SIAM J. MATH. DATA SCI.
Vol. 3, No. 4, pp. 1067–1092

Communication-Efficient Distributed Eigenspace Estimation[*]

Vasileios Charisopoulos[†], Austin R. Benson[‡], and Anil Damle[‡]

Abstract. Distributed computing is a standard way to scale up machine learning and data science algorithms to process large amounts of data. In such settings, avoiding communication amongst machines is paramount for achieving high performance. Rather than distribute the computation of existing algorithms, a common practice for avoiding communication is to compute local solutions or parameter estimates on each machine and then combine the results; in many convex optimization problems, even simple averaging of local solutions can work well. However, these schemes do not work when the local solutions are not unique. *Spectral methods* are a collection of such problems, where solutions are orthonormal bases of the leading invariant subspace of an associated data matrix. These solutions are only unique up to rotation and reflections. Here, we develop a communication-efficient distributed algorithm for computing the leading invariant subspace of a data matrix. Our algorithm uses a novel alignment scheme that minimizes the Procrustean distance between local solutions and a reference solution and only requires a single round of communication. For the important case of principal component analysis (PCA), we show that our algorithm achieves a similar error rate to that of a centralized estimator. We present numerical experiments demonstrating the efficacy of our proposed algorithm for distributed PCA as well as other problems where solutions exhibit rotational symmetry, such as node embeddings for graph data and spectral initialization for quadratic sensing.

Key words. distributed computing, spectral methods, nonconvex optimization, principal component analysis, statistics

AMS subject classifications. 62-08, 65F15, 65F55, 68Q87

DOI. 10.1137/20M1364862

1. Overview and background.

The paradigm of distributed computing, where data collection and/or computation for a fixed task happens on several interconnected machines, is by now standard in machine learning and data science [1]. Typically, each machine holds its own data points or samples and a global solution is approximated using only local computation and communication between machines. Since communication is the bottleneck operation [5, 59], it is highly desirable to avoid multiple rounds of communication (i.e., multiple instances where machines broadcast or exchange information with each other). One of several flavors of distributed computing, which is the focus of this paper, is *federated learning* [33, 34]. In federated learning, local compute nodes communicate local information to a central "coordinator," though there are several other configurations used in distributed computing. When

[*]Received by the editors September 10, 2020; accepted for publication (in revised form) June 28, 2021; published electronically October 5, 2021.
https://doi.org/10.1137/20M1364862
Funding: This research was supported by NSF award DMS-1830274. The second author was supported by ARO award W911NF19-1-0057, ARO MURI, and JPMorgan Chase & Co.

[†]Department of Operations Research & Information Engineering, Cornell University, Ithaca, NY 14853 USA (vc333@cornell.edu).

[‡]Department of Computer Science, Cornell University, Ithaca, NY 14853 USA (arb@cs.cornell.edu, damle@cornell.edu).

the underlying problem is "simple" (e.g., convex), combining local information and solutions is relatively well studied, and even the simplest schemes (such as one-shot averaging of local solutions) often work well with minimal communication costs [58]. However, many problems are not amenable to such schemes.

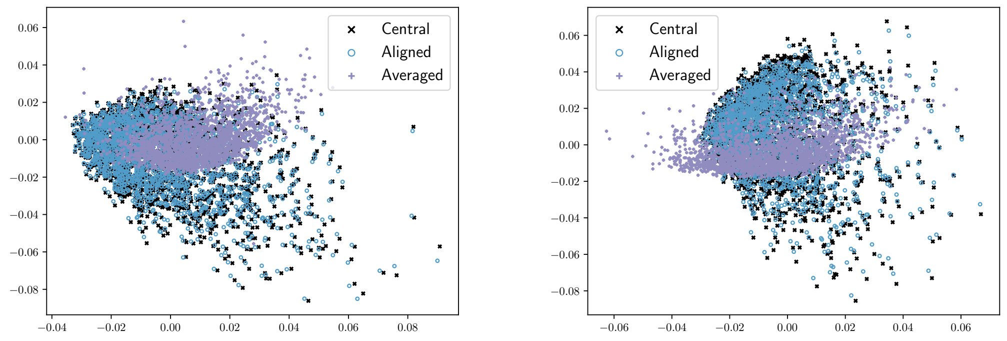
Consider the model problem of distributed computation of the first principal component in principal component analysis (PCA). We assume that each of m machines draws n independent and identically distributed (i.i.d.) samples from some underlying distribution $\mathcal{D}$ and that, for simplicity, $\mathbb{E}_{x \sim \mathcal{D}}[x] = 0$. Then we want to to estimate the leading eigenvector of the covariance matrix $\Sigma := \mathbb{E}_{x \sim \mathcal{D}}[xx^{\mathsf{T}}]$. Here, solutions are all subject to a natural *sign ambiguity* (in addition to a scale ambiguity, which is trivial to handle by normalizing); if $\hat{v}_1^i$ is the estimate produced by machine i, then $-\hat{v}_1^i$ is also a valid local solution. Thus, naively averaging two solutions from two machines could result in an estimate close to zero. Even though the resulting estimate may still be aligned with, e.g., v_1, its magnitude will decrease at the same rate as the magnitude of the noise, and thus averaging offers no improvement. Continuing this, suppose we fix some unit-norm eigenvector v_1 and assume that roughly half of the local solutions are aligned with v_1 while the other half are aligned with $-v_1$; it is clear that we should not expect naive averaging to work in this situation. Indeed, Garber, Shamir, and Srebro [22] showed that the resulting estimate will have $\Omega(\sqrt{1/n})$ error. On the other hand, a centralized estimator with access to all $m \cdot n$ samples would achieve an error of $\mathcal{O}(\sqrt{1/mn})$. To address this, they developed a "sign-fixing" scheme for combining eigenvectors to achieve an error rate similar to the centralized estimator.

When estimating eigenspaces of higher dimension (e.g., the first r principal components for $r > 1$), we have to deal with an *orthogonal ambiguity* (i.e., for any solution V and orthogonal matrix Z, VZ is also a solution), making the task of combining local solutions highly nontrivial for problems exhibiting natural symmetries. In particular, adapting spectral algorithms to the distributed setting poses a significant challenge, and these algorithms form the basis of a rich set of applications such as dimensionality reduction [28], clustering [54], ranking [44], and high-dimensional estimation [30]. Figure 1.1 depicts the results of distributed PCA on examples from the `MNIST` dataset; indeed, naive averaging of local solutions can be catastrophic in practice. On the other hand, the solution produced by an appropriate alignment algorithm (in particular, using our Algorithm 2.1) is very close to that of the central algorithm.

1.1. Contribution and outline.

In this paper, we propose a general technique for averaging local solutions to subspace estimation problems in the distributed setting. In particular, we assume that every machine i observes a noisy version $\hat{X}^i$ of a symmetric "ground truth" X, and we wish to estimate the principal r-dimensional eigenspace of X. Our method relies on aligning local estimates with a reference solution (which can be any of the local solutions), followed by an averaging step. In the special case $r = 1$, our method recovers the *sign-fixing* scheme of [22]. When the underlying matrix X has a nontrivial eigengap and the local matrices $\hat{X}^i$ are not "too far" from X, we show that the error of the resulting estimate is a combination of a term that scales *quadratically* with local errors $E^i := \hat{X}^i - X$ and a term that measures how well the empirical average of the local matrices approximates X. Our main result is a *deterministic* bound that does not require any application-specific information.

