vectorization of \( E(\Lambda) \).

\[
\text{vec}(E(\Lambda)) = ((MA_k)^\top \otimes I_r)\text{vec}(D(\Lambda)^\top) + (I_r \otimes (MA_k)^\top)K_{rn}\text{vec}(D(\Lambda)^\top) \\
= (I_r + K_{rr})((MA_k)^\top \otimes I_r)[\text{prox}_f(B(\cdot))(\text{vec}((MA_k)^\top - t\nabla h(A_k)^\top)) \\
+ 2t((MA_k) \otimes I_r)\text{vec}(\Lambda)) - \text{vec}(X_k^\top),
\]

where \( K_{rn} \) and \( K_{rr} \) denote the commutation matrices. We define the following matrix

\[
G(\text{vec}(\Lambda)) = t((MA_k)^\top \otimes I_r)\mathcal{J}(y)|_{y = \text{vec}(B(\Lambda)^\top)((MA_k) \otimes I_r)},
\]

The definition is given in the Appendix. The proximal mapping of \( \ell_p(p \geq 1) \) norm is strongly semi-smooth [13, 45]. From [43], Prop. 2.26], if \( F : V \rightarrow \mathbb{R}^m \) is a piecewise \( C^1 \) (piecewise smooth) function, then \( F \) is semi-smooth. If \( F \) is a piecewise \( C^2 \) function, then \( F \) is strongly semi-smooth. It is known that proximal mappings of many interesting functions are piecewise linear or piecewise smooth.
where $\otimes$ denotes the Kronecker product, and $\mathcal{J}(y)$ is the generalized Jacobian of $\text{prox}_{tf(\cdot)}(y)$ which is defined as follows:

\[
\mathcal{J}(y)_{y=\text{vec}(B(\Lambda)\top)} = \begin{pmatrix}
\Delta_1 \\
\vdots \\
\Delta_p
\end{pmatrix},
\]

where the matrices $\Delta_j, j = 1, \ldots, p$ are defined as

\[
\Delta_j = \begin{cases}
I_r - \frac{r_{1j}}{\|b_j\|_2^2} (I_r - \frac{b_j b_j^\top}{\|b_j\|_2^2}), & \text{if } \|b_j\|_2 > t_r \\
\frac{b_j b_j^\top}{(t_r)^2} : \gamma \in [0, 1], & \text{if } \|b_j\|_2 = t_r \\
0, & \text{otherwise,}
\end{cases}
\]

where $b_j$ is the $j$-th column of matrix $B(\Lambda)^\top$. It is then easy to see that $G(\text{vec}(\Lambda))$ is positive-semidefinite\footnote{We say a matrix $A$ is positive semi-definite if $A + A^\top$ is positive semi-definite.}. From [21, Example 2.5], we know that $G(\text{vec}(\Lambda)) \xi = \partial \text{vec}(E(\text{vec}(\Lambda))) \xi, \forall \xi \in \mathbb{R}^r$. So, $G(\text{vec}(\Lambda))$ serves as an alternative of $\partial \text{vec}(E(\text{vec}(\Lambda)))$. It is known that the global convergence of regularized SSN is guaranteed if any element of $G(\text{vec}(\Lambda))$ is positive semi-definite [51]. For local convergence rate, one needs more conditions on $\partial \text{vec}(E(\text{vec}(\Lambda)))$. We refer to [51] for more details.

Note that since $\Lambda$ is a symmetric matrix, we can work with the lower triangular part of $\Lambda$ only and remove the duplicated entries in the upper triangular part. To do so, we use $\text{vec}(\Lambda)$ to denote the $\frac{1}{2}r(r+1)$-dimensional vector obtained from $\text{vec}(\Lambda)$ by eliminating all super-diagonal elements of $\Lambda$. It is known that there exists a unique $r^2 \times \frac{1}{2}r(r+1)$ matrix $U_r$, which is called the duplication matrix [31, Ch 3.8], such that $U_r \text{vec}(\Lambda) = \text{vec}(\Lambda)$. The Moore-Penrose inverse of $U_r$ is $U_r^+ = (U_r^\top U_r)^{-1} U_r^\top$ and it satisfies $U_r^+ \text{vec}(\Lambda) = \text{vec}(\Lambda)$. Note that both $U_r$ and $U_r^+$ have only $r^2$ nonzero elements. The alternative of generalized Jacobian of $E(U_r \text{vec}(\Lambda)))$ is given by

\[
G(\text{vec}(\Lambda)) = tU_r^+ G(\text{vec}(\Lambda)) U_r = 4tU_r^+ ((MA_k)^\top \otimes I_r)\mathcal{J}(y)_{y=\text{vec}(B(\Lambda)^\top)}((MA_k) \otimes I_r)U_r,
\]

where we used the identity $K_{rr} + I_{r,2} = 2U_r U_r^+$. Therefore, (3.9) can be simplified to

\[
G(\text{vec}(\Lambda)) = 4tU_r^+ ((MA_k)^\top \otimes I_r) \begin{pmatrix}
\Delta_1 \\
\vdots \\
\Delta_p
\end{pmatrix} ((MA_k) \otimes I_r)U_r \\
= 4tU_r^+ \begin{pmatrix}
\sum_{j=1}^p (MA_k)_{1j}^2 \Delta_j & \sum_{j=1}^p (MA_k)_{1j} (MA_k)_{2j} \Delta_j & \cdots & \sum_{j=1}^p (MA_k)_{1j} (MA_k)_{jr} \Delta_j \\
\sum_{j=1}^p (MA_k)_{2j} (MA_k)_{1j} \Delta_j & \sum_{j=1}^p (MA_k)_{2j}^2 \Delta_j & \cdots & \sum_{j=1}^p (MA_k)_{2j} (MA_k)_{jr} \Delta_j \\
\vdots & \vdots & \ddots & \vdots \\
\sum_{j=1}^p (MA_k)_{jr} (MA_k)_{1j} \Delta_j & \sum_{j=1}^p (MA_k)_{jr} (MA_k)_{2j} \Delta_j & \cdots & \sum_{j=1}^p (MA_k)_{jr}^2 \Delta_j
\end{pmatrix} U_r.
\]

(3.10)

The regularized SSN in [51] first computes the Newton’s direction $d_k$ by solving

\[
(G(\text{vec}(\Lambda_k)) + \eta I)d = -\text{vec}(E(\text{vec}(\Lambda_k)));
\]

(3.11)

where $\eta > 0$ is a regularization parameter. Note that $\eta$ is necessary here because $G(\text{vec}(\Lambda))$ could be singular if $\Delta_j = 0$ for some $j$. $\Lambda_k$ is then updated by

\[
\text{vec}(\Lambda_{k+1}) = \text{vec}(\Lambda_k) + d_k.
\]

The regularized SSN proposed in [51] combines some other techniques to make the algorithm more robust, but we omit the details here. We refer to [51] for more details on this algorithm.
4 Numerical Experiments

4.1 Sparse PCA

In this section, we apply our algorithm A-ManPG to solve Sparse PCA (1.4), and compare its performance with three existing methods: AMA [60], PALM [6] and VP [16]. The details of the parameter settings of these algorithms are given below.

- **AMA (1.6)**: FISTA [4] is used to solve the $B$-subproblem. Maximum iteration number is set to 1000.

- **PALM (2.2)**: $t_1 := 1$, $t_2 := 1/(2\lambda_{\text{max}}(X^TX))$. Maximum iteration number is set to 10000.

- **VP (2.3)**: $t_2 := 1/(2\lambda_{\text{max}}(X^TX))$. Maximum iteration number is set to 10000.

- **A-ManPG**: $t_1 = 100/p$, $t_2 := 1/(2\lambda_{\text{max}}(X^TX))$. Maximum iteration number is set to 10000.

The algorithms are terminated using the following criteria. First, we use PALM as a baseline, and we denote the objective function value in (1.4) as $F(A, B)\ldots$, $F(A, B) = H(A, B) + \mu \sum_{j=1}^r \|B_j\|_2 + \sum_{j=1}^r \mu_{1,j} \|B_j\|_1$. We terminate PALM when we find that $|F_{\text{PALM}}(A_{k+1}, B_{k+1}) - F_{\text{PALM}}(A_k, B_k)| < 10^{-5}$. (4.1)

We then terminate AMA, A-ManPG, and VP when their objective function value is smaller than $F_{\text{PALM}}$ and the change of their objective values in two consecutive iterations is less than $10^{-5}$.

We generate the data matrix $X$ in the following manner. First, the entries of $X$ are generated following standard normal distribution $\mathcal{N}(0, 1)$. The columns of $X$ are then centered so that the columns have zero mean and they are then scaled by dividing the largest $\ell_2$ norm of the columns.

We report the comparison results of the four algorithms in Tables 1 and 2 where $r = 6$ for all cases. In particular, Table 1 reports the results for $n < p$, and we tested $\mu = 1$ and $\mu = 10$, because it is suggested in [60] that $\mu$ should be relatively large in this case. Table 2 reports the results for $n > p$, and we set $\mu = 10^{-6}$, because it is suggested in [60] that $\mu$ should be sufficiently small in this case. In these tables, CPU times are in seconds, and 'sp' denotes the percentage of zero entries of matrix $B$.

From Tables 1 and 2 we see that the four algorithms generated solutions with similar objective function value $F(A, B)$ and similar sparsity 'sp'. In terms of CPU time, AMA is the slowest one, and the other three are comparable and are all much faster than AMA. This is due to the reason that AMA needs an iterative solver to solve the $B$-subproblem, which is time-consuming in practice.

