DISTRIBUTED ROBUST SUBSPACE RECOVERY
VAHAN HUROYAN∗ AND GILAD LERMAN∗

Abstract. We study Robust Subspace Recovery (RSR) in distributed settings. We consider a huge data set in an ad hoc network without a central processor, where each node has access only to one chunk of the data set. We assume that part of the whole data set lies around a low-dimensional subspace and the other part is composed of outliers that lie away from that subspace. The goal is to recover the underlying subspace for the whole data set, without transferring the data itself between the nodes. We apply the Consensus Based Gradient method for the Geometric Median Subspace algorithm for RSR. We propose an iterative solution for the local dual minimization problem and establish its $r$-linear convergence. We show that this mathematical framework also extends to two simpler problems: Principal Component Analysis and the geometric median. We also explain how to distributely implement the Reaper and Fast Median Subspace algorithms for RSR. We demonstrate the competitive performance of our algorithms for both synthetic and real data.

Key words. Distributed Algorithms, Consensus-Based Algorithms, Principal Component Analysis (PCA), Robust Subspace Recovery (RSR), Geometric Median

AMS subject classifications. 68W15, 65K05, 62H25, 90C06

1. Introduction. Distributed computing is a central theme in modern computation. Its setting includes a system with multiple components, which communicate and coordinate in order to achieve their common computational goal. A special distributed setting assumes a central processor, which is connected to all other processors. This processor contains no data, but has enough memory to handle some computations, such as averaging communicated estimates. A more general distributed setting assumes an arbitrarily connected network of processors, among which the data is partitioned. Each processor computes a local estimate of the desired output based on its local data and on estimates passed by its neighbors. It then communicates its estimate to its neighbors. This procedure iterates until convergence.

Some common approaches for solving distributed computing problems are the diffusion method [12], Consensus-Based Gradient Ascent (CBGA) [5, 7, 9, 15] and the Consensus Alternating Direction Method of Multipliers (CADMM) [9, 23]. Some of these algorithms have been successfully adapted to important applied problems of signal processing and wireless communications [13, 20, 29, 33]. Various distributed algorithms have been proposed for the important problem of Principal Component Analysis (PCA). Most of them are for centrally-processed networks [3, 18, 22, 25, 26, 30], but some of them are for arbitrarily connected networks [1, 5]. To the best of our knowledge there are no distributed algorithms for robust versions of PCA.

This work discusses distributed algorithms for Robust Subspace Recovery (RSR) with arbitrarily connected networks. RSR is an alternative paradigm for PCA that is more robust to outliers. The underlying problem of RSR assumes data points, composed of inliers and outliers, where the inliers are well-explained by an affine low-dimensional subspace and the outliers come from a different model. The goal is to recover the underlying subspace in the presence of outliers; see e.g., [17] and [21] for careful reviews of the problem and its solutions.

We first suggest a distributed implementation for the Geometric Median Subspace (GMS) [32] algorithm for RSR, which applies to arbitrarily connected networks. We propose an iterative algorithm for the local dual problem and establish its $r$-linear convergence...
convergence. Our mathematical framework for distributed GMS extends to two simpler problems: distributed PCA and distributed geometric median. We also propose distributed implementations for two other RSR algorithms: Reaper [17] and FMS [21]. This is done by iterative application of distributed PCA. On the other hand, the GMS implementation does not iterate the distributed scheme and is thus more efficient in terms of the communication cost. We remark that the theorems for robustness of GMS, Reaper and FMS carry over to our distributed setting.

The paper is organized as follows: §2 contains a short introduction to CBGA and its convergence analysis; §3 demonstrates the basic ideas behind the distributed CBGA algorithm for GMS on two simpler but important problems: distributed PCA and distributed geometric median; §4 proposes the distributed CBGA algorithm for GMS and discusses its various properties; §5 proposes immediate distributed implementations for the Reaper and FMS algorithms; and §6 concludes with numerical experiments that test the proposed algorithms for distributed RSR. Details of proofs of all theoretical statements are left to the appendix.

2. Review of Consensus-Based Gradient Ascent (CBGA). The setting of CBGA [5,6] assumes a connected network, with \( K \) nodes and \( M \) edges. It also assumes a convex set of matrices \( S \subseteq \mathbb{R}^{D \times D} \) and convex functions \( F_1, \ldots, F_K \) on \( S \), associated with the \( K \) nodes. The goal is to minimize \( \sum_{k=1}^{K} F_k \) over \( S \), where each node \( k \) has only access to \( F_k \) and may communicate to its neighbors. We reformulate this problem by using local neighborhoods. For \( 1 \leq j \leq K \), let \( N_j \) denote the set of all nodes connected (by an edge) to the node \( j \). The desired problem, \( \min_{Q \in S} \sum_{k=1}^{K} F_k(Q) \), can be computed locally as follows:

\[
(1) \quad \min_{Q_1 \ldots Q_K \in S} \sum_{k=1}^{K} F_k(Q_k), \text{ where } Q_k = Q_q, \forall 1 \leq k \leq K, q \in N_k, q < k.
\]

The constraints in the right side of (1) are called consensus constraints. The consensus constraints can be formulated with matrices as follows. For \( 1 \leq m \leq M \), let \( e_m \) denote the edge indexed by \( m \). We write \( e_m = \{k, q\} \) whenever \( e_m \) connects the nodes indexed by \( k \) and \( q \). For \( 1 \leq k \leq K \) and \( 1 \leq m \leq M \), \( C_{mk} \) is the following \( D \times D \) matrix

\[
(2) \quad C_{mk} = c_{mk} I, \text{ where } c_{mk} = \begin{cases} 
1, & \text{if } e_m = \{k, q\} \text{ and } k < q; \\
-1, & \text{if } e_m = \{k, q\} \text{ and } q < k; \\
0, & \text{otherwise}.
\end{cases}
\]

Let \( C \) denote the \( DM \times DK \) block matrix with blocks \( \{C_{mk}\}_{m=1,k=1}^{M,K} \) and let \( \bar{Q} = [Q_1^T, \ldots, Q_K^T]^T \), then the consensus constraints can be formulated as \( C\bar{Q} = 0 \).

The minimization problem of (1) is inseparable and thus hard to compute in a distributed setting. That is, one cannot find the exact solution by just computing and adding results from each node. Instead, one needs to invoke the dual problem, which we describe next. The Lagrangian for problem (1) is

\[
L(\Lambda, \bar{Q}) = \sum_{k=1}^{K} F_k(Q_k) + \text{tr}(\Lambda^T C\bar{Q}),
\]

where \( \Lambda = [\Lambda_1^T, \ldots, \Lambda_M^T]^T \in \mathbb{R}^{MD \times D} \), and the dual function is

\[
(3) \quad d(\Lambda) = \min_{\bar{Q} \in S^K} L(\Lambda, \bar{Q}).
\]
Finally, the dual problem of (1) is

\[ \hat{\Lambda} = \arg \max_{\Lambda \in \mathbb{R}^{MD \times D}} d(\Lambda). \]

Recall that strong duality means that the minimizer of (3) with \( \hat{\Lambda} \) found by the dual problem (4) coincides with the minimizer of (1). In the case of strong duality, the dual function in (3) is separable, i.e., by solving the problem in each node one can recover the solution for the whole data. Indeed, we may write

\[ d(\Lambda) = \sum_{k=1}^{K} d_k(\Lambda), \]

where

\[ d_k(\Lambda) = \min_{Q \in S} \left( \sum_{x \in \mathcal{X}_k} \|Qx\|^2 + \sum_{m \in \mathcal{E}_k} c_{mk} \text{tr}(\Lambda^T_m Q) \right) \]

and \( \mathcal{E}_k \) is the set of all edges that contain the node \( k \). One can solve (4) by subgradient descent over \( \Lambda \) and then try to locally solve the separable problem in (3). According to [5,6], one possible subgradient for solving (4) is

\[ g(\Lambda) = C\bar{Q}(\Lambda), \]

where \( \bar{Q}(\Lambda) \) is the solution of (3) for the given \( \Lambda \). Moreover, if \( d(\Lambda) \) is differentiable, then \( g(\Lambda) = C\bar{Q}(\Lambda) \) is the gradient. This subgradient algorithm converges if the following conditions are satisfied (see [5,6]): 1. the set \( H \) is convex and the functions \( F_k \) are convex; 2. strong duality holds for (1); 3. the subgradients of \( d(\Lambda) \) are uniformly bounded for all values of \( \Lambda \).

3. Demonstration of the Main Idea on Two Simpler Problems. Before explaining how to use the CBGA algorithm for GMS, we demonstrate the basic idea on two simpler problems, which share similarities with GMS: distributed computation of the PCA subspace and distributed computation of the geometric median.

3.1. Distributed PCA for Arbitrarily Distributed Network. We first review a convex formulation of PCA for full-rank data and later use it to apply CBGA to distributed PCA. Assume \( X = \{x_i\}_{i=1}^N \) is a data set of \( N \) points in \( \mathbb{R}^D \), centered at 0. The PCA \( d \)-subspace is the \( d \)-dimensional linear subspace minimizing the sum of squared residuals. If the data set \( X \) is full rank, then according to Theorem 10 of [32] the PCA \( d \)-subspace is spanned by the bottom \( d \) eigenvectors of the following matrix \( \hat{Q} \) (or equivalently, the top \( d \) eigenvectors of \( -\hat{Q} \)):

\[ \hat{Q} = \arg \min_{Q \in S} \sum_{x \in \mathcal{X}} \|Qx\|^2, \text{ where } S = \{ Q \in S^D, \text{tr}(Q) = 1 \}. \]

Here and throughout the paper \( S^D \) denotes the set of \( D \)-dimensional symmetric matrices, \( S^D \) denotes the set of \( D \)-dimensional positive semi-definite matrices and \( S^D_{++} \) denotes the set of \( D \)-dimensional positive definite matrices.