We specialize these results to distributed PCA and show that, with high probability, the



(a) Samples from digit 4 (b) Samples from digit 8

Figure 1.1. *Projection of samples to the first two principal components of the* `MNIST` *dataset, in a distributed setting where samples are split across $m = 25$ machines. The central algorithm (Central) produces a very different scatterplot than naive averaging (Average) of local solutions. In contrast, our alignment algorithm (Aligned) leads to a projection very similar to that of the central algorithm. The subspace distance of the averaged solution is ≈ 0.95 (indicating that the subspaces are near-orthogonal to each other), while the subspace distance of the aligned solution is ≈ 0.35.*

estimate produced by our algorithm matches the error rate of a centralized estimator, with access to all $m \cdot n$ samples. In addition, we show that the per-node sample complexity scales with the *stable rank* $r_\star := \mathsf{intdim}(\Sigma)$ of the population covariance matrix Σ, which is typically much smaller than both the ambient dimension d and the algebraic rank r, for sub-Gaussian designs. Our rates improve by a factor of $\sqrt{r}$ upon those of [18] for sub-Gaussian distributions, though the upper bound given in the latter work is for the subspace distance measured in the Frobenius norm and thus not directly comparable with ours (on the other hand, applying norm equivalence to our result to obtain a bound on the Frobenius subspace distance exactly recovers the rate of [18]). We also recover the result of [22] when $r = 1$. Finally, we conduct several numerical experiments that demonstrate the efficacy of our proposed scheme on applications covering (i) PCA, (ii) node embeddings for graph data, and (iii) spectral initializations for quadratic sensing.

1.2. Related work. Distributed computing has received widespread attention in recent years, leading to a number of different methods of aggregating information; these include

1. solving subproblems locally and then forming a central solution by averaging all local solutions,
2. "distributing" an iterative procedure across machines, for example by communicating first- and/or second-order information to the coordinator to perform a "central" gradient step, and
3. using gossip algorithms, in which individual machines are allowed to communicate with peers (instead of only a central coordinator) and can aggregate updates from their neighbors.

In the first of the above settings, the average of local solutions may be computed only once [58] or communicated back to the local machines to repeat multiple rounds of computation [56]. The second and third settings have received widespread attention from the convex optimization community [15, 27, 39, 41, 49, 50], even allowing *optimal* algorithms in both centralized and decentralized settings [46, 52]. From a statistical perspective, other works have shown that distributed algorithms can achieve minimax-optimal rates for statistical estimation [16, 29, 45, 57]. Lately, a recurring theme is also *robustness*, in order to deal with the possibility that some computing nodes may be malicious [12, 20].

Dealing with natural symmetries. As mentioned above, problems with solutions exhibiting natural symmetries require more careful analyses that address the symmetry at hand. For spectral methods, a major portion of the literature has focused on distributed PCA (and the related task of eigenspace estimation of covariance matrices). For example, recent work formulates PCA in a streaming setting and adapts techniques such as gradient descent and variance reduction [2,3,47]. However, the resulting methods are not communication-efficient—at least not without nontrivial modifications—as they need to access sequences of samples that may be scattered across machines. Other works [9, 22] have adapted the shift-and-invert framework [21] (which reduces an eigenproblem to approximately solving a sequence of linear systems) to the distributed setting, though the resulting algorithms still require multiple communication rounds.

More attractive options in terms of communication cost were proposed in [19, 31, 37] for PCA and related problems. In these algorithms, each node performs a local SVD and broadcasts its top $r_1 \geq r$ singular values and vectors (Σ^i, V^i), which act as a summary of the local data, to a central node. That node then forms the matrix $Y := [\Sigma^1(V^1)^\mathsf{T} \quad \dots \quad \Sigma^m(V^m)^\mathsf{T}]$ and computes its SVD, returning the top $r_2 = r$ right singular vectors, where r is the dimension of the desired subspace. This procedure can also be augmented by sketching to reduce the communication cost, and adapted to other settings such as kernel PCA [4]. An alternative approach is the Frequent Directions method [23], which can incrementally update a sketch of a matrix that serves as a low-rank approximation; in particular, the sketches produced by the method are *mergeable*, making the method amenable to parallelization. Since the aforementioned works are primarily interested in getting a high quality low-rank approximation of the data matrices, approximating the leading invariant subspace of the population covariance matrix in a distributed fashion was largely overlooked until recently. One of the first works in this direction is [17], which focuses on leading eigenvector estimation for large matrices by aggregating the eigenvector of randomly sampled submatrices using an alignment step that removes sign ambiguity. Removing sign ambiguity from singular vectors is also discussed in [7]; therein, the authors utilize dataset information to determine meaningful signs for the computed singular vectors. However, in settings where the data are random, the algorithm from [7] can still result in arbitrarily chosen signs; moreover, it attempts to fix signs of individual singular vectors rather than arbitrary orthogonal ambiguities; this makes it unsuitable as a preprocessing step for an averaging method when $r > 1$.

A similar approach to [17] is adopted in [22], which proposes an optimal averaging method for computing the leading eigenvector by aligning all local estimates with a reference solution (e.g., the estimate of the first node); in addition to showing that the algorithm essentially matches the performance of "centralized" PCA, the authors in [22] also show that "naive" averaging can produce arbitrarily bad estimates. However, they do not generalize their technique to more than one eigenvector. More recent works [6, 18] show that an aggregation method similar to that of [37] achieves error competitive with a centralized estimator, even for $r > 1$, via careful statistical analyses. The resulting algorithm deals with orthogonal ambiguity by averaging the local *spectral projectors*; however, this leaves open the question of how to generalize the methods of [17, 22] to higher-dimensional subspaces.

1.3. Notation. We denote $\{1, \ldots, n\}$ by $[n]$ and write $\langle x, y\rangle = x^{\mathsf{T}}y$ for vectors x and y, as well as $\langle X, Y\rangle = \operatorname{Tr}(X^{\mathsf{T}}Y)$ for compatible matrices X and Y. Given $A \in \mathbb{R}^{d_1 \times d_2}$, we write $A_{i,:} \in \mathbb{R}^{d_1}$ for its ith row vector and $A_{:,j} \in \mathbb{R}^{d_2}$ for its jth column vector. We write $\|A\|_F$ and $\|A\|_2$ for the Frobenius and spectral norms of A and $\|A\|_{2\to\infty} := \max_{i\in[d_1]} \|A_{i,:}\|_2$. We let $\mathbb{S}^{d-1} := \{z \in \mathbb{R}^d \mid \|z\|_2 = 1\}$ denote the unit sphere in d dimensions, and we let $\mathbb{O}_{d_1,d_2}$ denote the set of $d_1 \times d_2$ matrices with orthonormal columns (i.e., the set of $d_1 \times d_2$ orthogonal matrices), omitting the second subscript when $d_1 = d_2$. Throughout, we write $\operatorname{dist}_2(U, V) := \|UU^{\mathsf{T}} - VV^{\mathsf{T}}\|_2$ for the distance between the subspaces spanned by the columns of U and V. We use the letters m to denote the number of machines, n to denote the number of samples drawn per machine, and d to denote the dimension of each sample. Finally, we use $A \lesssim B$ to denote that $A \leq c \cdot B$ for some constant c independent of m, n, and d.