4.2 Sparse CCA: Vector Case

In this section, we report the numerical results of A-ManPG for solving the single Sparse CCA (1.10), and compare its performance with a recent approach proposed by Suo et al. [44]: AMA+LADMM. More specifically, AMA+LADMM aims at solving the relaxation of (1.10) as follows

$$\begin{align*}
\min_{u \in \mathbb{R}^p, v \in \mathbb{R}^q} & -u^T X^T Y v + \tau_1 \|u\|_1 + \tau_2 \|v\|_1 \\
\text{s.t.} & u^T X^T X u \leq 1, \quad v^T Y^T Y v \leq 1.
\end{align*}$$

(4.2)

AMA+LADMM works in the following manner. In the $k$-th iteration, $v$ is fixed as $v_k$ and the following convex problem of $u$ is solved:

$$u_{k+1} := \arg\min_u -u^T X^T Y v_k + \tau_1 \|u\|_1, \quad \text{s.t.} \quad u^T X^T X u \leq 1.$$  

(4.3)
Table 1: Comparison of the algorithms for solving (1.4) with \( n < p \).

| \( \mu = 1 \) | \( \mu = 10 \) |
|----------------|----------------|
| \( F(A, B) \) | sp | CPU | iter | \( O(A, B) \) | sp | CPU | iter |
|----------------|----------------|
| \((n, p) = (100, 1000), \mu_{1,j} = 0.1, j = 1, \ldots, r\) | | | |
| AMA | -4.90778e+1 | 59.7 | 11.48 | 648 | -2.69714e+1 | 25.2 | 4.06 | 409 |
| A-ManPG | -4.90771e+1 | 59.4 | 0.36 | 1172 | -2.69716e+1 | 25.4 | 0.12 | 375 |
| PALM | -4.90776e+1 | 59.4 | 0.39 | 1394 | -2.69711e+1 | 25.3 | 0.15 | 518 |
| VP | -4.90770e+1 | 59.4 | 0.37 | 1335 | -2.69712e+1 | 25.2 | 0.13 | 453 |
| \((n, p) = (100, 1000), \mu_{1,j} = 0.2, j = 1, \ldots, r\) | | | |
| AMA | -4.16070e+1 | 76.5 | 7.28 | 433 | -2.16374e+1 | 42.6 | 7.28 | 433 |
| A-ManPG | -4.16057e+1 | 76.4 | 0.22 | 712 | -2.16371e+1 | 42.7 | 0.09 | 265 |
| PALM | -4.16055e+1 | 76.4 | 0.46 | 1044 | -2.16371e+1 | 42.6 | 0.12 | 343 |
| VP | -4.16056e+1 | 76.4 | 0.23 | 825 | -2.16372e+1 | 42.6 | 0.10 | 301 |
| \((n, p) = (500, 1000), \mu_{1,j} = 0.1, j = 1, \ldots, r\) | | | |
| AMA | -1.00053e+1 | 87.3 | 7.06 | 464 | -3.19608e+0 | 68.3 | 4.95 | 386 |
| A-ManPG | -9.98687e+0 | 86.9 | 0.24 | 486 | -3.18822e+0 | 68.3 | 0.13 | 265 |
| PALM | -9.98680e+0 | 87.1 | 0.24 | 533 | -3.18791e+0 | 68.0 | 0.22 | 495 |
| VP | -9.98688e+0 | 87.2 | 0.21 | 445 | -3.19602e+0 | 68.2 | 0.22 | 301 |
| \((n, p) = (500, 5000), \mu_{1,j} = 0.1, j = 1, \ldots, r\) | | | |
| AMA | -5.56171e+1 | 75.8 | 728.29 | 1000 | -3.18762e+1 | 40.5 | 452.19 | 1407 |
| A-ManPG | -5.56134e+1 | 75.7 | 8.90 | 1982 | -3.18753e+1 | 40.5 | 4.73 | 1021 |
| PALM | -5.56131e+1 | 75.8 | 10.77 | 2210 | -3.18750e+1 | 40.3 | 4.94 | 1023 |
| VP | -5.56132e+1 | 75.7 | 10.87 | 2147 | -3.18759e+1 | 40.5 | 7.17 | 1643 |
| \((n, p) = (1000, 5000), \mu_{1,j} = 0.1, j = 1, \ldots, r\) | | | |
| AMA | -1.94321e+1 | 93.9 | 398.16 | 666 | -7.41353e+0 | 77.1 | 317.83 | 841 |
| A-ManPG | -1.94306e+1 | 93.9 | 23.33 | 2346 | -7.41377e+0 | 77.4 | 10.38 | 1033 |
| PALM | -1.94306e+1 | 93.9 | 21.54 | 2104 | -7.41316e+0 | 77.1 | 16.54 | 565 |
| VP | -1.94306e+1 | 93.9 | 19.27 | 1899 | -7.41354e+0 | 77.1 | 11.33 | 1138 |
Table 2: Comparison of the algorithms for (1.4) with \( n > p \) and \( \mu = 10^{-6} \).

| \( (n, p) = (5000, 500) \), \( \mu_{1,j} = 0.01 \), \( j = 1, \ldots, r \) | \( F(A, B) \) | sp | CPU | iter |
|---|---|---|---|---|
| AMA | -8.54858e+0 | 31.9 | 10.62 | 700 |
| A-ManPG | -8.54859e+0 | 31.8 | 0.14 | 541 |
| PALM | -8.54739e+0 | 31.4 | 0.23 | 1077 |
| VP | -8.54841e+0 | 31.7 | 0.17 | 757 |

| \( (n, p) = (5000, 500) \), \( \mu_{1,j} = 0.05 \), \( j = 1, \ldots, r \) | \( F(A, B) \) | sp | CPU | iter |
|---|---|---|---|---|
| AMA | -6.45735e+0 | 90.1 | 6.29 | 428 |
| A-ManPG | -6.45546e+0 | 89.8 | 0.11 | 402 |
| PALM | -6.45532e+0 | 89.7 | 0.14 | 689 |
| VP | -6.45698e+0 | 90.0 | 0.12 | 571 |

| \( (n, p) = (5000, 2000) \), \( \mu_{1,j} = 0.01 \), \( j = 1, \ldots, r \) | \( F(A, B) \) | sp | CPU | iter |
|---|---|---|---|---|
| AMA | -1.29155e+1 | 37.8 | 133.04 | 539 |
| A-ManPG | -1.29151e+1 | 37.6 | 2.44 | 493 |
| PALM | -1.29142e+1 | 37.6 | 3.54 | 768 |
| VP | -1.29150e+1 | 37.5 | 2.93 | 640 |

| \( (n, p) = (5000, 2000) \), \( \mu_{1,j} = 0.05 \), \( j = 1, \ldots, r \) | \( F(A, B) \) | sp | CPU | iter |
|---|---|---|---|---|
| AMA | -8.83497e+0 | 89.4 | 44.30 | 636 |
| A-ManPG | -8.83440e+0 | 89.4 | 3.87 | 842 |
| PALM | -8.83437e+0 | 89.3 | 4.36 | 977 |
| VP | -8.83452e+0 | 89.3 | 3.48 | 773 |

| \( (n, p) = (8000, 1000) \), \( \mu_{1,j} = 0.01 \), \( j = 1, \ldots, r \) | \( F(A, B) \) | sp | CPU | iter |
|---|---|---|---|---|
| AMA | -9.04522e+0 | 37.9 | 23.53 | 325 |
| A-ManPG | -9.04477e+0 | 37.6 | 0.22 | 276 |
| PALM | -9.04471e+0 | 37.7 | 0.31 | 507 |
| VP | -9.04511e+0 | 37.8 | 0.24 | 360 |

| \( (n, p) = (8000, 1000) \), \( \mu_{1,j} = 0.05 \), \( j = 1, \ldots, r \) | \( F(A, B) \) | sp | CPU | iter |
|---|---|---|---|---|
| AMA | -6.59097e+0 | 95.6 | 44.30 | 636 |
| A-ManPG | -6.58996e+0 | 95.7 | 0.78 | 897 |
| PALM | -6.58995e+0 | 95.7 | 0.82 | 1486 |
| VP | -6.60764e+0 | 95.9 | 1.22 | 1907 |

| \( (n, p) = (8000, 2000) \), \( \mu_{1,j} = 0.01 \), \( j = 1, \ldots, r \) | \( F(A, B) \) | sp | CPU | iter |
|---|---|---|---|---|
| AMA | -1.07975e+1 | 40.3 | 116.65 | 388 |
| A-ManPG | -1.07975e+1 | 40.2 | 2.19 | 363 |
| PALM | -1.07966e+1 | 40.2 | 3.00 | 550 |
| VP | -1.07972e+1 | 40.4 | 2.70 | 437 |

| \( (n, p) = (8000, 2000) \), \( \mu_{1,j} = 0.05 \), \( j = 1, \ldots, r \) | \( F(A, B) \) | sp | CPU | iter |
|---|---|---|---|---|
| AMA | -7.32162e+0 | 95.3 | 167.36 | 597 |
| A-ManPG | -7.31837e+0 | 95.0 | 3.34 | 578 |
| PALM | -7.30781e+0 | 95.0 | 3.97 | 780 |
| VP | -7.30822e+0 | 95.0 | 3.29 | 606 |
Then, \( u \) is fixed as \( u_{k+1} \) and the following convex problem of \( v \) is solved
\[
v_{k+1} := \arg\min_v -u_{k+1}^T X^T Y v + \tau_2 \|v\|_2, \quad \text{s.t.} \quad v^T Y^T Y v \leq 1. \tag{4.4}
\]
The linearized ADMM (LADMM) is used to solve the two convex subproblems \((4.3)\) and \((4.4)\).