Our proposed CBGA-PCA algorithm is similar to [1,5,30], but uses instead the PCA formulation involving (5). This formulation leads to a direct solution of the local optimization problem. In order to apply (5), we need to guarantee full ranks at all nodes. We use again the setting of a data set \( X \) partitioned into \( \{\mathcal{X}_k\}_{k=1}^K \). If \( \mathcal{X}_k \) is not full rank for \( 1 \leq k \leq K \), one could sample an Oblivious Subspace Embedding (OSE) matrix \( H [28] \) and instead of \( \mathcal{X}_k \) consider \( H\mathcal{X}_k \). One common OSE \( H \) has only one non-zero entry per row. Now, by taking appropriate number of rows for \( H \) we can assume that the projected data at each node has full rank.

Next, we clarify the application of CBGA to (5). In view of §2, it is sufficient to compute the dual function of (5) at each node, that is, compute for each \( 1 \leq k \leq K \):

\[ d_k(\Lambda) = \min_{Q \in S} \left( \sum_{x \in \mathcal{X}_k} \|Qx\|^2 + \sum_{m \in \mathcal{E}_k} c_{mk} \text{tr}(\Lambda^T_m Q) \right). \]
We can rewrite (6) as
\[
    d_k(\Lambda) = \min_{Q \in \mathbb{S}^d} \left( \sum_{x \in A_k} \|Qx\|^2 + \text{tr}(A_kQ) \right), \text{ where } A_k = \sum_{m \in E_k} c_m \Lambda_m^T.
\]

Appendix A.1 guarantees the unique minimizer of (7) and explains how to find it.

Since the minimized function in (5) is strongly convex, it follows from [16] that its dual function, defined in (6), is Lipschitz smooth. This implies that the CBGA algorithm for PCA converges to the PCA solution for the total data with rate $O(1/t)$. The complexity of CBGA-PCA is $O(T_{CBGA} \times N_{\max} \times D^2)$ (see §A.1.3). This algorithm is not optimal in terms of complexity and communication. Indeed, one can achieve a simpler and exact distributed PCA algorithm by propagating the local covariance matrices among the network and recovering the exact covariance matrix at each processor. Nevertheless, we find this CBGA-PCA interesting for two reasons. First of all, it is similar to previous attempts [1, 5, 30] that did not clarify how to solve the local dual problem. Second of all, CBGA-PCA simply demonstrates the main idea of the more complicated CBGA-GMS procedure.

3.2. Distributed Geometric Median. The geometric median of a discrete data set $X \subset \mathbb{R}^D$ is defined as
\[
    \arg\min_{y \in \mathbb{R}^D} \sum_{x \in X} \|x - y\|.
\]

Weiszfeld’s algorithm [31] is a common numerical approach to approximating (8) within a sufficiently small error. It applies an iteratively reweighted least squares (IRLS) procedure. However, if in one of the iterations, the estimate coincides with one of the data points, then Weiszfeld’s algorithm fails to converge to the geometric median. To avoid this issue, we consider the following regularized version of (8):
\[
    \arg\min_{y \in \mathbb{R}^D} \sum_{x \in X, \|x - y\| \geq \delta} \|x - y\| + \sum_{x \in X, \|x - y\| < \delta} \left( \frac{\|x - y\|^2}{2\delta} + \frac{\delta}{2} \right),
\]
where $\delta > 0$ is a small regularization parameter. We can solve (9) by the generalized Weiszfeld’s algorithm [11, §4]. This algorithm runs as follows: it starts with an initial guess $y_0 \in \mathbb{R}^D$, and at iteration $s \geq 1$ it computes
\[
    y_s = \sum_{x \in X} \max\left(\frac{x}{\|x - y_{s-1}\|}, \delta\right) \frac{1}{\sum_{x \in X} \max(\|x - y_{s-1}\|, \delta)}.
\]

The sequence $\{y_s\}_{s \in \mathbb{N}}$ r-linearly converges to the solution of (9) (see [11]).

We assume a data set $\mathcal{X}$ with $\{X_k\}_{k=1}^K$ distributed at $K$ nodes, and distributedly compute the regularized geometric median of $\mathcal{X}$ by CBGA. In view of §2, it is enough to compute the dual function of (9) at each node, that is, compute for each $1 \leq k \leq K$
\[
    d_k(\lambda) = \min_{y \in \mathbb{R}^D} \sum_{x \in X_k, \|x - y\| \geq \delta} \|x - y\| + \sum_{x \in X_k, \|x - y\| < \delta} \left( \frac{\|x - y\|^2}{2\delta} + \frac{\delta}{2} \right) + \sum_{m \in E_k} \lambda_m c_m \lambda_m^T y,
\]
where $\lambda = [\lambda_1^T, \ldots, \lambda_M^T]^T \in \mathbb{R}^{MD \times 1}$. We suggest solving (10) by IRLS as follows: start with an initial guess $y_0^s \in \mathbb{R}^D$ and at iteration $s \geq 1$ compute
\[
    y_k^s = \left( 2 \sum_{x \in X_k} \frac{x}{\max(\|x - y_{k-1}^s\|, \delta)} - \sum_{m \in E_k} \lambda_m c_m \lambda_m^T \right) \frac{1}{\left( 2 \sum_{x \in X_k} \frac{1}{\max(\|x - y_{k-1}^s\|, \delta)} \right)}.
\]
DISTRIBUTED ROBUST SUBSPACE RECOVERY

The convergence of \( \{y_k^i\}_{i \in \mathbb{N}} \) follows from that of IRLS (see [11]) and CBGA (see §2).

4. Distributed GMS. We review the GMS problem in §4.1, propose a distributed solution in §4.2, establish convergence guarantees in §4.3 and discuss the time complexity and possible reduction of the communication cost in §4.4.

4.1. Review of GMS. Given a data set \( \mathcal{X} = \{x_i\}_{i=1}^N \subset \mathbb{R}^D \) and a regularization parameter \( \delta > 0 \), the underlying convex minimization problem of GMS [32] is

\[
Q = \arg \min_{Q \in \mathcal{S}} F^\delta(Q),
\]

where \( \mathcal{S} \) is defined in (5) and

\[
F^\delta(Q) = \sum_{x \in \mathcal{X}, \|Qx\| \geq \delta} \|Qx\| + \sum_{x \in \mathcal{X}, \|Qx\| < \delta} \left( \frac{\|Qx\|^2}{2\delta} + \frac{\delta}{2} \right).
\]

Given a target dimension \( 1 \leq d \leq D - 1 \), the output of GMS is a \( d \)-dimensional subspace spanned by the bottom \( d \) eigenvectors of \( \hat{Q} \) (or the top ones of \( -\hat{Q} \)).

4.2. Consensus-Based Subgradient Algorithm for Distributed GMS. We assume a data set \( \mathcal{X} = \{X_k\}_{k=1}^K \) distributed at \( K \) nodes. We further assume that for \( 1 \leq k \leq K \), \( X_k \) is full-rank and generated from a typical model with noisy inliers concentrated around a subspace, then \( X_k \) is expected to satisfy the two-subspaces criterion. If the data set is not full rank, we suggest preprocessing it following the discussion in §3.1.

We follow §2 and solve the minimization problem for the dual function of GMS in each node, while communicating these solutions via CBGA. Following (11) and (12), we need to solve at each node the following optimization problem:

\[
d_k(\Lambda) = \min_{Q \in \mathcal{S}} G^\delta_k(Q) \text{ for } G^\delta_k(Q) = F^\delta_k(Q) + \text{tr}(QA_k),
\]

where \( A_k = \sum_{m \in E_k} c_m A_m^T \), \( \{c_m\}_{m=1}^M \) are defined in (2) and

\[
F^\delta_k(Q) = \sum_{x \in \mathcal{X}_k, \|Qx\| \geq \delta} \|Qx\| + \sum_{x \in \mathcal{X}_k, \|Qx\| < \delta} \left( \frac{\|Qx\|^2}{2\delta} + \frac{\delta}{2} \right).
\]

To find the minimizer of (14) sufficiently fast, we introduce an iterative algorithm similar to Algorithm 2 of [32] and guarantee its \( r \)-linear convergence. Let \( Q^0_k = I/D \) (or arbitrarily fix \( Q^0_k \in S^D_{++} \cap \mathcal{S} \)) and for iteration \( 1 \leq t \leq T \), let \( Q^t_k \) be the solution of the following Lyapunov equation in \( Q \), where \( c_k \in \mathbb{R} \) is chosen such that \( \text{tr}(Q^t_k) = 1 \):

\[
Q \left( \sum_{x \in \mathcal{X}_k} \frac{x x^T}{2 \max(\|Q^{t-1}x\|, \delta)} \right) + \left( \sum_{x \in \mathcal{X}_k} \frac{x x^T}{2 \max(\|Q^{t-1}x\|, \delta)} \right) = c_k I - A_k.
\]

In §4.3 we establish the \( r \)-linear convergence of \( \{Q^t_k\}_{t \in \mathbb{N}} \) to the minimizer of (14). The following lemma establishes the existence and uniqueness of \( c_k \in \mathbb{R} \) and \( Q^t_k \in S^D_{++} \cap \mathcal{S} \), which satisfy (15). It is proved in §A.2.
**Lemma 1.** Let $\mathcal{X} = \{x_i\}_{i=1}^N$ be a full rank data set in $\mathbb{R}^D$, $Q \in S^D_{++} \cap \mathbb{S}$ and $A \in S^D$ with $\text{tr}(A) = 0$ and

$$\|A\|_2 \leq 1/\text{tr}\left(\frac{\sum_{x \in \mathcal{X}} xx^T}{2 \max(\|x\|, \delta)}\right)^{-1}. \tag{16}$$

There exists a unique $c' \in \mathbb{R}$ such that the following equation with $c = c'$

$$P \left(\sum_{x \in \mathcal{X}} \frac{xx^T}{2 \max(\|Qx\|, \delta)}\right) + \left(\sum_{x \in \mathcal{X}} \frac{xx^T}{2 \max(\|Qx\|, \delta)}\right) P + A = cI \tag{17}$$

has a unique solution $P \in S^D_{++} \cap \mathbb{S}$.