2. Distributed eigenspace estimation.

We study distributed eigenspace estimation when all machines observe a "noisy" version of an underlying symmetric matrix X. In particular, we assume that X admits an eigendecomposition of the form

$$X = \begin{bmatrix} V_1 & V_2 \end{bmatrix} \begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix} \begin{bmatrix} V_1 & V_2 \end{bmatrix}^{\mathsf{T}}, \tag{2.1}$$

where $\Lambda_1 := \operatorname{diag}(\lambda_1, \ldots, \lambda_r)$ contains the r largest eigenvalues and Λ_2 contains the $d - r$ smallest eigenvalues, ordered algebraically so that $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_d$. In our setting, the objective is to estimate the leading r-dimensional invariant subspace V_1 given a set of m machines with noisy observations $\hat{X}^i \in \mathbb{R}^{d\times d}$, $i \in [m]$, which are also symmetric. Though our presentation assumes that we are interested in the leading r-dimensional subspace, this assumption is without loss of generality; our results also apply to, e.g., approximating the r-dimensional invariant subspace corresponding to the *smallest* eigenvalues, since the latter can be turned into the leading eigenspace by an appropriate shift.

To illustrate, consider the setting of distributed PCA. There, every machine i draws n i.i.d. samples $x_j^{(i)} \in \mathbb{R}^d$, $j \in [n]$, from a distribution $\mathcal{D}$ that we assume is zero-mean for the sake of simplicity and forms its "local" empirical covariance matrix

$$\hat{X}^i := \frac{1}{n}\sum_{j=1}^{n} x_j^{(i)} x_j^{(i)\mathsf{T}}. \tag{2.2}$$

Here $\mathbb{E}\left[\hat{X}^i\right] = \Sigma := \mathbb{E}_{x\sim\mathcal{D}}\left[xx^{\mathsf{T}}\right]$, where $\mathcal{D}$ is the underlying distribution, and the task at hand amounts to estimating the leading eigenspace of the population covariance matrix; without loss of generality, *assume that* $\lambda_1(\Sigma) = 1$. Note that by standard concentration arguments [55], a centralized version of this problem achieves an error rate of roughly $\tilde{\mathcal{O}}(\sqrt{1/mn})$ (with distance measured in the spectral norm), so we cannot expect a better rate in the distributed setting.

A natural first approach, inspired by one-shot averaging in convex optimization [58], has each machine compute a local approximation $\hat{V}_1^i$ and then averages all local solutions centrally, with the hope that averaging will further "smooth out" the errors from local solutions:

$$\text{form} \quad \bar{V} := \frac{1}{m}\sum_{i=1}^{m} \hat{V}_1^i \quad \text{and take } Q \text{ factor from} \quad \mathtt{qr}(\bar{V}). \tag{2.3}$$

However, this typically fails due to the inherent orthogonal ambiguity of the problem; in short, there is no guarantee that $\hat{V}_1^i$ will be sufficiently "aligned" with each other for their average to be close to V_1. One must first resolve the orthogonal ambiguity in local solutions to meaningfully aggregate them.

Garber, Shamir, and Srebro [22] solve this problem when $r = 1$ in the setting of distributed PCA, under fairly minimal assumptions. Specifically, they show that averaging a certain combination of the local eigenvector estimates recovers a vector $\bar{v}_1$ that satisfies (with high probability)

$$\mathrm{dist}_2(\bar{v}_1, v_1) = \tilde{\mathcal{O}}\left(\sqrt{\frac{1}{\delta^2 mn}} + \frac{1}{\delta^2 n}\right),$$

where v_1 is the leading eigenvector of X and $\delta := \lambda_1(X) - \lambda_2(X)$. Letting $\hat{v}_1^{(i)}$ denote the estimate of v_1 produced by the ith machine, the trick from Garber, Shamir, and Srebro [22] is to pick a "reference" vector, e.g., $\hat{v}_1^1$, and "align" all other local estimates with it to resolve the aforementioned ambiguity, which reduces to a sign ambiguity as $r = 1$. More specifically, up to normalization, they compute $\bar{v}_1$ by the "sign-fixed" average:

$$\bar{v}_1 := \frac{1}{m}\sum_{i=1}^{m} \mathrm{sign}\left(\left\langle \hat{v}_1^{(i)}, \hat{v}_1^{(1)} \right\rangle\right)\hat{v}_1^{(i)}, \quad \hat{v}_1^{(i)} := \operatorname*{argmax}_{v\in\mathbb{S}^{d-1}} v^{\mathsf{T}}\hat{X}^i v. \tag{2.4}$$

This is the same as (2.3) for dimension $r = 1$, with the additional sign-fixing in front of the estimate $\hat{v}_1^{(i)}$. Conceptually, if we omitted the sign-fixing step, we would be averaging eigenvectors that are "spread" around two directions: $\{\pm v_1\}$. If each local estimate has the same probability of pointing to v_1 and $-v_1$, the average will be no better at estimating v_1 than any local solution, with error at least on the order of $\sqrt{1/n}$ [22, Theorem 3].

2.1. Procrustes fixing. We propose a high-order analogue of the sign-fixing average over local estimates. When $r > 1$, the canonical way to align two matrices $\hat{V}, V \in \mathbb{O}_{n,r}$ is via the solution of the so-called *orthogonal Procrustes problem* [24, Chapter 6.4]:

$$\operatorname*{argmin}_{Z\in\mathbb{O}_r} \left\|\hat{V} - VZ\right\|_F. \tag{2.5}$$

The alignment problem in (2.5) admits a closed form solution [26]: let $P\Sigma Q^{\mathsf{T}}$ be the SVD of $V^{\mathsf{T}}\hat{V}$; then $Z := PQ^{\mathsf{T}}$ is the solution to (2.5). Given the local estimates $\hat{V}_1^{(1)}, \dots, \hat{V}_1^{(m)}$ of the principal eigenspace, we may choose one of them to become the "reference" solution (e.g., $\hat{V}_1^{(1)}$) and return $\tilde{V}$ defined below as our estimator:

$$\tilde{V}, \tilde{R} := \mathtt{qr}(\bar{V}), \quad \bar{V} := \frac{1}{m}\sum_{i=1}^{m} \hat{V}_1^{(i)} Z_i, \quad Z_i := \operatorname*{argmin}_{Z\in\mathbb{O}_r} \left\|\hat{V}_1^{(i)} Z - \hat{V}_1^{(1)}\right\|_F. \tag{2.6}$$