We generate the data following the same manner as in \([44]\). Specifically, two data sets \( X \in \mathbb{R}^{n \times p} \) and \( Y \in \mathbb{R}^{n \times q} \) are generated from the following model:
\[
\begin{pmatrix} x \\ y \end{pmatrix} \sim \mathcal{N} \left( \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \Sigma_x & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_y \end{pmatrix} \right), \tag{4.5}
\]
where \( \Sigma_{xy} = \rho \Sigma_x \hat{u} \hat{v}^T \Sigma_y \), \( \hat{u} \) and \( \hat{v} \) are the true canonical vectors, and \( \rho \) is the true canonical correlation. In our numerical tests, \( \hat{u} \) and \( \hat{v} \) are generated randomly such that they both have 5 non-zero entries and the nonzero coordinates are set at the \( \{1, 6, 11, 16, 21\} \)-th coordinates. The nonzero entries are obtained from normalizing (with respect to \( \Sigma_x \) and \( \Sigma_y \)) random numbers drawn from the uniform distribution on the finite set \( \{-2, -1, 0, 1, 2\} \). We set \( \rho = 0.9 \) in all tests. We tested three different ways to generate the covariance matrices \( \Sigma_x \) and \( \Sigma_y \):

- Identity matrices: \( \Sigma_x = I_p \), \( \Sigma_y = I_q \).
- Toeplitz matrices: \( [\Sigma_x]_{ij} = 0.9^{i-j} \) and \( [\Sigma_y]_{ij} = 0.9^{|i-j|} \).
- Sparse inverse matrices: \( [\Sigma_x]_{ij} = \sigma_{ij}^0 / \sqrt{\sigma_{ii}^0 \sigma_{jj}^0} \), where \( \Sigma^0 = (\sigma_{ij}^0) = \Omega^{-1} \) and \( \Omega_{ij} = t_{i=j} + 0.5 \times 1_{|i-j|=1} + 0.4 \times 1_{|i-j|=2} \).

\( \Sigma_y \) is generated in the same way. The matrices \( X \) and \( Y \) are both divided by \( \sqrt{n-1} \) such that \( X^T Y \) is the estimated covariance matrix. Note that if \( n < p \) or \( n < q \), the covariance matrix \( X^T X \) or \( Y^T Y \) is not positive definite. In this case, we replace \( X^T X \) by \( (1-\alpha)X^T X + \alpha I_p \) and \( Y^T Y \) by \( (1-\alpha)Y^T Y + \alpha I_q \) in the constraints of \((1.10)\), so that we can still keep them as manifold constraints. In our experiments, we chose \( \alpha = 10^{-4} \). The same as \([44]\), we define two loss functions 'lossu' and 'lossv' to measure the distance between the ground truth \( (\hat{u}, \hat{v}) \) and estimation \( (u, v) \):
\[
\text{lossu} = 2(1 - |u^T u|), \quad \text{lossv} = 2(1 - |v^T v|),
\]
where \( (u, v) \) is the iterate returned by the algorithm. Moreover, the following procedure for initialization suggested in \([44]\) is adopted. First, we truncate the matrix \( X^T Y \) by soft-thresholding its small elements to be 0 and denote the new matrix \( S_{xy} \). More specifically, we set the entries of \( S_{xy} \) to zeros if their magnitudes are smaller than the largest magnitude of the diagonal elements. Secondly, we compute the singular vectors \( u_0 \) and \( v_0 \) corresponding to the largest singular value of \( S_{xy} \) and then normalize them using \( u_0 := u_0 / \sqrt{u_0^T X^T X u_0} \) and \( v_0 := v_0 / \sqrt{v_0^T Y^T Y v_0} \) as initialization of \( u \) and \( v \). We set \( \tau_1 = \tau_2 = \frac{1}{2} b / \sqrt{\log(p+q)/n} \) in \((1.10)\) where \( b \) was set to \( b = \{1, 1.2, 1.4, 1.6\} \). We report the best result among all the candidates. For each \( b \), we solved \((1.10)\) by A-ManPG with \( \delta = 10^{-4}, \gamma = 0.5, t_1 = t_2 = 1 \). The A-ManPG was stopped if \( \max \{ \|D_k^A\|_F^2, \|D_k^B\|_F^2 \} \leq 10^{-8} \) and the regularized SSN was stopped if \( \|E(A_k)\|_F \leq 10^{-5} \) in \((3.8)\). For AMA+LADMM, we set the stopping criteria of LADMM as \( \|u_j - u_{j-1}\| \leq 10^{-3} \) and \( \|v_j - v_{j-1}\| \leq 10^{-3} \), where \( u_j \) and \( v_j \) are iterates in LADMM. We set the stopping criteria of AMA as \( \|u_k - u_{k-1}\| \leq 10^{-3} \) and \( \|v_k - v_{k-1}\| \leq 10^{-3} \), where \( u_k \) and \( v_k \) are iterates in AMA.

We report the numerical results in Table \(3\) where 'nu' and 'nv' denote the number of nonzeros in \( u \) and \( v \) after setting their entries whose magnitudes are smaller than \( 10^{-4} \) to 0, and \( \rho \) denotes
the canonical correlation computed from the solution returned by the algorithms. All reported values in Table 3 are the medians from 20 repetitions. From Table 3 we see that A-ManPG and AMA+LADMM achieve similar loss function values 'lossu' and 'lossv', but A-ManPG is usually faster than AMA+LADMM, and for some cases, it is even two to three times faster. More importantly, AMA+LADMM lacks convergence analysis, but A-ManPG is guaranteed to converge to a stationary point (see the Appendix). Furthermore, AMA+LADMM is very time consuming for the multiple Sparse CCA (1.9), but A-ManPG is suitable for (1.9) as we show in the next section.

Table 3: Comparison of A-ManPG and AMA+LADMM \[44\] for solving single sparse CCA \[1.10\].

| (n, p, q) | cpu | lossu | lossv | \(\rho\) | mu | nv |
|---------|-----|-------|-------|------|----|----|
| Identity matrix |
| 500,800,800 | 0.265 | 3.955e-3 | 4.635e-3 | 0.900 | 4 | 4.5 |
| 1000,800,800 | 0.395 | 2.477e-3 | 2.350e-3 | 0.900 | 4 | 4.5 |
| 500,1600,1600 | 0.990 | 6.071e-3 | 4.247e-3 | 0.900 | 5 | 4.5 |
| 1000,1600,1600 | 1.244 | 1.351e-3 | 2.081e-3 | 0.900 | 5 | 5 |
| Toeplitz matrix |
| 500,800,800 | 0.279 | 3.569e-3 | 5.570e-3 | 0.902 | 7 | 5.5 |
| 1000,800,800 | 0.395 | 2.152e-3 | 2.165e-3 | 0.902 | 5 | 5 |
| 500,1600,1600 | 0.955 | 5.802e-3 | 4.758e-3 | 0.908 | 4 | 4.5 |
| 1000,1600,1600 | 1.172 | 1.913e-3 | 1.602e-3 | 0.901 | 5 | 5 |
| Sparse inverse matrix |
| 500,800,800 | 0.527 | 7.749e-3 | 1.248e-2 | 0.896 | 7 | 6.5 |
| 1000,800,800 | 0.618 | 5.920e-3 | 4.631e-3 | 0.898 | 5 | 5 |
| 500,1600,1600 | 1.589 | 9.624e-3 | 1.052e-2 | 0.889 | 5 | 5 |
| 1000,1600,1600 | 1.951 | 2.799e-3 | 3.812e-3 | 0.900 | 6 | 6 |

4.3 Sparse CCA: Matrix Case

In this section, we apply A-ManPG to solve the multiple Sparse CCA (1.9) and compare its performance with CoLaR method proposed by Gao et al. in \[19\]. CoLaR is a two-stage method based on convex relaxations. In the first stage, CoLaR solves the following convex program

\[
\min_F \ - \text{Tr}(F^TX^TY) + \tau \|F\|_1, \quad \text{s.t.} \quad \|(X^TX)^{\frac{1}{2}}F(Y^TY)^{\frac{1}{2}}\|_2 \leq 1, \|(X^TX)^{\frac{1}{2}}F(Y^TY)^{\frac{1}{2}}\|_* \leq r, \tag{4.6}
\]

where \(\| \cdot \|_2\) and \(\| \cdot \|_*\) respectively denote the operator norm and nuclear norm, \(F\) is the surrogate of \(AB^\top\) and the constraint is the convex hull of \(\{AB^\top : A \in \text{St}(p, r), B \in \text{St}(q, r)\}\). Here \(\text{St}(p, r)\) denotes the Stiefel manifold with matrix size \(p \times r\). Gao et al. \[19\] suggest to use ADMM to solve (4.6). The main purpose of the first stage is to provide a good initialization for the second stage. Assume solution to (4.6) is \(\hat{F}\), and \(A_0\) and \(B_0\) are matrices whose column vectors are respectively the top \(r\) left and right singular vectors of \(\hat{F}\). A refinement of \(A_0\) is adopted in the second stage, in which the following group Lasso problem is solved:

\[
\min_L \text{Tr}(L^T(X^TX)L) - 2\text{Tr}(L^TX^TYB_0) + \tau' \sum_{j=1}^{p} \|L_j\|. \tag{4.7}
\]

A similar strategy for \(B_0\) is taken. Suppose the solutions to the group Lasso problems are \(A_1\) and \(B_1\), the final estimations are normalized as \(A = A_1(A_1^TX^XA_1)^{-\frac{1}{2}}\) and \(B = B_1(B_1^TY^YB_1)^{-\frac{1}{2}}\). We found that the efficiency of CoLaR highly relies on the first stage. Since a good initialization is
crucial for the nonconvex problem, we also use the solution returned from the first stage \((4.6)\) as the initial point for our A-ManPG algorithm. We follow the same settings of all numerical tests as suggested in \([10]\). We tested three different ways to generate the covariance matrix \(\Sigma_x = \Sigma_y\) with \(p = q\).