If $Q_*$ is the solution of (17) with $c = 0$ and $A = A_k$, then

$$c' = -2(\text{tr}(Q_*) - 1)/\text{tr}\left(\frac{\sum_{x \in \mathcal{X}} xx^T}{2 \max(\|Q_*x\|, \delta)}\right)^{-1}. \tag{18}$$

Algorithm 1 summarizes the above procedure of solving (14). Using this algorithm and CBGA, Algorithm 2 (CBGA-GMS) distributively solves the GMS problem. In §A.3 we discuss how a sufficiently small step size in Algorithm 2 ensures that the above condition (16) is satisfied at each node for all iterations of Algorithm 1. We also explain there why typically the upper bound of (16) is higher.

**Algorithm 1** Algorithm for computing the minimizer of (14)

**Input:** $\mathcal{X} = \{x_1, \ldots, x_N\} \subseteq \mathbb{R}^D$: data, $A_k \in S^D$ with $\text{tr}(A_k) = 0$, $T_{GMS}$: stopping iteration number, $\delta$: regularization parameter (default: $10^{-10}$)

**Output:** $Q_k \in \mathbb{S}$

Set: $Q_k^0 = I/D$ and $t = 0$

while $t \leq T_{GMS}$ or $G_k^t(Q_k^{t+1}) > G_k^t(Q_k^t)$ do

- Let $Q_*$ be the solution of (17) with $Q = Q_k^t$, $c = 0$ and $A = A_k$
- Compute $c'$ according to (18)
- Let $Q_k^{t+1}$ be the solution of (17) with $Q = Q_k^t$, $c = c'$ and $A = A_k$
- $t := t + 1$

end while

return $Q_k := Q_k^t$


4.3. Convergence of CBGA-GMS. The following theorem guarantees that the iterative procedure described in Algorithm 1 $r$-linearly converges to the unique minimizer of (14). This theorem is later proved in §A.4.

**Theorem 1.** Assume $\mathcal{X}_k = \{x_i\}_{i=1}^{N_k} \subseteq \mathbb{R}^D$ satisfies the two-subspaces criterion, $A_k \in S^D$ satisfies (16) and $\text{tr}(A_k) = 0$. If $\{Q_k^t\}_{t \in \mathbb{N}}$ is obtained by Algorithm 1 at node $k$ with $T_{GMS} = \infty$, then it $r$-linearly converges to the unique minimizer of (14).

The conditions for convergence of CBGA discussed in §2 are satisfied for CBGA-GMS. Indeed, the first condition is straightforward, since $G_k^t$ and $\mathbb{S}$ are convex. The
are obtained by directly applying a distributed PCA algorithm. They are the top \(d\) eigenvectors, it reconstructs the \(d\times D\) matrix \(U^T U / \text{tr}(U^T U)\), where \(U \in \mathbb{R}^{d \times D}\) contains the orthogonal top \(d\) eigenvectors as rows. We cannot guarantee the convergence of this modified procedure, but it seems to work well in practice.

5. Distributed Reaper and Distributed FMS. In this section we present distributed versions of two other RSR algorithms: Reaper [17] and FMS [21]. They are obtained by directly applying a distributed PCA algorithm.

5.1. Review of the Reaper and FMS Algorithms. Assume a data set \(X = \{x_i\}_{i=1}^N \subset \mathbb{R}^D\), a target dimension \(d \in \{1, 2, \ldots, D-1\}\), a regularization parameter \(\mu\) sufficiently small constant step size. The computation of the coefficient of (15), \(\sum_{i=1}^N x_i x_i^T / (2 \max(\|Q_k^{-1} x_i\|, \delta))\), requires \(O(N_k \times D^2)\) operations. Solving (15) requires \(O(D^3)\) operations (see [4]). Since \(N_k \geq D\), the total complexity for each iteration of Algorithm 1 at node \(k\) is \(O(N_k \times D^2)\). Denoting \(N_{\text{max}} = \max_{1 \leq k \leq K} N_k\), we conclude that the complexities of Algorithms 1 and 2 are \(O(T_{\text{GMS}} \times N_{\text{max}} \times D^2)\) and \(O(T_{\text{CBGA}} \times T_{\text{GMS}} \times N_{\text{max}} \times D^2)\) respectively.

Algorithm 2 transfers \(D \times D\) matrices between nodes in each iteration, which might not be cost efficient. In order to reduce the communication cost we suggest transferring only the top \(d\) eigenvectors of those matrices. Once a node receives the top \(d\) eigenvectors, it reconstructs the \(D \times D\) matrix \(U^T U / \text{tr}(U^T U)\), where \(U \in \mathbb{R}^{d \times D}\) contains the orthogonal top \(d\) eigenvectors as rows. We cannot guarantee the convergence of this modified procedure, but it seems to work well in practice.

5. Distributed Reaper and Distributed FMS. In this section we present distributed versions of two other RSR algorithms: Reaper [17] and FMS [21]. They are obtained by directly applying a distributed PCA algorithm.

5.1. Review of the Reaper and FMS Algorithms. Assume a data set \(X = \{x_i\}_{i=1}^N \subset \mathbb{R}^D\), a target dimension \(d \in \{1, 2, \ldots, D-1\}\), a regularization parameter \(\mu\) sufficiently small constant step size. The computation of the coefficient of (15), \(\sum_{i=1}^N x_i x_i^T / (2 \max(\|Q_k^{-1} x_i\|, \delta))\), requires \(O(N_k \times D^2)\) operations. Solving (15) requires \(O(D^3)\) operations (see [4]). Since \(N_k \geq D\), the total complexity for each iteration of Algorithm 1 at node \(k\) is \(O(N_k \times D^2)\). Denoting \(N_{\text{max}} = \max_{1 \leq k \leq K} N_k\), we conclude that the complexities of Algorithms 1 and 2 are \(O(T_{\text{GMS}} \times N_{\text{max}} \times D^2)\) and \(O(T_{\text{CBGA}} \times T_{\text{GMS}} \times N_{\text{max}} \times D^2)\) respectively.
\( \delta > 0 \) and for FMS assume an additional parameter \( 0 < p \leq 1 \).

The Reaper algorithm [17] solves the following convex optimization problem:

\[
\min_{0 \leq P \preceq I, \|x - Px\| = \delta} \sum_{x \in X: \|x - Px\| \geq \delta} \|x - Px\| + \sum_{x \in X: \|x - Px\| < \delta} \left( \frac{\|x - Px\|^2}{2\delta} + \frac{\delta}{2} \right).
\]

It uses an IRLS framework, which iteratively weighs the data and applies full PCA [17].

The FMS algorithm [21] starts with \( L_0 \), the PCA subspace. At each iteration \( s \geq 1 \), it computes the \( d \)-dimensional PCA subspace \( L_s \) of the scaled data set:

\[
\{x_i / \max(\text{dist}(x_i, L_{s-1})^{(2-p)/2}, \sqrt{p\delta})\}_{i=1}^N.
\]

This procedure iterates until convergence.

5.2. Distributed Implementations for Reaper and FMS. We assume a data set \( X \) with \( \{X_k\}_{k=1}^K \) distributed at \( K \) nodes so that \( X_k \) has full rank for \( 1 \leq k \leq K \). If the data is not full rank, it is preprocessed according to the discussion in §3.1.

Distributed Reaper requires distributedly solving (20). This can be done by applying distributed full PCA at each IRLS iteration of Algorithm 4.1 of [17]. More precisely, this procedure first initializes the IRLS weights by \( \beta^0_x = 1 \) for all data points \( x \in X \). Then, at each iteration \( s \geq 1 \) it applies distributed full PCA of the weighted data set \( \{\sqrt{\beta^s_x}x\}_{x \in X} \) to obtain \( P^s_k \) at each processor with index \( k \). Then, it updates the weights by \( \beta_x \leftarrow 1 / \max(\delta, \|x - P^s_k x\|) \), for all \( x \in X \). This procedure is iterated until convergence and the local subspace is obtained by the top \( d \) eigenvectors of \( P^s_k \).