Algorithm 2.1 Distributed eigenspace estimation with Procrustes fixing.

```
Input: local principal subspaces $\left\{\hat{V}_1^{(i)} \mid i \in [m]\right\}$, reference solution $\hat{V}$ (default: $\hat{V}_1^{(1)}$)
for $i = 1, \dots, m$ do
    $\tilde{V}^{(i)} := \hat{V}_1^{(i)} Z_i, \quad Z_i := \operatorname{argmin}_{Z \in \mathbb{O}_r} \left\| \hat{V}_1^{(i)} Z - \hat{V} \right\|_F$
end for
Form $\bar{V} := \frac{1}{m} \sum_{i=1}^m \tilde{V}^{(i)}$
return $\tilde{V}$ from $\tilde{V}, \tilde{R} = \mathtt{qr}(\bar{V})$.
```

In particular, note that (2.6) recovers the sign-fixed average of (2.4) when $r = 1$, since it follows that $\operatorname{argmin}_{s \in \{\pm 1\}} \|\hat{v}_1^{(i)} - s\hat{v}_1^1\|_2 = \operatorname{sign}(\langle \hat{v}_1^{(i)}, \hat{v}_1^{(1)} \rangle)$. Our algorithm is more formally described in Algorithm 2.1. By default, Algorithm 2.1 uses the first local solution as a reference for (2.6); however, since the order is arbitrary, our results are valid for any local solution used as a reference. In the experimental section, we demonstrate that iteratively refining the reference solution $\hat{V}$ can further reduce the empirical error of the resulting estimator.

Remark 2.1 (Runtime comparison with [18]). The algorithm proposed in [18] for the distributed PCA problem works by spectral projector averaging. After local solutions have been transmitted to the central node, the algorithm therein requires computing the top r eigenvectors of $\frac{1}{m} \sum_{i=1}^m \hat{V}_1^{(i)} (\hat{V}_1^{(i)})^\mathsf{T}$. Even forming this average would require roughly $\mathcal{O}(md^2 r)$ operations, which can be prohibitive; therefore, one would instead opt for a standard iterative algorithm such as orthogonal iteration. Each iteration would go through the following steps:

1. Compute $X \mapsto \hat{V}_1^{(i)} (\hat{V}_1^{(i)})^\mathsf{T} X$ for $X \in \mathbb{R}^{d \times r}$ and all $i \in [m]$. This step would require $\mathcal{O}(mdr^2)$ operations, yielding a set of matrices of size $d \times r$ as intermediate results.
2. Average the m intermediate results for a total of $\mathcal{O}(mdr)$ operations.
3. Orthogonalize the iterate (possibly every few iterations instead of at every single iteration), incurring a cost of $\mathcal{O}(dr^2)$ operations (e.g., by using the QR factorization).

Therefore, the cost of a *single step* of the iterative algorithm of choice would be $\mathcal{O}(mr^2 d)$ operations in the central node when using the method of [18].

In contrast, and *assuming no further parallelization*, our algorithm only needs to solve $m-1$ Procrustes problems, which are equivalent to $m-1$ SVDs of the matrices $(\hat{V}_1^{(1)})^\mathsf{T} \hat{V}_1^{(i)} \in \mathbb{R}^{r \times r}$. Forming each of these matrices takes time $\mathcal{O}(r^2 d)$, and computing the SVD would require time $\mathcal{O}(r^3)$ (using, e.g., the Golub–Kahan bidiagonalization method). The total cost therefore scales as $\mathcal{O}(mr^2 d)$.

Remark 2.2 (Further parallelization). Note that Algorithm 2.1 can be further parallelized at the expense of only a couple of additional communication rounds. In particular, the central node can first broadcast the reference solution $\hat{V}_1^{(1)}$ to the rest of the $m-1$ nodes, each of which solves the Procrustes problem locally and only transmits back the "aligned" solution $\tilde{V}^{(i)}$. The observed cost given this additional parallelization step is equal to $\mathcal{O}(T_{\mathsf{comm}} + r^2 d)$, where T_{comm} is the time required for a single round of communication.

Ideally, we would like to show that the performance of Algorithm 2.1 matches that of the centralized version. A naive attempt to adapt the proof of [22] poses a set of challenges: on one hand, the case $r = 1$ relies on a Taylor-like expansion of each local estimate, which is not transferable to $r > 1$ in the absence of stronger assumptions on the spectrum of the matrices involved. On the other hand, some of the arguments in [22] that deal with the ambiguity of the local estimates are no longer applicable because of the presence of an arbitrary orthogonal transform, instead of just a sign ambiguity.

Instead, we use recent results from numerical analysis and matrix perturbation theory. First, we show that if the local estimates $\hat{V}_1^{(i)}$ were *already aligned with* V_1, in the sense that

$$\min_{Z \in \mathbb{O}_r} \left\| V_1 Z - \hat{V}_1^{(i)} \right\|_F = \left\| V_1 - \hat{V}_1^{(i)} \right\|_F, \tag{2.7}$$

then averaging yields an estimate whose error is the sum of a term whose magnitude is *quadratic* in the local errors $\left\{ \left\| \hat{X}^i - X \right\|_2 \mid i \in [m] \right\}$[1] plus another term that depends on the error of the empirical mean approximation to X, i.e., $m^{-1} \sum_i \left(\hat{X}^i - X \right)$. To show this, we express $\hat{V}_1^{(i)}$ using V_1 as a local basis—a method that has been used to analyze invariant subspace perturbations [14, 32]. Indeed, in the setting of distributed PCA, this yields an estimator whose error matches that of the centralized estimator.

Recall that if the columns of V_1 form a basis for the leading invariant subspace of X, so do those of $V_1 U$, where $U \in \mathbb{O}_r$ is any orthogonal matrix. Thus we have a degree of freedom in choosing an appropriate "version" of V_1 to work with; without loss of generality, we choose V_1 so that it minimizes the Procrustes distance to the first local solution $\hat{V}_1^{(1)}$; in other words,

$$\operatorname*{argmin}_{U \in \mathbb{O}_r} \left\| \hat{V}_1^{(1)} U - V_1 \right\|_F = I_r. \tag{2.8}$$

Once the matrix V_1 is fixed, Algorithm 2.1 aligns all other local estimates with a solution "sufficiently close" to V_1. Indeed, Stewart [51] showed that aligning $\hat{V}_1^{(i)}$ with $\hat{V}_1^{(1)}$ is the same as aligning $\hat{V}_1^{(i)}$ with V_1, up to a *quadratic* error term.

Before we state our generic result, we formalize our assumptions below.

Assumption 2.3. *There is a collection of matrices $\left\{ \hat{X}^i \mid i \in [m] \right\}$ as well as a reference matrix X (with leading invariant subspace V_1) satisfying the following:*
- *$\lambda_r(X) - \lambda_{r+1}(X) \geq \delta$ for some $\delta > 0$.*
- *The error matrices $E^i := \hat{X}^i - X$ satisfy $\|E^i\|_2 < \frac{\delta}{8}$ for all $i \in [m]$.*

In particular, we will see later that Assumption 2.3 is satisfied with high probability in the setting of distributed PCA with i.i.d. samples. We can now state our deterministic result.