- **Identity matrices:** \(\Sigma_x = \Sigma_y = I_p\).
- **Toeplitz matrices:** \(|\Sigma_x|_{ij} = |\Sigma_y|_{ij} = 0.3^{|i-j|}\).
- **Sparse inverse matrices:** \(|\Sigma_x|_{ij} = |\Sigma_y|_{ij} = \sigma_0^2/\sqrt{\sigma_0^2 \sigma_{jj}^2}\), where \(\Sigma^0 = (\sigma_{ij}^0) = \Omega^{-1}\) and
  \[\Omega_{ij} = \iota_{(i=j)} + 0.5 \times \iota_{|i-j|=1} + 0.4 \times \iota_{|i-j|=2}.
\]

In all tests, we chose \(r = 2\) and generated \(\Sigma_{xy} = \Sigma_x U \Lambda V^\top \Sigma_y\), where \(\Lambda \in \mathbb{R}^{r \times r}\) is a diagonal matrix with diagonal entries \(\Lambda_{11} = 0.9\) and \(\Lambda_{22} = 0.8\). The nonzero rows of both \(U\) and \(V\) are set at the \(\{1, 6, 11, 16, 21\}\)-th rows. The values at the nonzero coordinates are obtained from normalizing (with respect to \(\Sigma_x\) and \(\Sigma_y\)) random numbers drawn from the uniform distribution on the finite set \(\{-2, -1, 0, 1, 2\}\). The two datasets \(X \in \mathbb{R}^{n \times p}\) and \(Y \in \mathbb{R}^{n \times q}\) are then generated from \((4.5)\). The matrices \(X\) and \(Y\) are both divided by \(\sqrt{n-1}\) such that \(X^\top Y\) is the estimated covariance matrix. The loss between the estimation \(A\) and the ground truth \(U\) is measured by the subspace distance loss \(u = \|P_U - P_A\|_F^2\), where \(P_U\) denotes the projection matrix onto the column space of \(U\). Similarly, the loss for \(B\) and \(V\) is measured as loss \(v = \|P_V - P_B\|_F^2\).

The codes of CoLaR were downloaded from the authors’ webpage.\(^6\) We used all default settings of their codes. In particular, ADMM is used to solve the first stage problem \((4.5)\) and it is terminated when it does not make much progress, or it reaches the maximum iteration number 100. For our A-ManPG, we run only one iteration of ADMM for \((4.6)\) and use the returned solution as the initial point of A-ManPG, because we found that this already generates a very good solution for A-ManPG. To be fair, we also compare the same case for CoLaR where only one iteration of ADMM is used for \((4.6)\). The parameter \(\tau\) in \((4.6)\) is set to \(\tau = 0.55 \sqrt{\log(p+q)/n}\). We set \(\tau_1 = \tau_2 = \frac{1}{2} b \sqrt{\log(p+q)/n}\) in \((4.9)\) and \(\tau' = b\) in \((4.7)\) where \(b\) was set to \(b = \{0.8, 1, 1.2, 1.4, 1.6\}\). For each \(b\), we solved \((4.9)\) by A-ManPG with \(\delta = 10^{-4}, \gamma = 0.5, t_1 = t_2 = 1\). The A-ManPG was stopped if \(\max\{\|D_k^A\|_F^2, \|D_k^R\|_F^2\} \leq 10^{-8}\) and the regularized SSN was stopped if \(\|E(\Lambda_k)\|_F \leq 10^{-5}\) in \((3.8)\).

We report the numerical results in Tables \(\{4|7\}\) where CPU times are in seconds, ‘nA’ and ‘nB’ denote the number of nonzeros of \(A\) and \(B\) respectively, after truncating the entries whose magnitudes are smaller than \(10^{-4}\) to zeros. \(\rho_1\) and \(\rho_2\) are the two canonical correlations and they should be close to 0.9 and 0.8, respectively. All reported values in Tables \(\{4|7\}\) are the medians from 20 repetitions. More specifically, Table \(\{4\}\) reports the results obtained from the first stage where ADMM was used to solve \((4.6)\). ‘Init-1’ indicates that we only run one iteration of ADMM, and ‘Init-100’ indicates the case where we run ADMM until it does not make much progress, or the maximum iteration number 100 is reached. From Table \(\{4\}\) we see that solving the first stage problem by running ADMM for 100 iterations indeed improves the two losses significantly. Tables \(\{5|7\}\) report the results for the three different types of covariance matrices. A-ManPG-1 and CoLaR-1 are the cases where we only run one iteration of ADMM for the first stage, and CoLaR-100 is the case where the first stage problem \((4.6)\) is solved more accurately by ADMM, as discussed above. We observed that running more iteration of ADMM in the first stage does not help much for A-ManPG; we thus only report the results of A-ManPG-1. From Tables \(\{5|7\}\) we see that CoLaR-100 gives much better

\(^6\)http://www-stat.wharton.upenn.edu/~zongming/research.html
results than CoLaR-1 in terms of the two losses 'lossu' and 'lossv', especially when the sample size is relatively small compared with the matrix sizes. Moreover, we see that A-ManPG-1 outperforms both CoLaR-1 and CoLaR-100 significantly. In particular, A-ManPG-1 generates comparable and very often better solutions than CoLaR-1 and CoLaR-100 in terms of solution sparsity and losses 'lossu' and 'lossv'. Furthermore, A-ManPG-1 is usually faster than CoLaR-1 and much faster than CoLaR-100.

Table 4: Losses returned from the first stage problem (4.6).

| (n, p, q)     | lossu | lossv | lossu | lossv |
|---------------|-------|-------|-------|-------|
| Identity matrix |       |       |       |       |
| 200,300,300   | 0.304 | 0.374 | 0.107 | 0.124 |
| 500,300,300   | 0.114 | 0.103 | 0.050 | 0.037 |
| 200,600,600   | 0.394 | 0.393 | 0.146 | 0.116 |
| 500,600,600   | 0.137 | 0.139 | 0.048 | 0.035 |
| Toeplitz matrix |       |       |       |       |
| 200,300,300   | 0.318 | 0.375 | 0.120 | 0.107 |
| 500,300,300   | 0.126 | 0.090 | 0.038 | 0.028 |
| 200,600,600   | 0.427 | 0.401 | 0.103 | 0.110 |
| 500,600,600   | 0.101 | 0.133 | 0.028 | 0.039 |
| Sparse inverse matrix |       |       |       |       |
| 200,300,300   | 0.609 | 0.658 | 0.253 | 0.281 |
| 500,300,300   | 0.231 | 0.191 | 0.098 | 0.085 |
| 200,600,600   | 0.837 | 0.749 | 0.328 | 0.233 |
| 500,600,600   | 0.311 | 0.318 | 0.102 | 0.118 |

5 Conclusion

In this paper, we proposed an efficient algorithm for solving two important and numerically challenging optimization problems arising from statistics: sparse PCA and sparse CCA. These two problems are challenging to solve because they are manifold optimization problems with nonsmooth objectives, a topic that is still underdeveloped in optimization. We proposed an alternating manifold proximal gradient method (A-ManPG) to solve these two problems. Convergence and convergence rate to a stationary point of the proposed algorithm are established. Numerical results on statistical data demonstrate that A-ManPG is comparable to existing algorithms for solving sparse PCA, and is significantly better than existing algorithms for solving sparse CCA.

A Preliminaries on Manifold Optimization

We now introduce some preliminaries on manifold optimization. An important concept in manifold optimization is retraction, which is defined as follows.

Definition A.1. [1, Definition 4.1.1] A retraction on a differentiable manifold $\mathcal{M}$ is a smooth mapping from the tangent bundle $T\mathcal{M}$ onto $\mathcal{M}$ satisfying the following two conditions, where $R_X$ denotes the restriction of $R$ onto $T_X\mathcal{M}$.

1. $R_X(0) = X, \forall X \in \mathcal{M}$, where 0 denotes the zero element of $T_X\mathcal{M}$. 

16
Table 5: Comparison of A-ManPG and CoLaR for multiple sparse CCA \(^{(1.9)}\). Covariance matrix: identity matrix

|         | A-ManPG-1 | CoLaR-1 | CoLaR-100 |
|---------|-----------|---------|-----------|
| \(b\)   | 0.8       | 1       | 1.2       | 1.4       | 1.6   | 0.8 | 1 | 1.2 | 1.4 | 1.6 |
| CPU     | 0.387     | 0.320   | 0.288     | 0.327     | 0.274 | 0.763| 0.760| 0.725| 0.667| 0.619 | 5.071 | 5.020 | 4.844 | 4.724 | 4.663 |
| lossu   | 0.064     | 0.043   | 0.038     | 0.036     | 0.045 | 0.094| 0.062| 0.049| 0.046| 0.051 | 0.081 | 0.051 | 0.038 | 0.041 | 0.047 |
| losav   | 0.075     | 0.053   | 0.044     | 0.047     | 0.058 | 0.110| 0.079| 0.072| 0.080| 0.094 | 0.096 | 0.063 | 0.061 | 0.059 | 0.072 |
| nA      | 44        | 23.5    | 15.5      | 10.1      | 10    | 64   | 32.5 | 18   | 12   | 10    | 64    | 30   | 16.5 | 10   | 10   |
| nB      | 45.5      | 24.5    | 16.0      | 10.1      | 10    | 64   | 32.5 | 18   | 12   | 11    | 63    | 32   | 19   | 12   | 10   |
| \(\rho_1\) | 0.919    | 0.907   | 0.900     | 0.897     | 0.894 | 0.925| 0.910| 0.900| 0.895| 0.893 | 0.925 | 0.911| 0.900 | 0.897 | 0.895 |
| \(\rho_2\) | 0.863    | 0.833   | 0.822     | 0.833     | 0.811 | 0.877| 0.840| 0.822| 0.813| 0.804 | 0.879 | 0.841| 0.821 | 0.815 | 0.810 |

\[(n, p, q) = (200, 400, 300)\]