The distributed FMS is obtained by distributed PCA at each iteration of FMS. Note that FMS uses randomized SVD to find only the top \( d \) principal components. For central processing and \( D \gg d \), we recommend applying a distributed randomized SVD algorithm [14]. For an ad hoc network, we are not aware of effective implementation of a distributed algorithm that can find only the top \( d \) principal components.

6. Numerical Experiments. This section tests the distributed algorithms proposed in this paper using both synthetic and real data. It is organized as follows: §6.1 describes the synthetic data model, §6.2 contains experiments on synthetic data sets and §6.3 contains experiments on real data sets.

In Algorithm 1 \( T_{GMS} = 30 \). In Algorithm 2 \( T_{CBGA} = 250 \) and the step sizes are specified constants. In FMS \( p = 1 \) and in all algorithms \( \delta = 10^{-10} \).

6.1. Synthetic Data Model for Distributed RSR. In §6.2 we use the following synthetic model to generate distributed RSR data. It depends on the following parameters: \( K, N^0, N^1, D, d \) and \( \sigma \). We create a connected graph with \( K \) nodes, and we randomly fix \( L \in G(D, d) \). For each node we sample \( N^1/K \) inliers from the \( d \)-dimensional Multivariate Normal distribution \( N(0, \mathbf{P}_L) \), where \( \mathbf{P}_L \) denotes the orthoprojector onto \( L \), with additive Gaussian noise \( N(0, \sigma^2 \mathbf{I}) \), where \( 0 \leq \sigma < 1 \). Furthermore, for each node we sample \( N^0/K \) outliers from the uniform distribution on \([0, 1]^D\). Note that the outliers are asymmetric.

6.2. Demonstration on Synthetic Data. We study the effect of the step size on the convergence rate of CBGA-GMS in §6.2.1 and the effect of the graph topology on the same convergence rate in §6.2.2. In §6.2.3 we compare the accuracy of a CADMM version of GMS with CBGA-GMS for fully connected network. In §6.2.4 we compare our proposed distributed RSR algorithms. In each experiment 50
random samples are generated according to the model of §6.1. The recovery error of the tested algorithm is averaged over the random 50 samples. For Figs. 1a-1c we further average the recovery error over the K processors to demonstrate the average rate of convergence. For Figs. 1d-1f we only present the results for one processor, as the results are identical among all processors.

**6.2.1. The Influence of the Step-size on the Convergence Rate.** We generate data according to the model of §6.1, where $K = 10$, $N^1 = 200$, $N^0 = 2,000$, $D = 50$, $d = 3$, and $\sigma = 0.1$. We report the average recovery error for CBGA-GMS as a function of the number of iterations for 6 different step sizes in Fig. 1a. They imply that the convergence rate increases with the step size. However, additional experiments, not reported in here, indicate that for a very large step size the algorithm does not converge. We also note that for large step sizes, the increase of the step size does not change the convergence rate significantly, for example, for step sizes 150 and 200 we see almost the same result, while the difference between convergence results is obvious between step sizes 10 and 30.

**6.2.2. The Influence of the Graph Topology on the Convergence Rate.** To check the effect of the network topology on the convergence rate we use three different networks, whose graphs are shown in Fig. 2. The graph in Fig. 2a is sparse and the graph in Fig. 2b is fully connected. The graph in Fig. 2c is obtained by arbitrarily generating a spanning tree with $K$ nodes and then randomly and independently connecting 2 nodes with probability $1/2$. We generate data according to the model of §6.1, where $K = 10$, $N^1 = 200$, $N^0 = 2,000$, $D = 50$, $d = 3$, $\sigma = 0.1$ and $\mu = 100$. The average recovery error as a function of the number of iterations for the 3 different networks is shown in Fig. 1b. The fully connected network has the fastest convergence and as the network gets sparser, the convergence rate decreases.

**6.2.3. Comparison of CADMM-GMS and CBGA-GMS.** We generate the data according to the model described in §6.1 with $K = 10$, $D = 50$, $d = 3$, $\sigma = 0.1$, $N^0 = 1,800$ and $N^1 = 200$. Using this data, we compare CBGA-GMS (for fully connected network) with CADMM-GMS (for a central processor) in Fig. 1c. For description of CADMM, independent of GMS, see §7 of [9]. To apply CADMM to GMS, we need to solve the same local optimization problem as for CBGA-GMS. Thus, the same theory works for CADMM-GMS. Fig. 1c demonstrates that the average recovery errors for CBGA-GMS and CADMM-GMS (after convergence) are very comparable, but CADMM-GMS converges slightly faster than CBGA-GMS. Therefore, if a central processor is available one could use CADMM-GMS and achieve faster convergence.

**6.2.4. Comparison of Distributed PCA, GMS, Reaper and FMS.** We compare the 3 proposed distributed RSR algorithms and distributed PCA (CBGA-PCA from §3.1) in different settings and report the results in Figures 1d-1f. Fig. 1d demonstrates how the inlier noise variance $\sigma$ effects the convergence of the four methods. We create the data according to the model described in §6.1, where $D = 50$, $d = 3$, $K = 5$, $N^0 = 3,000$ $N^1 = 1,000$ and $\sigma$ varies between 0 and 0.2 with increments of 0.01. In this figure, for all tested values of $\sigma$, CBGA-PCA performs the worst and distributed FMS performs the best. We notice that CBGA-GMS performs slightly better than distributed Reaper if $\sigma \leq 0.1$, otherwise, distributed Reaper performs slightly better. Another thing to notice is that as the inlier noise variance increases the error for distributed RSR methods also increase.

Figs. 1e and 1f demonstrate the influence of the outlier percentage on the average recovery error for the four methods with and without inlier noise. We generate data
(a) Influence of different step sizes on the convergence rate of CBGA-GMS

(b) Influence of the network topology on the convergence rate of CBGA-GMS

(c) CBGA-GMS vs CADMM-GMS

(d) Influence of inlier noise variance on distributed PCA, Reaper, GMS and FMS.

(e) Influence of outlier percentage on distributed PCA, Reaper, GMS, FMS; $\sigma = 0$.

(f) Influence of outlier percentage on distributed PCA, Reaper, GMS, FMS; $\sigma = 0.5$.

Fig. 1: Demonstration of properties of the distributed algorithms on synthetic data.
Fig. 2: Three types of connected networks with 8 nodes. Fig. 2a: sparsely connected network; Fig. 2b: fully connected network; and Fig. 2c: randomly connected network.

according to the model of §6.1, where \( D = 50, d = 3, K = 10, \sigma = 0 \) for Fig. 1e, \( \sigma = 0.5 \) for Fig. 1f, \( N^0 = 5,000 \) and \( N^1 \) varies between 50 and 1,050 with increments of 50. For both cases (\( \sigma = 0 \) and \( \sigma = 0.5 \)) and for all percentages of outliers, the recovery error for distributed FMS is the smallest one and that of CBGA-PCA is the largest one. We also note that if \( \sigma = 0 \) (inliers lie exactly on a \( d \)-dimensional subspace) and the percentage of outliers is less than 95%, the recovery error for FMS is close to 0, while for CBGA-GMS and distributed Reaper it increases with the outlier percentage. On the other hand, for \( \sigma = 0.5 \), when the outlier percentage is around 80% we get similar results for the three distributed RSR algorithms, but when the outlier percentage is around 90% FMS has a clear advantage. Figs. 1e and 1f also demonstrate that when the data is corrupted with outliers, distributed RSR algorithms perform significantly better than distributed PCA.

6.3. Real Data Experiments. Distributed RSR algorithms can be used as a preprocessing step for clustering, classification and regression. We apply our proposed distributed algorithms as a preprocessing step for two different tasks: linear regression, where we use the CTslices data set \((N = 53,500, D = 386)\) [19], and classification (multiclass SVM), where we use the Human Activity Recognition (HAR) data set \((N = 10,299, D = 561)\) [2,19]. For both data sets we apply initial centering by the mean and to ensure full-rank data in all processors we reduce dimension by PCA to \( D = 150 \); higher dimensions were also possible, but did not effect the results. We report the results for one of the processors as they are the same for all of them.

For the CTslices data, the algorithms are trained on 50,000 data points and tested on 3,500 data points. The training data is divided between 5 processors, each containing 10,000 data points. We apply CBGA-PCA, CBGA-GMS, distributed FMS and distributed Reaper to reduce the dimension of the data set to lie between 5 and 30. We then apply Ordinary Least Squares for linear regression in the reduced dimension. Fig. 3a reports the relative regression error for the different projected dimensions. The relative regression error is the regression error for the data with the reduced dimension divided by the relative error for the data in 150 dimensions. We notice that for almost all dimensions, the relative errors of distributed FMS and GMS are lower than those of distributed PCA, and the relative errors of distributed Reaper are either lower or comparable to those of distributed PCA.
For the HAR data, the algorithms are trained on 7,352 data points and tested on 2,947 data points. The training data is divided between 8 processors, each containing 919 data points. We apply CBGA-PCA, CBGA-GMS, distributed FMS and distributed Reaper to reduce the dimension of the data set to lie between 2 and 20. We then apply classification in the reduced dimensions. Fig. 3b reports classification error for the different projected dimensions. It demonstrates that in dimension 2, the distributed RSR algorithms, in particular, distributed FMS and GMS, have a clear advantage over distributed PCA. In other dimensions, distributed RSR algorithms perform at least as good as distributed PCA.