Theorem 2.4. *Let $X \in \mathbb{R}^{d \times d}$ have spectral decomposition (2.1), and let $\hat{X}^i$, $i \in [m]$, denote the local samples of X with leading invariant subspaces $\hat{V}_1^{(i)} \in \mathbb{O}_{d,r}$. Let Assumption 2.3 hold; then, if $\tilde{V}$ is the output of Algorithm 2.1, it satisfies the following error bound:*

$$\operatorname{dist}_2(\tilde{V}, V_1) \lesssim \frac{1}{\delta^2} \max_{i \in [m]} \left\| \hat{X}^i - X \right\|_2^2 + \frac{1}{\delta} \left\| \frac{1}{m} \sum_{i=1}^{m} \hat{X}^i - X \right\|_2. \tag{2.9}$$

[1]These errors are typically $o(1)$.

Remark 2.5. The error expression in (2.9) naturally decomposes into two terms. The second term is the result of the Davis–Kahan theorem applied to quantify the distance of the top eigenspace of the empirical average $\frac{1}{m}\sum_i \hat{X}^i$ from the true eigenspace and is precisely the error of a centralized algorithm that approximates V_1 by the top eigenspace of the empirical average of the local matrices. The first term in (2.9) is an order of magnitude smaller than the error of approximating the leading eigenspace using just the local solution $\hat{V}_1^{(i)}$, as long as the individual errors are $o(1)$. In this case, the contribution of the first term in (2.9) is negligible, and the total error is comparable to that of a centralized estimator.

In the next section, we give a more detailed outline of the proof of Theorem 2.4 that highlights the individual technical components and how they fit together. The proofs of the individual components are deferred to the supplementary material.

2.2. Proof outline. The proof of Theorem 2.4 first analyzes the performance of an idealized (but fictitious) version of Algorithm 2.1 that uses an "ideal" reference solution. In particular, we first "fix" a matrix V_1 with orthogonal columns spanning the leading eigenspace of X and assume (without loss of generality, due to symmetry) that V_1 is already aligned with $\hat{V}_1^{(1)}$, in the sense that it satisfies (2.8). Then, we introduce a collection of fictitious iterates $\widehat{V}_1^{(i)}$—not to be confused with the (unaligned) local solutions $\hat{V}_1^{(i)}$—which represent precisely what the alignment algorithm would output if V_1 were used as a reference solution. Then, we use the path-independence result of [51] to relate the Procrustes fixing estimator to this idealized version.

For brevity, we use the following notation: if $V = [V_1 \quad V_2]$ is the matrix of eigenvectors of X, with V_1 containing the principal eigenvectors, we write for an arbitrary matrix Z

$$Z_{ij} := V_i^{\mathsf{T}} Z V_j, \quad i, j \in \{1, 2\}\,. \tag{2.10}$$

The first ingredient in our proof is a local expansion lemma motivated by the arguments in [14].

Lemma 2.6. *Let Assumption 2.3 hold, and choose $\widehat{V}_1^{(i)}$ to be the matrix whose columns are a basis of the leading invariant subspace of $\hat{X}^i$ and furthermore which is maximally aligned with V_1, in the sense that*

$$\operatorname*{argmin}_{U \in \mathbb{O}_r} \left\| \widehat{V}_1^{(i)} U - V_1 \right\|_F = I_r \quad \forall\, i \in [m]. \tag{2.11}$$

Let $\widehat{Z}^{(i)}$ be the root of the equation[2]

$$0 = -\hat{X}_{21}^i + \widehat{Z}^{(i)} \hat{X}_{11}^i - \hat{X}_{22}^i \widehat{Z}^{(i)} + \widehat{Z}^{(i)} \hat{X}_{12}^i \widehat{Z}^{(i)}, \tag{2.12}$$

and define $\widehat{Y}^{(i)} := V_2 \widehat{Z}^{(i)}$ for $i \in [m]$. Then the following holds:

$$\left\| \widehat{V}_1^{(i)} - V_1 - \widehat{Y}^{(i)} \right\|_2 \lesssim \frac{\left\| \hat{X}^i - X \right\|_2^2}{\delta^2} \quad \forall\, i \in [m]. \tag{2.13}$$

[2]See [14, section 3.2] for motivation of this quantity.

Proof. See section SM2.1. ■

The expansion in (2.13) is nearly sufficient for our purposes; in particular, if we knew that $V_2 Z^{(i)}$ has small enough spectral norm, summing over i and applying the triangle inequality would be sufficient. Even though this is not the case here, we can still show that the average $m^{-1}\sum_i V_2 Z^{(i)}$ depends on the approximation error of the empirical average $m^{-1}\sum_i \hat{X}^i - X$, as well as the squares of the local approximation errors $\hat{X}^i - X$, which can be studied and bounded on a per-application basis.

Lemma 2.7. *In the same setting as Lemma 2.6, the set of matrices $\widehat{Y}^{(i)}$ for $i \in [m]$ satisfies*

$$\left\| \frac{1}{m}\sum_{i=1}^m \widehat{Y}^{(i)} \right\|_2 \lesssim \frac{1}{\delta^2 m}\sum_{i=1}^m \|\hat{X}^i - X\|_2^2 + \frac{1}{\delta}\left\| \frac{1}{m}\sum_{i=1}^m \hat{X}^i - X \right\|_2 . \tag{2.14}$$

Proof. See subsection SM2.2. ■

With Lemmas 2.6 and 2.7 at hand, we can finally give an error bound for the average of the fictitious estimates $\widehat{V}_1^{(i)}$.

Proposition 2.8. *Let Assumption 2.3 hold and $\widehat{V}_1^{(i)}$ be defined as in Lemma 2.6. Then the following holds:*

$$\left\| \frac{1}{m}\sum_{i=1}^m \widehat{V}_1^{(i)} - V_1 \right\|_2 \lesssim \frac{1}{\delta^2 m}\sum_{i=1}^m \|\hat{X}^i - X\|_2^2 + \frac{1}{\delta}\left\| \frac{1}{m}\sum_{i=1}^m \hat{X}^i - X \right\|_2 . \tag{2.15}$$

Proof. Applying the triangle inequality and the above results, we obtain

$$\left\| \frac{1}{m}\sum_{i=1}^m \widehat{V}_1^{(i)} - V_1 \right\|_2 = \left\| \frac{1}{m}\sum_{i=1}^m \widehat{V}_1^{(i)} - V_1 - \widehat{Y}^{(i)} + \widehat{Y}^{(i)} \right\|_2 \tag{2.16}$$

$$\leq \frac{1}{m}\left\| \sum_{i=1}^m \widehat{V}_1^{(i)} - V_1 - \widehat{Y}^{(i)} \right\|_2 + \left\| \frac{1}{m}\sum_{i=1}^m \widehat{Y}^{(i)} \right\|_2 \tag{2.17}$$

$$\overset{(\text{Lemma 2.6})}{\lesssim} \frac{1}{\delta^2 m}\sum_{i=1}^m \|\hat{X}^i - X\|_2^2 + \left\| \frac{1}{m}\sum_{i=1}^m \widehat{Y}^{(i)} \right\|_2 \tag{2.18}$$

$$\overset{(\text{Lemma 2.7})}{\lesssim} \frac{1}{\delta^2 m}\sum_{i=1}^m \|\hat{X}^i - X\|_2^2 + \frac{1}{\delta}\left\| \frac{1}{m}\sum_{i=1}^m \hat{X}^i - X \right\|_2 . \tag{2.19}$$

■

This result holds for the fictitious estimates $\widehat{V}_1^{(i)}$; however, it is not clear that the Procrustes-aligned estimates, $\tilde{V}^{(i)}$, of Algorithm 2.1 also satisfy such a property. To show this, we leverage a recent result by Stewart [51] below; informally, the result says that aligning with an "accurate enough" reference $\hat{V}_1^{(1)}$ is equivalent to directly aligning with V_1, up to quadratic error in $\hat{X}^i - X$.