|         | A-ManPG-1 | CoLaR-1 | CoLaR-100 |
|---------|-----------|---------|-----------|
| \(b\)   | 0.369     | 0.275   | 0.264     | 0.263     | 0.265 | 0.693| 0.680| 0.612| 0.524| 0.431 | 2.995 | 2.924 | 2.780 | 2.653 | 2.647 |
| lossu   | 0.031     | 0.021   | 0.017     | 0.018     | 0.020 | 0.032| 0.018| 0.017| 0.018| 0.019 | 0.032 | 0.018| 0.015| 0.017 | 0.020 |
| losav   | 0.031     | 0.019   | 0.018     | 0.019     | 0.021 | 0.038| 0.023| 0.022| 0.023| 0.022 | 0.037 | 0.022| 0.019| 0.019 | 0.021 |
| nA      | 62        | 25.1    | 14.0      | 10.1      | 10    | 67   | 28.5 | 16   | 10.5| 10    | 63    | 28.5 | 15   | 10   | 10   |
| nB      | 58        | 30.1    | 16.5      | 11.1      | 10    | 64.5 | 31.5 | 17.5 | 11   | 10    | 69    | 31.5 | 18   | 12   | 10   |
| \(\rho_1\) | 0.907    | 0.901   | 0.898     | 0.897     | 0.897 | 0.906| 0.901| 0.898| 0.897| 0.896 | 0.907 | 0.901| 0.898| 0.897 | 0.896 |
| \(\rho_2\) | 0.833    | 0.816   | 0.808     | 0.804     | 0.803 | 0.838| 0.817| 0.808| 0.804| 0.803 | 0.838 | 0.818| 0.808| 0.804 | 0.802 |

\[(n, p, q) = (200, 600, 600)\]

|         | A-ManPG-1 | CoLaR-1 | CoLaR-100 |
|---------|-----------|---------|-----------|
| \(b\)   | 1.329     | 1.131   | 1.062     | 1.021     | 0.992 | 1.441| 1.373| 1.321| 1.229| 1.217 | 63.371| 63.236| 63.092| 62.835| 62.706 |
| lossu   | 0.010     | 0.068   | 0.056     | 0.062     | 0.070 | 0.182| 0.117| 0.094| 0.097| 0.103 | 0.141 | 0.095| 0.071| 0.071 | 0.085 |
| losav   | 0.091     | 0.069   | 0.059     | 0.057     | 0.073 | 0.162| 0.115| 0.093| 0.093| 0.091 | 0.127 | 0.085| 0.065| 0.066 | 0.081 |
| nA      | 54.5      | 31.1    | 18.0      | 12.1      | 10    | 91.5 | 49.5 | 23   | 16   | 12    | 78    | 37   | 18   | 12   | 10   |
| nB      | 55        | 29.5    | 18.0      | 13.1      | 10   | 100  | 49.5 | 25.5 | 15   | 12    | 78    | 35   | 21   | 13.5 | 10   |
| \(\rho_1\) | 0.926    | 0.915   | 0.910     | 0.906     | 0.903 | 0.934| 0.918| 0.907| 0.904| 0.903 | 0.932 | 0.912| 0.905| 0.903 | 0.902 |
| \(\rho_2\) | 0.879    | 0.843   | 0.821     | 0.804     | 0.798 | 0.903| 0.858| 0.823| 0.808| 0.795 | 0.904 | 0.852| 0.823| 0.805 | 0.799 |

\[(n, p, q) = (200, 600, 600)\]
Table 6: Comparison of A-ManPG and CoLaR for multiple sparse CCA [19]. Covariance matrix: Topelitz matrix

| b   | 0.8 | 1   | 1.2  | 1.4  | 1.6  | 0.8 | 1   | 1.2  | 1.4  | 1.6  |
|-----|-----|-----|------|------|------|-----|-----|------|------|------|
| CPU | 0.380 | 0.327 | 0.292 | 0.283 | 0.282 | 0.791 | 0.761 | 0.729 | 0.666 | 0.622 |
| lossu | 0.069 | 0.045 | 0.043 | 0.049 | 0.064 | 0.103 | 0.079 | 0.069 | 0.070 | 0.085 |
| lossv | 0.075 | 0.057 | 0.046 | 0.050 | 0.060 | 0.116 | 0.079 | 0.065 | 0.067 | 0.075 |
| nA | 43 | 26 | 15.5 | 12 | 10 | 61.5 | 31 | 18.5 | 12 | 10 |
| nB | 44.5 | 27 | 16 | 12 | 10 | 64.5 | 36 | 20 | 14 | 12 |
| $\rho_1$ | 0.921 | 0.911 | 0.906 | 0.902 | 0.898 | 0.925 | 0.912 | 0.905 | 0.902 | 0.900 |
| $\rho_2$ | 0.864 | 0.835 | 0.818 | 0.803 | 0.794 | 0.869 | 0.839 | 0.818 | 0.803 | 0.797 |

| b   | 0.8 | 1   | 1.2  | 1.4  | 1.6  | 0.8 | 1   | 1.2  | 1.4  | 1.6  |
|-----|-----|-----|------|------|------|-----|-----|------|------|------|
| CPU | 0.319 | 0.287 | 0.266 | 0.261 | 0.260 | 0.707 | 0.667 | 0.646 | 0.492 | 0.431 |
| lossu | 0.025 | 0.015 | 0.010 | 0.010 | 0.010 | 0.029 | 0.017 | 0.012 | 0.013 | 0.014 |
| lossv | 0.027 | 0.016 | 0.012 | 0.010 | 0.012 | 0.031 | 0.015 | 0.012 | 0.011 | 0.013 |
| nA | 54 | 27 | 14.5 | 10 | 10 | 60.5 | 25.5 | 15 | 12 | 10 |
| nB | 56.5 | 28 | 16 | 10 | 10 | 63 | 31 | 18 | 10 | 10 |
| $\rho_1$ | 0.905 | 0.899 | 0.896 | 0.896 | 0.895 | 0.906 | 0.900 | 0.897 | 0.896 | 0.896 |
| $\rho_2$ | 0.835 | 0.819 | 0.810 | 0.807 | 0.806 | 0.838 | 0.820 | 0.810 | 0.807 | 0.806 |

| b   | 0.8 | 1   | 1.2  | 1.4  | 1.6  | 0.8 | 1   | 1.2  | 1.4  | 1.6  |
|-----|-----|-----|------|------|------|-----|-----|------|------|------|
| CPU | 1.427 | 1.214 | 1.120 | 1.048 | 1.034 | 1.504 | 1.445 | 1.345 | 1.273 | 1.272 |
| lossu | 0.077 | 0.055 | 0.050 | 0.051 | 0.059 | 0.158 | 0.108 | 0.077 | 0.079 | 0.090 |
| lossv | 0.079 | 0.063 | 0.044 | 0.044 | 0.047 | 0.158 | 0.112 | 0.112 | 0.105 | 0.116 |
| nA | 60 | 35 | 20 | 12 | 10 | 114 | 56 | 31.5 | 16 | 12 |
| nB | 59.5 | 33 | 20 | 12 | 10 | 104 | 53.5 | 25 | 16 | 12 |
| $\rho_1$ | 0.925 | 0.911 | 0.903 | 0.900 | 0.897 | 0.936 | 0.915 | 0.901 | 0.897 | 0.894 |
| $\rho_2$ | 0.879 | 0.842 | 0.815 | 0.797 | 0.789 | 0.896 | 0.853 | 0.823 | 0.796 | 0.788 |

| b   | 0.8 | 1   | 1.2  | 1.4  | 1.6  | 0.8 | 1   | 1.2  | 1.4  | 1.6  |
|-----|-----|-----|------|------|------|-----|-----|------|------|------|
| CPU | 1.144 | 1.079 | 1.029 | 1.019 | 1.011 | 1.419 | 1.349 | 1.290 | 1.178 | 1.071 |
| lossu | 0.033 | 0.022 | 0.018 | 0.018 | 0.020 | 0.041 | 0.022 | 0.015 | 0.014 | 0.015 |
| lossv | 0.031 | 0.018 | 0.014 | 0.014 | 0.017 | 0.034 | 0.018 | 0.012 | 0.011 | 0.012 |
| nA | 79.5 | 37.5 | 16 | 11 | 10 | 92.5 | 38.5 | 18.5 | 12 | 10 |
| nB | 77.5 | 34.5 | 16 | 12 | 10 | 91 | 36 | 17.5 | 11 | 10 |
| $\rho_1$ | 0.913 | 0.904 | 0.902 | 0.898 | 0.913 | 0.904 | 0.902 | 0.900 | 0.899 | 0.915 |
| $\rho_2$ | 0.849 | 0.816 | 0.806 | 0.801 | 0.799 | 0.846 | 0.818 | 0.806 | 0.802 | 0.800 |
Table 7: Comparison of A-ManPG and CoLaR for multiple sparse CCA \([n, p, q]\). Covariance matrix: sparse inverse matrix

| n, p, q | \(A\)-ManPG-1 | CoLaR-1 | CoLaR-100 |
|--------|----------------|---------|-----------|
|        | \(\rho_1\)     | \(\rho_2\) | \(\rho_1\) |
|        | \(\rho_2\)     | \(\rho_1\) | \(\rho_2\) |
| (200, 300, 300) | | | |
| CPU  | 0.810 | 0.808 | 0.807 |
| lasso | 0.574 | 0.573 | 0.572 |
| lossu | 0.044 | 0.043 | 0.042 |
| lossv | 0.919 | 0.918 | 0.917 |
| nA   | 52.5 | 52.0 | 51.5 |
| nB   | 52.5 | 52.0 | 51.5 |

| (500, 300, 300) | | | |
| CPU  | 0.777 | 0.776 | 0.775 |
| lasso | 0.574 | 0.573 | 0.572 |
| lossu | 0.044 | 0.043 | 0.042 |
| lossv | 0.919 | 0.918 | 0.917 |
| nA   | 52.5 | 52.0 | 51.5 |
| nB   | 52.5 | 52.0 | 51.5 |

| (200, 600, 600) | | | |
| CPU  | 0.810 | 0.808 | 0.807 |
| lasso | 0.574 | 0.573 | 0.572 |
| lossu | 0.044 | 0.043 | 0.042 |
| lossv | 0.919 | 0.918 | 0.917 |
| nA   | 52.5 | 52.0 | 51.5 |
| nB   | 52.5 | 52.0 | 51.5 |

| (500, 600, 600) | | | |
| CPU  | 0.777 | 0.776 | 0.775 |
| lasso | 0.574 | 0.573 | 0.572 |
| lossu | 0.044 | 0.043 | 0.042 |
| lossv | 0.919 | 0.918 | 0.917 |
| nA   | 52.5 | 52.0 | 51.5 |
| nB   | 52.5 | 52.0 | 51.5 |
2. For any $X \in M$, it holds that
\[
\lim_{T \to X, M \ni \xi \to 0} \frac{||R_X(\xi) - (X + \xi)||_F}{||\xi||_F} = 0.
\]

Common retractions on the Stiefel manifold $\text{St}(p, r) = \{X : X^T X = I_r, X \in \mathbb{R}^{p \times r}\}$ include the polar decomposition
\[
R_X^{\text{polar}}(\xi) = (X + \xi)(I_r + \xi^\top \xi)^{-1/2},
\]
the QR decomposition
\[
R_X^{\text{QR}}(\xi) = \text{qf}(X + \xi),
\]
where $\text{qf}(A)$ is the $Q$ factor of the QR factorization of $A$, and the Cayley transformation
\[
R_X^{\text{cayley}}(\xi) = (I_p - \frac{1}{2} W(\xi))^{-1}(I_p + \frac{1}{2} W(\xi))X,
\]
where $W(\xi) = (I_p - \frac{1}{2} XX^\top)\xi X^\top - X\xi^\top (I_p - \frac{1}{2} XX^\top)$. In our numerical tests, we chose the polar decomposition for retraction.