Acknowledgements. We thank Amit Singer for his helpful comments on the presentation of this work and his suggestion to demonstrate the ideas of CBGA-GMS on the problem of distributed computation of the geometric median. This work was supported by NSF awards DMS-09-56072 and DMS-14-18386 and the Feinberg Foundation Visiting Faculty Program Fellowship of the Weizmann Institute of Science.

Appendix A. Supplementary Details.

A.1. On the Minimizer of (7). We first state the main result of this section:

**Lemma 2.** If $X_k \subset \mathbb{R}^D$ is full rank and $A_k \in S^D$, then the minimizer of (7) is unique. Furthermore, there exists a unique $c' \in \mathbb{R}$ such that this minimizer is the unique solution of the following equation with $c = c'$

$$Q \left( \sum_{x \in X_k} xx^T \right) + \left( \sum_{x \in X_k} xx^T \right) Q + A_k = cI.$$  \hspace{1cm} (21)

Section A.1.1 states and proves a lemma about the solution of the above Lyapunov equation and §A.1.2 then uses this latter lemma to conclude Lemma 2. At last, §A.1.3 briefly discusses the computation of the minimizer of (7).

**A.1.1. Preliminary lemma.** We verify the following lemma.
Lemmas.

Lemma 3. If \( c \in \mathbb{R} \), \( X \in S^{D}_{++} \), and \( A \in S^{D} \), then the following Lyapunov equation

\[
QX + XQ + A = cI
\]

has a unique solution in \( Q \in S^{D} \). Furthermore, \( \text{tr}(Q) \) is an increasing linear function of \( c \) with slope \( \text{tr}(X^{-1})/2 \).

Proof. The existence and uniqueness of the solution of (22) is well-known [8, page 107]. We thus only need to show that \( \text{tr}(Q) \) is an increasing linear function of \( c \). Assume that \( Q_1 \) and \( Q_2 \) are the solutions of (22) corresponding to \( c_1 \) and \( c_2 \), that is,

\[
Q_1X + XQ_1 + A = c_1I \quad \text{and} \quad Q_2X + XQ_2 + A = c_2I.
\]

Subtracting the two equations in (23), results in

\[
(Q_1 - Q_2)X + X(Q_1 - Q_2) = (c_1 - c_2)I,
\]

whose unique solution is \( (Q_1 - Q_2)X + X(Q_1 - Q_2) = (c_1 - c_2)X^{-1}/2 \). By taking traces of both sides of the solution, we get that \( \text{tr}(Q_1) - \text{tr}(Q_2)/(c_1 - c_2) = \text{tr}(X^{-1})/2 > 0 \).

A.1.2. Proof of Lemma 2. Since \( X_k \) is full rank, \( \sum_{x \in X_k} xx^T \in S^{D}_{++} \). Hence the minimized function in (7) is strongly convex and its minimizer is unique.

We note that (21) is a Lyapunov equation in \( Q \). Lemma 3 implies that there is a unique value \( c' \) for which the unique solution of (21) has trace 1. We denote this solution by \( Q' \). Next, we show that \( Q' \) is the minimizer of (7). The following two facts:

\[
\sum_{x \in X_k} \|Qx_k\|^2 + \text{tr}(A_kQ) = \sum_{x \in X_k} \text{tr}(Qx_kx_k^TQ) + \text{tr}(A_kQ) \quad \text{for} \quad Q \in S \quad \text{and} \quad \text{tr}(Q) = 1 \quad \text{for} \quad Q \in S,
\]

imply the same minimizer for (7) and

\[
\min_{Q \in S} l(Q), \quad l(Q) = \sum_{x \in X_k} \text{tr}(Qx_kx_k^TQ) + \text{tr}(A_kQ) - c' \text{tr}(Q).
\]

Since \( l(Q) \) is convex on \( S \), we conclude that \( Q' \) minimizes (25) by showing that the derivative of \( l(Q) \) at \( Q' \), when restricted to \( S \), is 0:

\[
\frac{d}{dQ} l(Q) \bigg|_{Q=Q'} = Q' \left( \sum_{x \in X_k} xx^T \right) + \left( \sum_{x \in X_k} xx^T \right) Q' + A_k - c'I = 0.
\]

A.1.3. Computing the Minimizer of (7). In view of Lemma 3 we compute \( c' \) and the corresponding solution of (23) as follows. We solve (21) with \( c = 0 \) to obtain \( Q_* \in S^{D} \). We then use \( \text{tr}(Q_*) \) and the slope \( \text{tr}(X^{-1})/2 \), where \( X = \sum_{x \in X_k} xx^T \), to find \( c' \). Therefore, computing this minimizer requires computing \( X \), which costs \( O(N_{\text{max}} \times D^2) \), and solving two Lyapunov equations, which costs \( O(D^3) \) (see [4]).

A.2. Proof of Lemma 1. Let \( X = \sum_{i=1}^{N} x_i x_i^T / (2 \max(\|Qx_i\|, \delta)) \) and note that \( X \in S^{D}_{++} \). This observation and Lemma 3 imply that there is a unique value \( c \in \mathbb{R} \) for which (17) has a unique solution in \( S \). We will show that \( c > \lambda_1(A) \), equivalently \( A - cI \leq 0 \), and thus in view of [8, page 107], this solution is in \( S^{D}_{++} \).

To get this estimate, we rewrite (17) as \( P + XPX^{-1} + AX^{-1} = cX^{-1} \). Applying trace to both sides and using the following facts: \( \text{tr}(P) = 1 \), \( \text{tr}(XPX^{-1}) = \text{tr}(X^{-1}XP) = 1 \) and \( \text{tr}(AX^{-1}) \geq \lambda_D(A) \text{tr}(X^{-1}) \) yields \( c \geq 2/\text{tr}(X^{-1}) + \lambda_D(A) \).

Let \( X_* = \sum_{i=1}^{N} x_i x_i^T / (2 \max(\|x_i\|, \delta)) \), since \( Q \in S^{D}_{++} \) and \( \max(\|Qx_i\|, \delta) \leq \max(\|x_i\|, \delta) \) for \( 1 \leq i \leq N \), \( X - X_* \in S^{D}_{++} \), which implies that \( X_* - X^{-1} \in S^{D}_{++} \). Combining the last result with (16), \( \|A\|_2 > \lambda_1(-A) \) and the estimate of \( c \) we obtain that \( c \geq 2/\text{tr}(X_*^{-1}) + \lambda_D(A) \geq 2/\|A\|_2 - \lambda_1(-A) \geq \lambda_1(A) \).

The last statement of the lemma is a direct application of Lemma 3.
A.3. Guaranteeing (16). In view of Lemma 1, we require that condition (16) holds at each iteration of Algorithm 2 and each node \( k \). The following lemma shows that a choice of a sufficiently small step size \( \mu \) guarantees this requirement. After verifying this lemma, we discuss weaker restrictions on the step size as well as a weaker practical version of condition (16).

**Lemma 4.** If \( \{X_k\}_{k=1}^K \subset \mathbb{R}^D \) are data sets distributed at \( K \) nodes, \( n \in \mathbb{N} \) and

\[
\mu \leq \frac{1}{n \cdot \max_{1 \leq k \leq K} |E_k| \cdot \text{tr} \left( \sum_{x \in X_k} \frac{x x^T}{\max(\|x\|, \delta)} \right)^{-1}},
\]

then at each iteration \( s \leq n \) of Algorithm 2 and node \( k \), \( A_k^s \) satisfies condition (16).

**Proof.** We estimate the LHS of (16) at iteration \( s \) as follows:

\[
\|A_k^s\|_2 = \left\| \sum_{m \in E_k} c_m A_m^{s-1} \right\|_2 \leq \sum_{m \in E_k} \|A_m^{s-1}\|_2.
\]

In order to evaluate \( \|A_m^{s-1}\|_2 \) for \( 1 \leq m \leq M \), we apply (19) and basic inequalities:

\[
\|A_m^{s-1}\|_2 = \|A_m^{s-1} + \mu(c_m Q_k^s - c_m Q_q^s)\|_2 \leq \|A_m^{s-1}\|_2 + \mu \|Q_k^s - Q_q^s\|_2 \leq \|A_m^{s-1}\|_2 + \mu \max\{\|Q_k^s\|_2, \|Q_q^s\|_2\} \leq \|A_m^{s-1}\|_2 + \mu \leq \cdots \leq s\mu \leq n\mu.
\]

Combining (27) and (28) results in

\[
\|A_k^s\|_2 \leq \sum_{m \in E_k} \|A_m^{s-1}\|_2 \leq |E_k| n\mu \leq 1 / \text{tr} \left( \sum_{x \in X_k} \frac{x x^T}{2 \max(\|x\|, \delta)} \right)^{-1}.
\]

The above estimate suggests that in practice one may apply several iterations with the same fixed step size and then multiply the step size by a factor lower than the inverse of the number of iterations and proceed this way. Nevertheless, this estimate represents a worst-case scenario and typically we expect an improved one. Indeed, first note that condition (16) represents a worse-case scenario. In the proof of Lemma 1 we used the worst-case estimate \( \|Q\| \leq 1 \). However, typically \( \|Q\| \sim 1/D \). This will introduce a multiplicative factor \( D \) for the RHS of (16) and thus of (26). Second, in (28) we used the estimate \( \|Q_k^s - Q_q^s\|_2 \leq \max\{\|Q_k^s\|_2, \|Q_q^s\|_2\} \leq 1 \). However, typically for \( Q_k^s, Q_q^s \in S \cap S_{D+1}^D, \max\{\|Q_k^s\|_2, \|Q_q^s\|_2\} \sim 1/D \). This observation introduces another multiplicative factor \( D \) for the RHS of (26). Third of all, we note that for sufficiently small step sizes the gradient descent gets closer to the solution, that is, \( \|Q_k^s - Q_q^s\|_2 \to 0 \), for \( 1 \leq k, q \leq K \). However, we used \( 1/D \) as an upper bound for \( \|Q_k^s - Q_q^s\|_2 \). At last, we comment that while the above analysis aims to guarantee that at each iteration the solution is in \( S \cap S_{D+1}^D \) (since (16) guarantees this), in practice it is not a main concern for small step sizes and large number of iterations. Indeed, the solution of (4) coincides with the solution of GMS for the total data, which is in \( S \cap S_{D+1}^D \). Thus, by choosing the step size small enough we will always converge to the solution.