Lemma 2.9. *Let $\widehat{V}_1^{(i)}$ be defined as in Proposition 2.8 and $\tilde{V}^{(i)}$ as in Algorithm 2.1. Then*

$$\tilde{V}^{(i)} = \widehat{V}_1^{(i)} + T^{(i)}, \quad \|T^{(i)}\|_2 \lesssim \max\left\{ \|\hat{X}^1 - X\|_2^2, \|\hat{X}^i - \hat{X}^1\|_2^2, \|\hat{X}^i - X\|_2^2 \right\}. \tag{2.20}$$

Proof. See subsection SM2.3. ∎

With Lemma 2.9 at hand, we are now ready to show that the Procrustes fixing estimates produce a good approximation to the leading invariant subspace V_1. Our results here do not depend on a particular application setting such as distributed PCA; we are only using matrix computations and Assumption 2.3.

Theorem 2.10 (Procrustes fixing). *Let Assumption 2.3 hold. The estimates $\tilde{V}^{(i)}$ from Algorithm 2.1 satisfy*

$$\left\| \frac{1}{m}\sum_{i=1}^{m} \tilde{V}^{(i)} - V_1 \right\|_2 \lesssim \frac{1}{\delta^2} \max_{i\in[m]} \left\|\hat{X}^i - X\right\|_2^2 + \frac{1}{\delta}\left\| \frac{1}{m}\sum_{i=1}^{m} \hat{X}^i - X \right\|_2. \tag{2.21}$$

Consequently, the output of Algorithm 2.1 satisfies

$$\operatorname{dist}_2(\tilde{V}, V_1) \lesssim \frac{1}{\delta^2} \max_{i\in[m]} \left\|\hat{X}^i - X\right\|_2^2 + \frac{1}{\delta}\left\| \frac{1}{m}\sum_{i=1}^{m} \hat{X}^i - X \right\|_2. \tag{2.22}$$

Proof. Under Assumption 2.3, applying Lemma 2.9 lets us rewrite $\tilde{V}^{(i)} = \widehat{V}^{(i)} + T^{(i)}$, where $T^{(i)}$ satisfies

$$\|T^{(i)}\|_2 \lesssim \max\left\{ \left\|\hat{X}^i - X\right\|_2^2, \left\|\hat{X}^i - \hat{X}^1\right\|_2^2, \left\|\hat{X}^1 - X\right\|_2^2 \right\} \quad \forall i \in [m], \tag{2.23}$$

where we can further upper bound (via Young's inequality $(a+b)^2 \le 2(a^2+b^2)$)

$$\left\|\hat{X}^i - \hat{X}^1\right\|_2^2 = \left\|\hat{X}^i - X + X - \hat{X}^1\right\|_2^2 \lesssim \max\left\{ \left\|\hat{X}^i - X\right\|_2^2, \left\|\hat{X}^1 - X\right\|_2^2 \right\} \tag{2.24}$$

$$\Rightarrow \|T^{(i)}\|_2 \lesssim \max_{i\in[m]} \left\|\hat{X}^i - X\right\|_2^2. \tag{2.25}$$

By rewriting the desired statement and applying the triangle inequality, we have

$$\begin{aligned}
\left\| \frac{1}{m}\sum_{i=1}^{m} \tilde{V}^{(i)} - V_1 \right\|_2 &\le \left\| \frac{1}{m}\sum_{i=1}^{m} \widehat{V}_1^{(i)} - V_1 \right\|_2 + \frac{1}{m}\sum_{i=1}^{m} \|T^{(i)}\|_2 \\
&\overset{\text{(Prop. 2.8)}}{\lesssim} \frac{1}{\delta^2 m}\sum_{i=1}^{m} \left\|\hat{X}^i - X\right\|_2^2 + \frac{1}{\delta}\left\| \frac{1}{m}\sum_{i=1}^{m} \hat{X}^i - X \right\|_2 + \frac{1}{m}\sum_{i=1}^{m} \|T^{(i)}\|_2 \\
&\overset{(2.25)}{\le} \frac{1}{\delta^2 m}\sum_{i=1}^{m} \left\|\hat{X}^i - X\right\|_2^2 + \frac{1}{\delta}\left\| \frac{1}{m}\sum_{i=1}^{m} \hat{X}^i - X \right\|_2 + \max_{i\in[m]} \left\|\hat{X}^i - X\right\|_2^2 \\
&\lesssim \frac{1}{\delta^2} \max_{i\in[m]} \left\|\hat{X}^i - X\right\|_2^2 + \frac{1}{\delta}\left\| \frac{1}{m}\sum_{i=1}^{m} \hat{X}^i - X \right\|_2,
\end{aligned}$$

which completes the proof. ∎

2.3. Consequences for distributed PCA.

With Theorem 2.10 at hand, we can show that applying Algorithm 2.1 to the distributed PCA problem can yield estimates whose error rate matches that of a centralized estimator. The first setting we consider is similar to that of [22], summarized below.

Assumption 2.11. *Consider a zero-mean distribution $\mathcal{D}$ supported on $\mathbb{R}^d$, with covariance matrix $X = \mathbb{E}_{x\sim\mathcal{D}}\left[xx^{\mathsf{T}}\right]$. We assume the following:*

1. *Each of the m available machines draws n i.i.d. samples from $\mathcal{D}$, denoted $x_j^{(i)}$ for $j \in [n]$, $i \in [m]$.*
2. *The population covariance matrix X satisfies an eigengap condition:*

$$\lambda_r(X) - \lambda_{r+1}(X) \geq \delta > 0, \tag{2.26}$$

   where δ is some fixed scalar and r is the rank of the target subspace.
3. *Any $x \sim \mathcal{D}$ satisfies $\|x\|_2 \leq \sqrt{b}$ almost surely.*

Note that part 3 of Assumption 2.11 is just for simplicity and is adopted in [22] as well. Later, we will show that covariance matrices with small intrinsic dimension enable improved statistical rates.