### A.1 Preliminaries of Generalized Stiefel manifold

We denote the generalized Stiefel manifold as $\mathcal{M} = \text{GSt}(p, r) = \{U \in \mathbb{R}^{p \times r} : U^T M U = I_r\}$, where $M \in \mathbb{R}^{p \times p}$ is positive definite. The tangent space of $\text{GSt}(p, r)$ at $U$ is given by $T_U \mathcal{M} = \{\delta : \delta^T M U + U^T M \delta = 0\}$.

The generalized polar decomposition of a tangent vector $Y \in T_U \mathcal{M}$ can be computed as follows:
\[
R_Y^{\text{polar}} = \bar{U}(Q\Lambda^{-1/2}Q^\top)^{\top},
\]
where $\bar{U}\Sigma\bar{V}^\top = Y$ is the truncated SVD of $Y$, and $Q, \Lambda$ are obtained from the eigenvalue decomposition $Q\Lambda Q^\top = \bar{U}^\top M \bar{U}$.

### A.2 Optimality Condition of Manifold Optimization

**Definition A.2.** (Generalized Clarke subdifferential [22]) For a locally Lipschitz function $F$ on $\mathcal{M}$, the Riemannian generalized directional derivative of $F$ at $X \in \mathcal{M}$ in direction $V$ is defined by
\[
F^\circ(X, V) = \limsup_{Y \to X, t \downarrow 0} \frac{F(\phi^{-1}(\phi(Y) + tD\phi(X)[V])) - f(\phi^{-1}(\phi(Y)))}{t},
\]
where $(\phi, U)$ is a coordinate chart at $X$ and $D\phi(X)$ denotes the Jacobian of $\phi(X)$. The generalized gradient or the Clarke subdifferential of $F$ at $X \in \mathcal{M}$, denoted by $\hat{\partial}F(X)$, is given by
\[
\hat{\partial}F(X) = \{\xi \in T_X \mathcal{M} : \langle \xi, V \rangle \leq F^\circ(X, V), \forall V \in T_X \mathcal{M}\}.
\]

**Definition A.3.** ([54]) A function $f$ is said to be regular at $X \in \mathcal{M}$ along $T_X \mathcal{M}$ if
- for all $V \in T_X \mathcal{M}$, $f'(X; V) = \lim_{t \downarrow 0} \frac{f(X+tV) - f(X)}{t}$ exists, and
- for all $V \in T_X \mathcal{M}$, $f'(X; V) = f^\circ(X; V)$. 

20
For smooth function $f$, we know that $\nabla f(X) = \text{Proj}_{T_X M} \nabla f(X)$ since the metric on the manifold is the Euclidean Frobenius metric. Here $\nabla f$ denotes the Riemannian gradient of $f$, and $\text{Proj}_{T_X M}$ denotes the projection onto $T_X M$. According to Theorem 5.1 in [54], for a regular function $F$, we have $\hat{\partial} F(X) = \text{Proj}_{T_X M}(\partial F(X))$. Moreover, let $X = (A, B)$, the function $F(X) = H(X) + f(A) + g(B)$ in problem (3.1) is regular according to Lemma 5.1 in [54]. Therefore, we have $\hat{\partial} F(X) = \nabla F(A, B) + \text{Proj}_{T_A M_1}(\partial f(A)) + \text{Proj}_{T_B M_2}(\partial g(B))$. By Theorem 4.1 in [54], the first-order optimality condition of problem (3.1) is given by

$$0 \in \nabla H(A, B) + \text{Proj}_{T_A M_1}(\partial f(A)) + \text{Proj}_{T_B M_2}(\partial g(B)).$$ \hfill (A.3)

Definition A.4. A point $X \in M$ is called a stationary point of problem (3.1) if it satisfies the first-order optimality condition (A.3).

B Semi-smoothness of Proximal Mapping

Definition B.1. Let $F : \Omega \to \mathbb{R}^q$ be locally Lipschitz continuous at $X \in \Omega \subset \mathbb{R}^p$. The $B$-subdifferential of $F$ at $X$ is defined by

$$\partial_B F(X) := \{ \lim_{k \to \infty} F'(X_k) | X^k \in D_F, X_k \to X \},$$

where $D_F$ be the set of differentiable points of $F$ in $\Omega$. The set $\partial F(X) = \text{conv}(\partial_B F(X))$ is called Clarke’s generalized Jacobian, where $\text{conv}$ denotes the convex hull.

Note that if $q = 1$ and $F$ is convex, then the definition is the same as that of standard convex subdifferential. So, we use the notation $\partial$ for the general purpose.

Definition B.2. [32, 39] Let $F : \Omega \to \mathbb{R}^q$ be locally Lipschitz continuous at $X \in \Omega \subset \mathbb{R}^p$. We say that $F$ is semi-smooth at $X \in \Omega$ if $F$ is directionally differentiable at $X$ and for any $J \in \partial F(X + \Delta X)$ with $\Delta X \to 0$,

$$F(X + \Delta X) - F(X) - J\Delta X = o(\|\Delta X\|).$$

We say $F$ is strongly semi-smooth if $F$ is semi-smooth at $X$ and

$$F(X + \Delta X) - F(X) - J\Delta X = O(\|\Delta X\|^2).$$

We say that $F$ is a semi-smooth function on $\Omega$ if it is semi-smooth everywhere in $\Omega$.

C Global Convergence of A-ManPG (Algorithm 1)

To show the global convergence of A-ManPG, we need the following assumptions for problem (3.1), which are commonly used in first-order methods.

Assumption C.1. (1) $f$ and $g$ are convex and Lipschitz continuous with Lipschitz constants $L_f$ and $L_g$, respectively.

(2) $\nabla_A H(A, B)$ is Lipschitz continuous with respect to $A$ when fixing $B$, and the Lipschitz constant is $L_A$. Similarly, $\nabla_B H(A, B)$ is Lipschitz continuous with respect to $B$ when fixing $A$, and the Lipschitz constant is $L_B$.

(3) $F$ is lower bounded by a constant $F^*$. 

21
Note here the Lipschitz continuity and convexity are all defined in the Euclidean space.

We prove that the sequence generated by A-ManPG converges to stationary point of \( \text{(3.1)} \) in this section. We need the following two properties of retraction whose proofs can be found in \([30]\).

**Lemma C.2.** Let \( \mathcal{M} \) be a compact embedded submanifold in Euclidean space. For all \( X \in \mathcal{M} \) and \( \xi \in T_X \mathcal{M} \), there exist constants \( M_1 > 0 \) and \( M_2 > 0 \) such that the following two inequalities hold:

\[
\|R_X(\xi) - X\|_F \leq M_1\|\xi\|_F, \forall X \in \mathcal{M}, \xi \in T_X \mathcal{M}, \quad (C.1)
\]

\[
\|R_X(\xi) - (X + \xi)\|_F \leq M_2\|\xi\|_F^2, \forall X \in \mathcal{M}, \xi \in T_X \mathcal{M}. \quad (C.2)
\]

Note that the (generalized) Stiefel manifold is compact, so the two inequalities hold naturally.

**Definition C.3.** A function \( f(X) \) is \( \alpha \)-strongly convex in \( \mathbb{R}^p \) if

\[
f(Y) \geq f(X) + \langle \partial f(X), Y - X \rangle + \frac{\alpha}{2}\|Y - X\|^2
\]

holds for \( \forall X, Y \in \mathbb{R}^p \).

The following lemma shows that \( D_k^A \) and \( D_k^B \) obtained from \( \text{(3.3)} \) are descent directions in the tangent space.