**A.4. Proof of Theorem 1.** We establish an auxiliary lemma in §A.4.1 and conclude Theorem 1 in §A.4.2 by following ideas of [11,32] and using this lemma.
A.4.1. Preliminary Proposition. We first apply Lemma 3 to define the mapping $T_A(Q)$ and then establish the continuity of $T_A(Q)$ in $S^D_{++}$.

**Definition 2** (The mapping $T_A(Q)$). If $\{x_i\}_{i=1}^N \subset \mathbb{R}^D$, $\delta > 0$, $Q \in S^D_{++} \cap S$ and $A \in S^D$ with $\text{tr}(A) = 0$, then $T_A(Q)$ is the solution of the following equation in $P$

$$
(30) \quad P \left( \sum_{i=1}^N \frac{x_i x_i^T}{\max(\|Qx_i\|, \delta)} \right) + \left( \sum_{i=1}^N \frac{x_i x_i^T}{\max(\|Qx_i\|, \delta)} \right) P + A = cI,
$$

where $c = c(Q) \in \mathbb{R}$ is uniquely chosen so that the solution has trace $1$.

**Lemma 5.** Assume a sequence $\{Q^ t\}_{t \in \mathbb{N}} \subset S^D_{++} \cap S$, $A \in S^D$ with $\text{tr}(A) = 0$, $\{x_i\}_{i=1}^N \subset \mathbb{R}^D$ and $\delta > 0$. If $Q^ t \to Q$, then $T_A(Q^ t) \to T_A(Q)$.

**Proof.** For $t \in \mathbb{N}$, let $P^ t = T_A(Q^ t)$ be the trace one solution of (30) with $Q = Q^ t$ and $c = c^ t$. Let $\hat{P} = T_A(Q)$ be the trace one solution of (30) with $Q = Q$ and $c = \hat{c}$.

We need to prove that $P^ t \to \hat{P}$ as $t \to \infty$. We write (30) with $P^ t, Q^ t$ and $c^ t$ as

$$
(31) \quad P^ t \left( \sum_{i=1}^N \frac{x_i x_i^T}{\max(\|Q^ t x_i\|, \delta)} \right) + \left( \sum_{i=1}^N \frac{x_i x_i^T}{\max(\|Q^ t x_i\|, \delta)} \right) P^ t + A = c^ t I.
$$

Note that $R^ t := \sum_{i=1}^N \frac{x_i x_i^T}{\max(\|Q^ t x_i\|, \delta)} \to \hat{R} := \sum_{i=1}^N \frac{x_i x_i^T}{\max(\|Qx_i\|, \delta)}$ as $t \to \infty$. Also observe that for $Q = Q, c = \hat{c}$ and $T_A(Q^ t) = P^ t$, (30) has the form

$$
(32) \quad P^ t - c^ t R^ t + R^ t P^ t + A = c^ t I.
$$

By subtracting $c^ t I$ from both sides of (32) and rewriting $c^ t I = c^ t R^{t-1} R^ t/2 + R^ t c^ t R^{t-1}/2$, (32) becomes $(P^ t - c^ t R^{t-1}/2) R^ t + R^ t (P^ t - c^ t R^{t-1}/2) + A = 0$. Similarly, $(\hat{P} - \hat{c} \hat{R}^{t-1}/2) \hat{R} + \hat{R} (\hat{P} - \hat{c} \hat{R}^{t-1}/2) + A = 0$. Since $A$ is fixed and $R^ t \to \hat{R}$ as $t \to \infty$, it follows from the last two expressions that

$$
(33) \quad P^ t - c^ t R^{t-1}/2 \to \hat{P} - \hat{c} \hat{R}^{t-1}/2 \text{ as } t \to \infty.
$$

By taking the trace of both sides of (33) and using the facts that $\text{tr}(P^ t) = \text{tr}(\hat{P}) = 1$ and $\hat{R} \to R$ as $t \to \infty$, we get that $c^ t \to \hat{c}$ and consequently $P^ t \to \hat{P}$ as $t \to \infty$. \qed

A.4.2. Conclusion of Theorem 1. We divide the proof of Theorem 1 into the following steps suggested in [32].

**Step 1:** The majorizing function $H$ and its minimizer. Let $H^k_\delta$ denote the following function

$$
(34) \quad H^k_\delta(Q, Q^*) = \sum_{x \in X_k} \left( \frac{\|Qx\|^2}{2 \max(\|Q^*x\|, \delta)} + \frac{\max(\|Q^*x\|, \delta)}{2} \right) + \text{tr}(QA_k).
$$

We show next that $H^k_\delta$ majorizes $G^k_\delta$, that is,

$$
(35) \quad H^k_\delta(Q, Q) = G^k_\delta(Q) \text{ and } G^k_\delta(Q) \leq H^k_\delta(Q, Q^*).
$$

The above equality is immediate. To prove the above inequality we define

$$
G^k_\delta(x, Q) = \begin{cases} 
\frac{\|Qx\|}{2\delta}, & \text{if } \|Qx\| \geq \delta; \\
\frac{\|Qx\|^2}{4\delta} + \frac{\delta}{2}, & \text{if } \|Qx\| < \delta,
\end{cases}
$$

$$
H^k_\delta(x, Q, Q^*) = \frac{\|Qx\|^2}{2 \max(\|Q^*x\|, \delta)} + \frac{\max(\|Q^*x\|, \delta)}{2}.
$$

We show that $G^k_\delta(x, Q) \leq H^k_\delta(x, Q, Q^*)$ by considering four complementing cases:
Case 1: $\|Qx\| \geq \delta$ and $\|Q^*x\| \geq \delta$. In this case
\[
G_k^\delta(x, Q) = \frac{\|Qx\|}{\|Q^*x\|} = \frac{\|Qx\|}{\|Q^*x\|} \leq \frac{\|Qx\|^2 + \|Q^*x\|^2}{2\|Q^*x\|} = H_k^\delta(x, Q, Q^*).
\]

Case 2: $\|Qx\| \geq \delta$ and $\|Q^*x\| < \delta$. We conclude the desired property as follows
\[
0 \leq (\|Qx\| - \delta)^2 = \|Qx\|^2 - 2\|Qx\|\delta + \delta^2 = \delta (H_k^\delta(x, Q, Q^*) - G_k^\delta(x, Q)).
\]

Case 3: $\|Qx\| < \delta$ and $\|Q^*x\| \geq \delta$. In this case
\[
G_k^\delta(x, Q) - H_k^\delta(x, Q, Q^*) = \frac{1}{2} \left( \frac{\|Qx\|^2}{\delta} + \delta - \frac{\|Q^*x\|^2}{\|Q^*x\|} - \|Q^*x\| \right) = \frac{\|Q^*x\| - \delta}{\delta\|Q^*x\|} \left( \frac{\|Qx\|^2}{\delta} - 1 \right) \leq 0.
\]

Case 4: $\|Qx\| < \delta$ and $\|Q^*x\| < \delta$. Then $G_k^\delta(x, Q) = H_k^\delta(x, Q, Q^*)$.

We thus conclude (35) as follows
\[
G_k^\delta(Q) = \sum_{x \in A_k} G_k^\delta(x, Q) + \text{tr}(QA_k) \leq H_k^\delta(x, Q, Q^*) + \text{tr}(QA_k) = H_k^\delta(Q, Q^*).
\]

Next, we claim that the minimizer of $H_k^\delta(Q, Q_k^t)$ over all $Q \in S$ is $Q_k^{t+1}$. First we note that since the data satisfies the two-subspaces criterion and since $\text{tr}(A_kQ_k)$ is a linear function, then according to Theorem 2 of [32], $H_k^\delta(Q, Q_k^*)$ is strictly convex over $Q \in S$. We further note that for $Q \in S$, $H_k^\delta(Q, Q^*) = H_k^\delta(Q, Q^*)$, where
\[
(37) \quad \tilde{H}_k^\delta(Q, Q^*) = \sum_{x \in A_k} \left( \frac{\text{tr}(Qxx^TQ)}{2\max(\|Qx\|, \delta)} + \frac{\max(\|Q^*x\|, \delta)}{2} \right) + \text{tr}(QA_k).
\]

Therefore, the minimizers over $S$ of $H_k^\delta(Q, Q_k^t)$ and $H_k^\delta(Q, Q_k^t)$ are the same. We compute the derivative of the latter term w.r.t. $Q$ as follows:
\[
(38) \quad \frac{d}{dQ} \left( \tilde{H}_k^\delta(Q, Q_k^t) - c_k\text{tr}(Q) \right) \bigg|_{Q=Q_k^{t+1}} = \frac{1}{2} \left( Q_k^{t+1} \sum_{x \in A_k} \frac{xx^T}{\max(\|Q_k^{t+1}x\|, \delta)} + \sum_{x \in A_k} \frac{xx^T}{\max(\|Q_k^{t}x\|, \delta)} Q_k^{t+1} \right) + A_k - c_kI = 0.
\]

The last equation follows from the definition of $Q_k^{t+1}$ (see (15)). Combining this with the fact that $H^\delta(Q, Q_k)$ is strictly convex when restricted to $Q \in S$, we conclude that $Q_k^{t+1}$ is the unique minimizer of $H^\delta(Q, Q_k)$ for $Q \in S$.