We first argue that Assumption 2.3 is satisfied with high probability in this setting. To do so, we first define the events

$$\mathcal{E} := \mathcal{E}_1 \cap \mathcal{E}_2, \tag{2.27}$$

$$\mathcal{E}_1 := \left\{ \left\| \frac{1}{m}\sum_{i=1}^{m} \hat{X}^i - X \right\|_2 \leq 2 \cdot \sqrt{\frac{b^2 \log(2d/p)}{mn}} \right\}, \tag{2.28}$$

$$\mathcal{E}_2 := \left\{ \max_{i\in[m]} \left\|\hat{X}^i - X\right\|_2 \leq \min\left\{ \frac{\delta}{8}, 2 \cdot \sqrt{\frac{b^2 \log(2dm/p)}{n}} \right\} \right\}, \tag{2.29}$$

where p is a parameter controlling the probability of failure and $\hat{X}^i := \frac{1}{n}\sum_{j=1}^{n} x_j^{(i)}(x_j^{(i)})^{\mathsf{T}}$ denotes the local empirical covariance matrices with leading r-dimensional invariant subspaces $\hat{V}_1^{(i)}$. By Lemma SM1.3, we have that

$$\mathbb{P}\left(\mathcal{E}\right) \geq 1 - 2p - 2dm \exp\left\{ -\frac{n\delta^2}{4b^2} \right\}. \tag{2.30}$$

Then we obtain the following theorem.

Theorem 2.12. *Under Assumption 2.11, the estimate $\tilde{V}$ returned by Algorithm 2.1 satisfies*

$$\mathrm{dist}_2(\tilde{V}, V_1) \lesssim \sqrt{\frac{b^2 \log(2d/p)}{\delta^2 mn}} + \frac{b^2 \log(2dm/p)}{\delta^2 n}, \tag{2.31}$$

with probability at least $1 - 2p - 2dm\exp\left\{-\frac{n\delta^2}{4b^2}\right\}$, whenever $n \gtrsim \log\frac{dm}{p}$.

Theorem 2.12 shows that the error rate of distributed PCA with Algorithm 2.1 decays roughly as $\sqrt{b^2\log(cd/p)/\delta^2 mn}$, under the assumption that $\|x_i\|_2 \leq \sqrt{b}$ almost surely; however, typical modeling choices, including the case where $x_i \sim \mathcal{N}(0, \Sigma)$, which is explored in the numerical experiments of subsection 3.1, do not guarantee such boundedness. Although rates for covariance estimation of sub-Gaussian vectors are on the order of $\sqrt{d/mn}$ [55, Chapter 6.3], when the covariance matrix exhibits rapid spectral decay the statistical error depends on its *intrinsic dimension*, defined for a PSD matrix A as

$$\mathsf{intdim}(A) := \frac{\mathrm{Tr}\,(A)}{\|A\|_2} = 1 + \sum_{i \geq 2} \frac{\lambda_i(A)}{\lambda_1(A)}. \tag{2.32}$$

Notice that in general $1 \leq \mathsf{intdim}(A) \leq \mathrm{rank}(A)$, and that it is possible to have $\mathsf{intdim}(A) \ll \mathrm{rank}(A)$. Using the intrinsic dimension, we can refine our rates as follows.

Theorem 2.13. *Modify Assumption 2.11 so that Item 3 (boundedness) is no longer required, and assume $\mathcal{D}$ is a zero-mean sub-Gaussian multivariate distribution with covariance matrix $X := \mathbb{E}_{x \sim \mathcal{D}}[xx^{\mathsf{T}}]$. Let $r_\star := \mathsf{intdim}(X)$. Then, as long as $n \gtrsim \frac{r_\star + \log(m/p)}{\delta^2}$, the output of Algorithm 2.1 satisfies*

$$\mathrm{dist}_2(\tilde{V}, V_1) \lesssim \frac{r_\star + \log(m/p)}{n} \cdot \left(\frac{\|X\|_2}{\delta}\right)^2 + \sqrt{\frac{r_\star + \log(c_1 n)}{mn}} \cdot \frac{\|X\|_2}{\delta} \tag{2.33}$$

with probability at least $1 - p - 2n^{-c_1}$, where c_1 is a dimension-independent constant.

Proof. The proof uses the matrix deviation inequality [53] to obtain tighter concentration for the empirical covariance matrices but is otherwise identical to the proof of Theorem 2.4. See subsection SM2.4 for details. ■

The rate from Theorem 2.13 is similar to that of [18, Theorem 4], absent an additional factor of $\sqrt{r}$ appearing in the latter. However, the two bounds are not directly comparable; the error bound in [18] is stated in terms of the Frobenius distance of the spectral projectors $\mathrm{dist}_F(V, \hat{V}) := \left\|VV^{\mathsf{T}} - \hat{V}\hat{V}^{\mathsf{T}}\right\|_F$. It may also be possible to adapt the analysis of [18] to remove the extra $\sqrt{r}$ factor when the subspace distance is measured in the spectral norm. Table 2.1 gives a summary of statistical error rates in distributed PCA under different settings.

Table 2.1
Comparison of rates for distributed PCA. Here, $\kappa := \frac{\|\Sigma\|_2}{\delta}$, where δ is the population eigengap, r is the dimension of the principal subspace, and $r_\star := \mathsf{intdim}(\Sigma)$.

Setting	Rate	Reference
$\mathcal{D} \subset \sqrt{b}\mathbf{B}^d$	$\tilde{\mathcal{O}}\left(\sqrt{\frac{b^2}{\delta^2 mn}} + \frac{b^2}{\delta^2 n}\right)$	[22] $(r = 1)$ Theorem 2.12 (general)
$\mathcal{D}$ sub-Gaussian	$\mathcal{O}\left(\kappa\sqrt{\frac{r_\star + \log n}{mn}} + \kappa^2 \cdot \frac{r_\star + \log(m)}{n}\right)$ $\mathcal{O}\left(\sqrt{r}\kappa\sqrt{\frac{r_\star}{mn}} + \sqrt{r}\kappa^2 \frac{r_\star}{n}\right)$	Theorem 2.13 [18]†

† Error bound given in terms of $\mathrm{dist}_F(V_1, \hat{V}) := \left\|V_1V_1^{\mathsf{T}} - \hat{V}\hat{V}^{\mathsf{T}}\right\|_F$.

3. Numerical experiments.

In this section, we present numerical experiments on real and synthetic data. The code to reproduce them, written in `Julia`, is available online.[3] In all experiments below, we use the label "Central" to refer to PCA using the empirical covariance matrix of all $m \cdot n$ samples. In the synthetic experiments, we use zero-mean multivariate Gaussians with covariance matrix

$$\Sigma := UTU^{\mathsf{T}}, \; U \sim \mathrm{Unif}(\mathbb{O}_d), \tag{3.1}$$

where $T := \mathrm{diag}(\{\tau_i\}_{i \in [d]})$ is generated according to one of the following two models (recall $r_\star := \mathsf{intdim}(\Sigma)$):

$$\text{(M1)} \qquad \tau_i := \begin{cases} \lambda_h - \frac{(\lambda_h - \lambda_\ell)\cdot(i-1)}{r-1}, & i \le r, \\ (\lambda_\ell - \delta)\cdot 0.9^{i-r-1}, & i > r, \end{cases} \quad \text{or}$$

$$\text{(M2)} \qquad \tau_i := \begin{cases} 1, & i \le r, \\ (1-\delta)\cdot \alpha^{i-r}, & i > r, \end{cases} \quad \text{where } \alpha \text{ solves } \quad \frac{(1-\delta)}{1-\alpha} = r_\star - r.$$

In model (M1), the r principal eigenvalues are linearly spaced in $[\lambda_\ell, \lambda_h]$. In contrast, in (M2) all principal eigenvalues are 1, while the trailing eigenvalues decay according to a specified rate $\alpha < 1$. Both constructions ensure that the eigengap is exactly equal to δ.