**Lemma C.4.** The following inequalities hold for any \( \alpha \in [0, 1] \) if \( t_1 \leq 1/L_A, t_2 \leq 1/L_B \):

\[
H(A_k + \alpha D_k^A, B_k) + f(A_k + \alpha D_k^A) \leq H(A_k, B_k) + f(A_k) - \frac{\alpha}{2t_1}\|D_k^A\|_F^2, \quad (C.3)
\]

\[
H(A_{k+1}, B_k + \alpha D_k^B) + g(B_k + \alpha D_k^B) \leq H(A_{k+1}, B_k) + g(B_k) - \frac{\alpha}{2t_2}\|D_k^B\|_F^2. \quad (C.4)
\]

**Proof.** For simplicity, we only prove inequality \( \text{(C.3)} \). The proof of \( \text{(C.4)} \) is similar. Since the objective function \( G(D) := \langle \nabla_A H(A_k, B_k), D \rangle + \frac{1}{2t_1}\|D\|_F^2 + f(A_k + D) \) is \( 1/t_1 \)-strongly convex, we have

\[
G(\hat{D}) \geq G(D) + \langle \partial G(D), \hat{D} - D \rangle + \frac{1}{2t_1}\|\hat{D} - D\|_F^2, \forall D, \hat{D}. \quad (C.5)
\]

Specifically, if \( D, \hat{D} \) are feasible, i.e., \( D, \hat{D} \in T_{A_k} \mathcal{M}_1 \), we have

\[
\langle \partial G(D), \hat{D} - D \rangle = \langle \text{Proj}_{T_{A_k} \mathcal{M}_1} \partial G(D), \hat{D} - D \rangle.
\]

From the optimality condition of \( \text{(3.3)} \), we have \( 0 \in \text{Proj}_{T_{A_k} \mathcal{M}_1} \partial G(D^A_k) \). Letting \( D = D^A_k \), \( \hat{D} = \alpha D^A_k, \alpha \in [0, 1] \) in \( \text{(C.5)} \) yields

\[
G(\alpha D^A_k) \geq G(D^A_k) + \frac{(1 - \alpha)^2}{2t_1}\|D^A_k\|_F^2,
\]

which implies

\[
\langle \nabla_A H(A_k, B_k), \alpha D^A_k \rangle + \frac{1}{2t_1}\|\alpha D^A_k\|_F^2 + f(A_k + \alpha D^A_k) \geq \langle \nabla_A H(A_k, B_k), D_k \rangle + \frac{1}{2t_1}\|D^A_k\|_F^2 + f(A_k) + D^A_k \cdot \left( \frac{1 - \alpha^2}{2t_1}\|D^A_k\|_F^2. \quad (C.6)
\]

Combining with the convexity of \( f \), \( \text{(C.6)} \) yields

\[
(1 - \alpha)\langle \nabla_A H(A_k, B_k), D_k \rangle + \frac{1 - \alpha}{t_1}\|D_k\|_F^2 + (1 - \alpha)(f(A_k + D^A_k) - f(A_k)) \leq 0. \quad (C.7)
\]
Combining the convexity of \( f \) and the Lipschitz continuity of \( \nabla H(A, B_k) \), we have
\[
H(A_k + \alpha D^A_k, B_k) - H(A_k, B_k) + f(A_k + \alpha D^A_k) - f(A_k)
\]
\[
\leq \alpha \langle \nabla A H(A_k, B_k), D^A_k \rangle + \frac{\alpha^2}{2t_1} \| D^A_k \|^2_F + \alpha (f(A_k + D_k) - f(A_k))
\]
\[
\leq -\frac{\alpha}{2t_1} \| D^A_k \|^2_F,
\]
where the last inequality holds from \( \alpha \in [0, 1] \) and \((C.7)\). \( \square \)

The following lemma shows that if one cannot make any progress by solving \((3.3)\), i.e., \( D^A_k = 0, D^B_k = 0 \), then a stationary point is found.

**Lemma C.5.** If \( D^A_k = 0 \) and \( D^B_k = 0 \), then \((A_k, B_k)\) is a stationary point of problem \((3.1)\).

**Proof.** By Theorem 4.1 in [54], the optimality conditions for the \( A \)-subproblem in \((3.3)\) are given by
\[
0 \in D^A_k/t_1 + \nabla A H(A_k, B_k) + \text{Proj}_{T_A k, M_1} \partial f(A_k + D), \quad \text{and} \quad D^A_k \in T_A k, M_1.
\]
If \( D^A_k = 0 \), it follows that
\[
0 \in \nabla A H(A_k, B_k) + \text{Proj}_{T_A k, M_1} \partial f(A_k).
\]
Similarly, if \( D^B_k = 0 \), we obtain
\[
0 \in \nabla B H(A_k, B_k) + \text{Proj}_{T_B k, M_2} \partial g(B_k).
\]
Combining \((C.8)\) and \((C.9)\) yields the first-order optimality condition of problem \((3.1)\) since \((A_k, B_k) \in (M_1, M_2)\). \( \square \)

**Lemma C.6.** There exist constants \( \bar{\alpha}_1, \bar{\alpha}_2 > 0 \) and \( \bar{\beta}_1, \bar{\beta}_2 > 0 \) such that for any \( 0 < \alpha_1 \leq \min\{1, \bar{\alpha}_1\}, 0 < \alpha_2 \leq \min\{1, \bar{\alpha}_2\} \), the sequence \( \{(A_k, B_k)\} \) generated by Algorithm \([2]\) satisfies the following inequalities:
\[
F(A_{k+1}, B_k) - F(A_k, B_k) \leq -\bar{\beta}_1 \| D^A_k \|^2_F, \tag{C.10}
\]
\[
F(A_{k+1}, B_{k+1}) - F(A_k, B_k) \leq -\bar{\beta}_2 \| D^B_k \|^2_F. \tag{C.11}
\]

**Proof.** We prove \((C.10)\) by induction, and the proof of \((C.11)\) is similar and thus omitted. Define \( A^k_k = A_k + \alpha_1 D^A_k \). For \( k = 0 \), by Lemma \([C.2]\) \((C.1)\) and \([C.2]\) hold for \( X = A_0 \). Note that \( A_{k+1} = R_A (\alpha_1 D^A_k) \). From the Lipschitz continuity of \( \nabla H(A, B_k) \), we have
\[
H(A_{k+1}, B_k) - H(A_k, B_k)
\]
\[
\leq \langle \nabla_A H(A_k, B_k), A_{k+1} - A_k \rangle + \frac{L_A}{2} \| A_{k+1} - A_k \|^2_F
\]
\[
= \langle \nabla_A H(A_k, B_k), A_{k+1} - A^+_k + A^+_k - A_k \rangle + \frac{L_A}{2} \| A_{k+1} - A_k \|^2_F
\]
\[
\leq M_2 \| \nabla_A H(A_k, B_k) \|_F \| \alpha_1 D^A_k \|^2_F + \alpha_1 \langle \nabla_A H(A_k, B_k), D^A_k \rangle + \frac{L_A M_1}{2} \| \alpha_1 D^A_k \|^2_F, \tag{C.12}
\]
where the last inequality is due to \((C.1)\) and \((C.2)\). Since \( \nabla_A H(A, B_k) \) is continuous on the compact set \( M_1 \), there exists a constant \( G > 0 \) such that \( \| \nabla_A H(A, B_k) \|_F \leq G \) for all \( A \in M_1 \). It then follows from \((C.12)\) that
\[
H(A_{k+1}, B_k) - H(A_k, B_k) \leq c_0 \alpha^2 \| D^A_k \|^2_F + \alpha_1 \langle \nabla_A H(A_k, B_k), D^A_k \rangle, \tag{C.13}
\]
\[23\]
where \( c_0 = M_2G + L_A M_1 / 2 \). From (C.13) we can show the following inequalities:

\[
F(A_{k+1}, B_k) - F(A_k, B_k) \leq \alpha_1(\nabla_A H(A_k, B_k), D_k^A) + c_0^2 \| D_k^A \|_F^2 + f(A_{k+1}) - f(A_k) + f(A_k^+) - f(A_k) \\
\quad \leq \alpha_1(\nabla_A H(A_k, B_k), D_k^A) + c_0^2 \| D_k^A \|_F^2 + L_f \| A_{k+1} - A_k^+ \|_F + \alpha_1 (f(A_k + D_k^A) - f(A_k)) \\
\leq (c_0 \alpha_1^2 + L_f M_2 \alpha_1^2) \| D_k^A \|_F^2 + \alpha_1 \left[ (\nabla_A H(A_k, B_k), D_k^A) + f(A_k + D_k^A) - f(A_k) \right] \\
\leq \left[ (c_0 + L_f M_2) \alpha_1^2 - \alpha_1 / t_1 \right] \| D_k^A \|_F^2, \\
\]  
where the second inequality follows from the Lipschitz continuity of \( f(A) \). Define function \( \beta(\alpha_1) = -(c_0 + L_f M_2) \alpha_1^2 + \alpha_1 / t_1, \tilde{\alpha}_1 = \frac{1}{2(c_0 + L_f M_2) t_1} \). It is easy to see from (C.14) that

\[
F(A_{k+1}, B_k) - F(A_k, B_k) \leq -\tilde{\beta}_1 \| D_k^A \|_F^2, \quad \text{if } 0 < \alpha_1 \leq \min\{1, \tilde{\alpha}_1\},
\]

where

\[
\tilde{\beta}_1 = \begin{cases} 
\beta(\alpha_1) & \text{if } \tilde{\alpha}_1 \leq 1, \\
\beta(1) & \text{if } \tilde{\alpha}_1 > 1.
\end{cases}
\]

Thus, (C.10) holds for \( k = 0 \). Suppose that (C.10) holds for \( k \geq 1 \), with the same argument, it follows that (C.10) holds for \( k + 1 \) and \( A_{k+1} \in \mathcal{M}_1 \).

Now we are ready to give the proof of Theorem 3.3.

**Proof.** By Lemma C.6 and the lower boundedness of \( F(A, B) \), we have

\[
\lim_{k \to \infty} (\tilde{\beta}_1 \| D_k^A \|_F^2 + \tilde{\beta}_2 \| D_k^B \|_F^2) = 0.
\]

Combining with Lemma C.5 it follows that any limit point of \( \{(A_k, B_k)\} \) is a stationary point of (3.1). Moreover, since \( \mathcal{M}_1 \) and \( \mathcal{M}_2 \) are compact, there exists at least one limit point of the sequence \( \{(A_k, B_k)\} \).

Furthermore, suppose that Algorithm 1 does not terminate after \( K \) iterations, i.e., \( \tilde{\beta}_1 \| D_k^A \|_F^2 + \tilde{\beta}_2 \| D_k^B \|_F^2 > \varepsilon^2 \) for all \( k = 0, 1, \ldots, K - 1 \). In this case, we have

\[
F(A_0, B_0) - F^* \geq F(A_K, B_K) \geq (\tilde{\beta}_1 + \tilde{\beta}_2) \sum_{k=0}^{K-1} (\| D_k^A \|_F^2 + \| D_k^B \|_F^2) > (\tilde{\beta}_1 + \tilde{\beta}_2) K \varepsilon^2.
\]

Therefore, Algorithm 1 finds an \( \varepsilon \)-stationary point, after \( K \simeq (F(A_0, B_0) - F^*) / ((\tilde{\beta}_1 + \tilde{\beta}_2) \varepsilon^2) \) iterations. 