**Step 2:** Convergence of $\{G_k^\delta(Q_k^t)\}_{t \in \mathbb{N}}$. We first note that $G_k^\delta(Q)$ is bounded from below on $S$. Indeed, $G_k^\delta(Q) \geq \text{tr}(QA_k) \geq \text{tr}(Q) \times \min \text{eig}(A_k) = \min \text{eig}(A_k)$.

Next, we show that $G_k^\delta(Q_k^t)$ decreases with $t$. By using (35) and the fact that $Q_k^{t+1}$ is the minimizer of $H_k^\delta(Q, Q_k^t)$ for $Q \in S$, we get that
\[
G_k^\delta(Q_k^{t+1}) \leq H_k^\delta(Q_k^{t+1}, Q_k^t) \leq H_k^\delta(Q_k, Q_k^t) = G_k^\delta(Q_k^t).
\]

Since $\{G_k^\delta(Q_k^t)\}_{t \in \mathbb{N}}$ is bounded from below and decreases, it converges.

**Step 3:** $\|Q_k^t - Q_k^{t+1}\| \to 0$ as $t \to \infty$. It follows from (38) and the fact that $Q_k^t - Q_k^{t+1} \in S^{D'}$ has trace 0, that
\[
\text{tr} \left( Q_k^{t+1} \sum_{x \in A_k} \frac{xx^T}{\max(\|Q_k^{t+1}x\|, \delta)} + \sum_{x \in A_k} \frac{xx^T}{\|Q_k^{t}x\|, \delta)} Q_k^{t+1} + 2A_k)(Q_k^t - Q_k^{t+1}) \right) = 0.
\]
Simplifying the above equation, we get that

\[ \text{tr} \left( A_k \left( Q_k^t - Q_k^{t+1} \right) \right) = - \text{tr} \left( Q_k^{t+1} \sum_{x \in X_k} \frac{x x^T}{\max(\|Q_k^t x\|, \delta)} (Q_k^t - Q_k^{t+1}) \right) = \text{tr} \left( \sum_{x \in X_k} \frac{Q_k^{t+1} x x^T (Q_k^t - Q_k^{t+1})}{\max(\|Q_k^t x\|, \delta)} \right) = \sum_{x \in X_k} \frac{x^T Q_k^{t+1} (Q_k^t - Q_k^{t+1}) x}{\max(\|Q_k^t x\|, \delta)}. \]

It follows from (39) and (34) that

\[ G_k^s(Q_k^t) - G_k^s(Q_k^{t+1}) \geq H_k^s(Q_k^t, Q_k^t) - H_k^s(Q_k^{t+1}, Q_k^t) = \frac{1}{2} \sum_{x \in X_k} \left( \frac{\|Q_k^t x\|^2 - \|Q_k^{t+1} x\|^2}{\max(\|Q_k^t x\|, \delta)} \right) + \text{tr}(\delta Q_k^t - Q_k^{t+1}) A = \frac{1}{2} \sum_{x \in X_k} \left( \frac{x^T (Q_k^t)^2 x - x^T (Q_k^{t+1})^2 x}{\max(\|Q_k^t x\|, \delta)} \right) + \text{tr}(\delta Q_k^t - Q_k^{t+1}) A. \]

The combination of (40) and (41) yields

\[ \sum_{x \in X_k} \frac{x^T Q_k^{t+1} (Q_k^t - Q_k^{t+1}) x}{\max(\|Q_k^t x\|, \delta)} = \frac{1}{2} \sum_{x \in X_k} \frac{\|(Q_k^t - Q_k^{t+1}) x\|^2}{\max(\|Q_k^t x\|, \delta)} \geq 0. \]

Since \( \{G(Q_k^t)\}_{t \in \mathbb{N}} \) converges, (42) implies that

\[ \sum_{x \in X_k} \frac{\|(Q_k^t - Q_k^{t+1}) x\|^2}{\max(\|Q_k^t x\|, \delta)} \to 0 \text{ as } t \to \infty \]

and consequently (using the fact that \( \text{Span}\{x \}_{x \in X_k} = \mathbb{R}^D \)):

\[ \|Q_k^t - Q_k^{t+1}\| \to 0 \text{ as } t \to \infty. \]

**Step 4:** Convergence of \( \{Q_k^t\}_{t \in \mathbb{N}} \) to the minimizer of \( G_k^s(Q) \). The sequence \( \{Q_k^t\}_{t \in \mathbb{N}} \) lies in the compact set of positive semi-definite matrices with trace 1. By Bolzano-Weierstrass theorem, \( \{Q_k^t\}_{t \geq 1} \) has a converging subsequence. Let \( \tilde{Q}_k \) denote the limit of the subsequence. We show that

\[ \tilde{Q}_k = \arg \min_{Q \in \mathcal{S}} G_k^s(Q). \]

By Lemma 5 and the fact that the limits of \( G_k^s(Q_k^t) \) and \( G_k^s(Q_k^{t+1}) \equiv G_k^s(T_A(Q_k^t)) \) are the same, we conclude that \( G_k^s(Q_k^t) = G_k^s(T_A(Q_k^t)) \). Combining this result with (39) we get that \( H_k^s(T_A(Q_k^t), Q_k^t) = H_k^s(Q_k^t, Q_k^t) \). Since \( T_A(Q_k^t) \) is the unique minimizer of \( H_k^s(Q, Q_k^t) \) among all \( Q \in \mathcal{S} \) we get that \( T_A(Q_k^t) = Q_k^t \). That is, \( \tilde{Q}_k \) is the unique minimizer of \( H_k^s(Q, \tilde{Q}_k) \) and \( \tilde{Q}_k \) among all \( Q \in \mathcal{S} \) and thus the directional derivatives of \( H_k^s(Q, \tilde{Q}_k) \) with respect to \( Q \) restricted to \( \mathcal{S} \) are 0. Hence,
where for the first equality we used (46) and for the last equality we used that
\( d = \frac{d}{dQ} H_k^\delta(Q, \hat{Q}_k)|_{Q=\hat{Q}_k} = cI \). This implies that

\[
\text{tr} \left( \left( \frac{d}{dQ} H_k^\delta(Q, \hat{Q}_k)|_{Q=\hat{Q}_k} \right) \left( P - \hat{Q}_k \right)^T \right) = 0 \quad \text{and thus there exists } c \in \mathbb{R} \text{ such that }
\]

\[
d = \frac{d}{dQ} H_k^\delta(Q, \hat{Q}_k)|_{Q=\hat{Q}_k} = cI .
\]

We note that [11] states the result for vector-valued functions, which can be easily

\( \text{tr} \left( \left( \frac{d}{dQ} \tilde{G}_k^\delta(Q)|_{Q=\hat{Q}_k} \right) \left( P - \hat{Q}_k \right)^T \right) = 0,
\]

where

\[
\tilde{G}_k^\delta(Q) = \sum_{x \in X_k : \|Qx\| \geq \delta} \sqrt{\text{tr}(Qxx^T)} + \sum_{x \in X_k : \|Qx\| < \delta} \left( \frac{\text{tr}(Qxx^T)}{2\delta} + \frac{\delta}{2} \right) + \text{tr}(QA_k).
\]

The directional derivatives of \( \frac{d}{dQ} \tilde{G}_k^\delta(Q)|_{Q=\hat{Q}_k} \) restricted to \( S \) are

\[
\text{tr} \left( \left( \frac{d}{dQ} \tilde{G}_k^\delta(Q)|_{Q=\hat{Q}_k} \right) \left( P - \hat{Q}_k \right)^T \right) = 0,
\]

where for the first equality we used (46) and for the last equality we used that \( P, \hat{Q}_k \in S \) and thus \( \text{tr}(P) = \text{tr}(\hat{Q}_k) = 1 \). Equation (47) and the fact that \( G_k^\delta(Q) = \tilde{G}_k^\delta(Q) \) for

\( Q \in S \) imply (45). Finally, combining (44), (45), the definition of \( \tilde{Q}_k \) and [24, Theorem 2.1], we conclude that \( Q_k \to \check{Q}_k \) as \( t \to \infty \).

Step 5: \textit{r-linear Convergence}. The proof of \( r \)-linear convergence of \( \check{Q}_k \) follows from Theorem 6.1 of [11] (similarly to the proof of Theorem 11 of [32]). To show that the conditions of the theorem are satisfied we just need to check that the functions \( G \) and \( H \) satisfy Hypotheses 4.1 and 4.2 of [11], see proof of Theorem 6.1 in there. However, \( G \) and \( H \) of this work are parallel to \( F \) and \( H \) of [11], respectively. We note that [11] states the result for vector-valued functions, which can be easily generalized for matrix-valued functions. Since \( \check{Q}_k \) converges, it is enough to show that Hypotheses 4.1 and 4.2 hold for some local neighborhood \( B(\hat{Q}_k, \epsilon) \) of \( \hat{Q}_k \), for some \( \epsilon > 0 \). Conditions 1 and 3 of Hypothesis 4.1 are easy to check, since \( G \) is twice differentiable on \( B(\hat{Q}_k, \epsilon) \) and \( G \) is bounded from below (as we have already shown). There is no need to check condition 2, since \( Q \) is restricted to \( H \). To verify condition 1 of Hypothesis 4.2 we need to show that

\[
H_k^\delta(Q_1, Q_2) = G_k^\delta(Q_2) + \text{tr}((Q_1 - Q_2)^TC(Q_2)) (Q_1 - Q_2).
\]

To prove (48), we write its RHS as follows:

\[
\sum_{x \in X_k : \|Q_2x\| \geq \delta} \|Q_2x\| + \sum_{x \in X_k : \|Q_2x\| < \delta} \left( \frac{\|Q_2x\|^2}{2\delta} + \frac{\delta}{2} \right) + \text{tr}(A_k) + \text{tr}((Q_1 - Q_2)^T \left( \frac{1}{2} \left( Q_2 \sum_{x \in X_k} \frac{xx^T}{\|Q_2x\|, \delta} + \sum_{x \in X_k} \frac{xx^T}{\|Q_2x\|, \delta} Q_2 \right) + A_k \right) + \text{tr}((Q_1 - Q_2)^T \frac{1}{2} C(Q_2)(Q_1 - Q_2)).
\]
By setting $C(Q) = \sum_{x \in X_k} x^T x / \max(\|Qx\|, \delta)$, the above equation becomes

$$
\sum_{x \in X_k} \frac{\|Q_2 x\|}{\max(\|Q_2 x\|, \delta)} + \sum_{x \in X_k} \frac{\|Q_2 x\|^2}{2\delta} + \frac{\delta}{2} + \text{tr}(Q_2 A_k) +
$$

$$
\text{tr}\left( (Q_1 - Q_2)^T Q_2 \sum_{x \in X_k} \frac{x x^T}{\max(\|Q_2 x\|, \delta)} + A_k \right) +
$$

$$
\text{tr}\left( (Q_1 - Q_2)^T \sum_{x \in X_k} \frac{xx^T}{2\max(\|Q_2 x\|, \delta)} (Q_1 - Q_2) \right) = \sum_{x \in X_k, \|Q_2 x\| \geq \delta} \|Q_2 x\| +
$$

$$
\sum_{x \in X_k, \|Q_2 x\| < \delta} \frac{\|Q_2 x\|^2}{2\delta} + \frac{\delta}{2} + \text{tr}(Q_1 A_k) - \sum_{x \in X_k} \frac{\|Q_2 x\|^2}{\max(\|Q_2 x\|, \delta)} +
$$

$$
\sum_{x \in X_k} \frac{\|Q_2 x\|^2}{2\max(\|Q_2 x\|, \delta)} + \sum_{x \in X_k} \frac{\|Q_1 x\|^2}{2\max(\|Q_2 x\|, \delta)} = \sum_{x \in X_k} \frac{\|Q_1 x\|^2}{2\max(\|Q_2 x\|, \delta)} +
$$

$$
\sum_{x \in X_k} \frac{\max(\|Q_2 x\|, \delta)}{2} + \text{tr}(Q_1 A_k) = H(Q_1, Q_2).
$$

That is, condition 1 of Hypothesis 4.2 is verified, conditions 2 and 3 follow directly from the definition of $C(Q)$ and condition 4 follows from (36). □

REFERENCES

[1] A. Aduroja, I. D. Schizas, and V. Maroulas. Distributed principal components analysis in sensor networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 5850–5854, May 2013.

[2] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *21st European Symposium on Artificial Neural Networks ESANN, Bruges, Belgium*, 2013.

[3] Z. Bai, H. C. Raymond, and T. L. Franklin. Principal component analysis for distributed data sets with updating. In *Proceedings of International workshop on Advanced Parallel Processing Technologies (APPT)*, 2005.

[4] R. H. Bartels and G. W. Stewart. Solution of the matrix equation AX+XB=C [F4] (algorithm 432). *Commun. ACM*, 15(9):820–826, 1972.

[5] A. Bertrand and M. Moonen. Consensus-based distributed total least squares estimation in ad hoc wireless sensor networks. *IEEE Trans. Signal Processing*, 59(5):2320–2330, 2011.

[6] D. P. Bertsekas. *Convex analysis and optimization*. Athena Scientific, Belmont, MA, 2003. With Angelia Nedić and Asuman E. Ozdaglar.

[7] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation: numerical methods*, volume 23. Prentice Hall, 1989.

[8] R. Bhatia and L. Elsner. Positive linear maps and the Lyapunov equation. In I. Gohberg and H. Langer, editors, *Linear Operators and Matrices: Advances and Applications*, pages 107–120. Birkhäuser, Basel, 2002.

[9] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, January 2011.

[10] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, Cambridge, 2004.

[11] T. Chan and P. Mulet. On the convergence of the lagged diffusivity fixed point method in total variation image restoration. *SIAM Journal on Numerical Analysis*, 36(2):354–367, 1999.

[12] J. Chen and A. H. Sayed. Diffusion adaptation strategies for distributed optimization and learning over networks. *IEEE Transactions on Signal Processing*, 60(8):4289–4305, 2012.

[13] P. Forero, A. Cano, and G. Giannakis. Consensus-based distributed support vector machines. *J. Mach. Learn. Res.*, 11:1663–1707, 2010.

[14] N. Halko, P. Martinsson, and J. Tropp. Finding structure with randomness: Probabilistic
algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[15] B. Johansson, C.M. Carretti, and M. Johansson. On distributed optimization using peer-to-peer communications in wireless sensor networks. In *Sensor, Mesh and Ad Hoc Communications and Networks*, pages 497–505, June 2008.

[16] S. M. Kakade, S. Shalev-Shwartz, and A. Tewari. Regularization techniques for learning with matrices. *J. Mach. Learn. Res.*, 13:1865–1890, 2012.

[17] G. Lerman, M. McCoy, J. Tropp, and T. Zhang. Robust computation of linear models by convex relaxation. *Foundations of Computational Mathematics*, 15(2):363–410, 2015.

[18] Y. Liang, M. Balcan, V. Kanchanapally, and D. Woodruff. Improved distributed principal component analysis. In *Advances in Neural Information Processing Systems*, pages 3113–3121, 2014.

[19] M. Lichman. UCI Machine Learning Repository, 2013.

[20] G. Mateos, J. Bazerque, and G. Giannakis. Distributed sparse linear regression. *IEEE Transactions on Signal Processing*, 10(58):5262–5276, 2010.

[21] T. Maunu and Lerman G. Fast algorithm for robust subspace recovery. arXiv:1406.6145, 2014.

[22] Z. Meng, A. Wiesel, and A. Hero III. Distributed principal component analysis on networks via directed graphical models. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 2877–2880. IEEE, 2012.

[23] A. Nedic and A. Ozdaglar. Cooperative distributed multi-agent optimization. In *Convex Optimization in Signal Processing and Communications*. Cambridge University Press, 2010.

[24] M. Ostrowski. *Solution of equations and systems of equations*. Pure and applied mathematics. Academic Press, 1966.

[25] H. Qi, T. Wang, and D. Birdwell. *Global Principal Component Analysis for Dimensionality Reduction in Distributed Data Mining*, chapter 19, pages 327–342. CRC Press, 2004.

[26] Y. Qu, G. Ostrouchov, N. Samatova, and A. Geist. Principal component analysis for dimension reduction in massive distributed data sets. In *SIAM International Conference on Data Mining*, 2002.

[27] R. T. Rockafellar. *Convex analysis*. Princeton Mathematical Series, No. 28. Princeton University Press, Princeton, N.J., 1970.

[28] T. Sarlos. Improved approximation algorithms for large matrices via random projections. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS’06)*, pages 143–152, Oct 2006.

[29] I. Schizas, A. Ribeiro, and G. Giannakis. Consensus in ad hoc wsn with noisy linkspart i: Distributed estimation of deterministic signals. *Signal Processing, IEEE Transactions on*, 56(1):350–364, 2008.

[30] M. Valcarcel, P. Belanovic, and S. Zazo. Consensus-based distributed principal component analysis in wireless sensor networks. In *11th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5. IEEE, 2010.

[31] E. Weiszfeld. Sur le point pour lequel la somme des distances de n points donne est minimum. *Tokohoku Mathematical Journal*, 43:355 – 386, 1937.

[32] T. Zhang and G. Lerman. A novel M-estimator for robust pca. *J. Mach. Learn. Res.*, 15(1):749–808, January 2014.

[33] H. Zhu, A. Cano, and G. Giannakis. Distributed consensus-based demodulation: algorithms and error analysis. *IEEE Transactions on Wireless Communications*, 9(6):2044–2054, 2010.