**References**

[1] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[2] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Lojasiewicz inequality. *Mathematics of Operations Research, 35*(2):438–457, 2010.

[3] J. Baik and J. W. Silverstein. Eigenvalues of large sample covariance matrices of spiked population models. *Journal of Multivariate Analysis, 97*(6):1382–1408, 2006.

[4] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences, 2*(1):183–202, 2009.
[5] T. Bendory, Y. C. Eldar, and N. Boumal. Non-convex phase retrieval from STFT measurements. *IEEE Transactions on Information Theory*, 2018.

[6] J. Bolte, S. Sabach, and M. Teboulle. Proximal alternating linearized minimization or nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1–2):459–494, 2014.

[7] N. Boumal. Nonconvex phase synchronization. *SIAM Journal on Optimization*, 26(4):2355–2377, 2016.

[8] N. Boumal and P.-A. Absil. RTRMC: A Riemannian trust-region method for low-rank matrix completion. In *NIPS*, 2011.

[9] N. Boumal, P.-A. Absil, and C. Cartis. Global rates of convergence for nonconvex optimization on manifolds. *IMA Journal of Numerical Analysis*, 2018.

[10] M. Chen, C. Gao, Z. Ren, and H. H. Zhou. Sparse CCA via precision adjusted iterative thresholding. *arXiv preprint arXiv:1311.6186*, 2013.

[11] S. Chen, S. Ma, A. M.-C. So, and T. Zhang. Proximal gradient method for manifold optimization. *arXiv preprint arXiv:1811.00980*, 2018.

[12] A. Cherian and S. Sra. Riemannian dictionary learning and sparse coding for positive definite matrices. *IEEE Trans. Neural Networks and Learning Systems*, 2016.

[13] A. d’Aspremont. Identifying small mean-reverting portfolios. *Quantitative Finance*, 11(3):351–364, 2011.

[14] A. d’Aspremont, L. El Ghaoui, M. I. Jordan, and G. R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*, 49(3):434–448, 2007.

[15] A. d’Aspremont, F. Bach, and L. El Ghaoui. Optimal solutions for sparse principal component analysis. *Journal of Mach. Learn. Res.*, 9:1269–1294, 2008.

[16] N. B. Erichson, P. Zheng, K. Manoharz, S. L. Brunton, J. N. Kutz, and A. Y. Aravkinz. Sparse principal component analysis via variable projection. *arXiv preprint arXiv:1804.00341*, 2018.

[17] F. Facchinei and J. Pang. *Finite-dimensional variational inequalities and complementarity problems*. Springer Science & Business Media, 2007.

[18] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[19] C. Gao, Z. Ma, and H.-H. Zhou. Sparse CCA: Adaptive estimation and computational barriers. *The Annals of Statistics*, 45(5):2074–2101, 2017.

[20] D. R. Hardoon and J. Shawe-Taylor. Sparse canonical correlation analysis. *Machine Learning*, 83(3):331–353, 2011.

[21] J.-B. Hiriart-Urruty, J.-J. Strodiot, and V. H. Nguyen. Generalized hessian matrix and second-order optimality conditions for problems with c 1, 1 data. *Applied mathematics and optimization*, 11(1):43–56, 1984.

[22] S. Hosseini and M. R. Pouryayevali. Generalized gradients and characterization of epi-lipschitz sets in Riemannian manifolds. *Nonlinear Analysis: Theory, Methods & Applications*, 72(12):3884–3895, 2011.
[23] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.

[24] W. Huang and P. Hand. Blind deconvolution by a steepest descent algorithm on a quotient manifold. https://arxiv.org/pdf/1710.03309.pdf, 2018.

[25] I. Jolliffe, N. Trendafilov, and M. Uddin. A modified principal component technique based on the LASSO. *Journal of computational and Graphical Statistics*, 12(3):531–547, 2003.

[26] M. Journee, Yu. Nesterov, P. Richtarik, and R. Sepulchre. Generalized power method for sparse principal component analysis. *Journal of Machine Learning Research*, 11(Feb):517–553, 2010.

[27] X. Li, D. Sun, and K.-C. Toh. A highly efficient semismooth Newton augmented Lagrangian method for solving Lasso problems. *SIAM J. Optim.*, 28:433–458, 2018.

[28] H. Liu, M.-C. Yue, and A. M.-C. So. On the estimation performance and convergence rate of the generalized power method for phase synchronizaton. *SIAM J. Optim.*, 27(4):2426–2446, 2017.

[29] Z. Lu and Y. Zhang. An augmented lagrangian approach for sparse principal component analysis. *Mathematical Programming*, 135:149–193, 2012.

[30] S. Ma. Alternating direction method of multipliers for sparse principal component analysis. *Journal of the Operations Research Society of China*, 1(2):253–274, 2013.

[31] J. R. Magnus and H. Neudecker. Matrix differential calculus with applications in statistics and econometrics. *Wiley series in probability and mathematical statistics*, 2018.

[32] R. Mifflin. Semismooth and semiconvex functions in constrained optimization. *SIAM J. Control Optim.*, 15:959–972, 1977.

[33] B. Moghaddam, Y. Weiss, and S. Avidan. Spectral bounds for sparse pca: Exact and greedy algorithms. In *Advances in neural information processing systems*, pages 915–922, 2006.

[34] B. Nadler. Finite sample approximation results for principal component analysis: A matrix perturbation approach. *The Annals of Statistics*, 36(6):2791–2817, 2008.

[35] E. Parkhomenko, D. Tritchler, and J. Beyene. Sparse canonical correlation analysis with application to genomic data integration. *Statistical applications in genetics and molecular biology*, 8(1):1–34, 2009.

[36] D. Paul. Asymptotics of sample eigenstructure for a large dimensional spiked covariance model. *Statistica Sinica*, pages 1617–1642, 2007.

[37] K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[38] H. Qi and D. Sun. An augmented Lagrangian dual approach for the H-weighted nearest correlation matrix problem. *IMA Journal of Numerical Analysis*, 31:491–511, 2011.

[39] L. Qi and J. Sun. A nonsmooth version of Newton’s method. *Math. Program.*, 58:353–367, 1993.
[40] H. Shen and J. Z. Huang. Sparse principal component analysis via regularized low rank matrix approximation. *Journal of Multivariate Analysis*, 99(6):1015–1034, 2008.

[41] M. V. Solodov and B. F. Svaiter. A globally convergent inexact Newton method for systems of monotone equations. In *Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods*, pages 355–369. Springer, 1998.

[42] J. Sun, Q. Qu, and J. Wright. Complete dictionary recovery over the sphere i: Overview and the geometric picture. *IEEE Trans. Information Theory*, 63(2):853–884, 2017.

[43] J. Sun, Q. Qu, and J. Wright. A geometrical analysis of phase retrieval. *Foundations of Computational Mathematics*, 18(5):1131–1198, 2018.

[44] X. Suo, V. Minden, B. Nelson, R. Tibshirani, and M. Saunders. Sparse canonical correlation analysis. *arXiv preprint arXiv:1705.10865*, 2017.

[45] M. Ulbrich. *Semismooth Newton methods for variational inequalities and constrained optimization problems in function spaces*, volume 11. SIAM, 2011.

[46] B. Vandereycken. Low-rank matrix completion by Riemannian optimization. *SIAM J. Optim.*, 23(2):1214–1236, 2013.

[47] V. Q. Vu, J. Cho, J. Lei, and K. Rohe. Fantope projection and selection: a near-optimal convex relaxation of sparse pca. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pages 2670–2678. Curran Associates Inc., 2013.

[48] C. Wang, D. Sun, and K.-C. Toh. Solving log-determinant optimization problems by a Newton-CG primal proximal point algorithm. *SIAM J. Optimization*, 20:2994–3013, 2010.

[49] A. Wiesel, M. Kliger, and A. O. Hero III. A greedy approach to sparse canonical correlation analysis. *arXiv preprint arXiv:0801.2748*, 2008.

[50] D. M. Witten, R. Tibshirani, and T. Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534, 2009.

[51] X. Xiao, Y. Li, Z. Wen, and L. Zhang. A regularized semi-smooth Newton method with projection steps for composite convex programs. *Journal of Scientific Computing*, 76(1):364–389, 2018.

[52] J. Yang, D. Sun, and K.-C. Toh. A proximal point algorithm for log-determinant optimization with group Lasso regularization. *SIAM J. Optim.*, 23:857–893, 2013.

[53] L. Yang, D. Sun, and K.-C. Toh. SDPNAL+: a majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints. *Mathematical Programming Computation*, 7(3):331–366, 2015.

[54] W. H. Yang, L.-H. Zhang, and R. Song. Optimality conditions for the nonlinear programming problems on Riemannian manifolds. *Pacific J. Optimization*, 10(2):415–434, 2014.

[55] X.-T. Yuan and T. Zhang. Truncated power method for sparse eigenvalue problems. *J. Mach. Learn. Res.*, 14(1):899–925, 2013.
[56] X. Zhao, D. Sun, and K.-C. Toh. A Newton-CG augmented Lagrangian method for semidefinite programming. *SIAM Journal on Optimization*, 20:1737–1765, 2010.

[57] G. Zhou and K.-C. Toh. Superlinear convergence of a Newton-type algorithm for monotone equations. *Journal of optimization theory and applications*, 125(1):205–221, 2005.

[58] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67(2):301–320, 2005.

[59] H. Zou and L. Xue. A selective overview of sparse principal component analysis. *Proceedings of the IEEE*, 106(8):1311–1320, 2018.

[60] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *J. Comput. Graph. Stat.*, 15(2):265–286, 2006.