3.1. Performance as a function of m and n.

We generate a set of synthetic experiments following the model (M1), setting $\lambda_\ell = 0.5$, $\lambda_h = 1$, and $\delta = 0.2$. We then apply Algorithm 2.1 for a variety of (m, n) after setting $d = 300$, which is the parameter used in [22] (wherein the case $r = 1$ is studied). Figure 3.1 depicts the performance of Algorithm 2.1 for a number of different configurations. For $r \in \{1, 4, 8, 16\}$, Algorithm 2.1 performs essentially as well as a centralized PCA, which uses all $m \cdot n$ samples to form the empirical covariance matrix. For $r = 1$, our error plots closely resemble those from [22]. In all configurations, "naive" averaging produces an estimate $\bar{V}$ with error on the order of $\Omega(1)$ that does not always decay as a function of n; for that reason, we omit its depiction.

Crucially, our theory requires n to be sufficiently large, since otherwise certain conditions may not hold with high probability. For example, if n is too small, it is unlikely that $\left\|\hat{X}^i - X\right\|_2 \le \frac{\delta}{8}$, as required. For that reason, we generate another synthetic experiment following model (M1). This time, we keep the total number of samples $m \cdot n$ *fixed* and vary m. The results are shown in Figure 3.2. Unsurprisingly, larger values of m lead to less accurate local solutions and smaller overall accuracy of the Procrustes-aligned solution. Note that even if most local solutions are accurate enough, larger values of m (or equivalently smaller values of n) imply there is a higher probability that the reference solution $\hat{V}_1^{(1)}$ is a poor approximation of V_1, which inevitably leads to loss of accuracy during the alignment step.

3.2. Iterative refinement.

The estimates generated by Algorithm 2.1 were empirically observed to be sensitive to the individual solution $\hat{V}_1^{(1)}$ chosen as a reference, especially when the ratio $\sqrt{1/n}$ is nonnegligible. To obtain a more robust estimate, we propose a practical iterative refinement step outlined in Algorithm 3.1. Instead of performing a single round of

[3]gitlab.com/vchariso/distributed-eigenspace-estimation

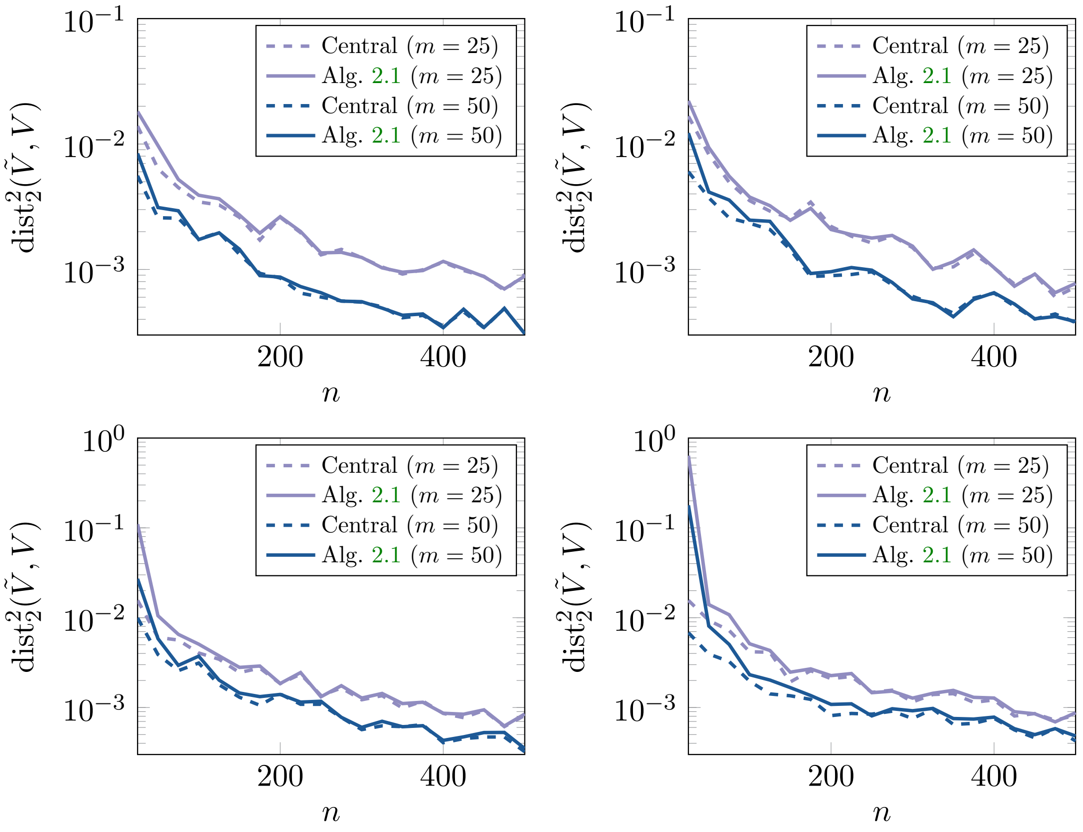


Figure 3.1. *Performance of centralized versus distributed PCA with Procrustes fixing for* $m \in \{25, 50\}$ *and* $n \in \{25, 50, \dots, 500\}$*, with* $\delta = 0.2$*. Target ranks (from left to right):* $r \in \{1, 4\}$ *(top),* $r \in \{8, 16\}$ *(bottom).*

Algorithm 3.1 Procrustes fixing with iterative refinement.

```
Input: local principal subspaces $\left\{\hat{V}_1^{(i)} \mid i \in [m]\right\}$, number of refinement steps n_iter
Set $\tilde{V}^{(0)} := \hat{V}_1^{(1)}$
for $k = 1, \dots,$ n_iter do
    compute $\tilde{V}^{(k)}$ using Algorithm 2.1 with reference solution $\tilde{V}^{(k-1)}$
end for
return $\tilde{V}^{(\texttt{n\_iter})}$.
```

Procrustes alignment, the algorithm goes through multiple iterations where the output of the previous alignment round is used as a reference solution for the current round. Intuitively, even if the initial reference solution was somewhat inaccurate, the averaging step will likely smooth out some of the error, leading to a more accurate reference solution.

To verify this, we perform two experiments. Initially, we compare the performance of Algorithms 2.1 and 3.1 (with `n_iter` $= 2$) in the experiment of subsection 3.1 where $m \cdot n$ is kept fixed and m varies. Empirically, we observe that the additional averaging steps make the most difference for small n, where individual reference solutions are more likely to

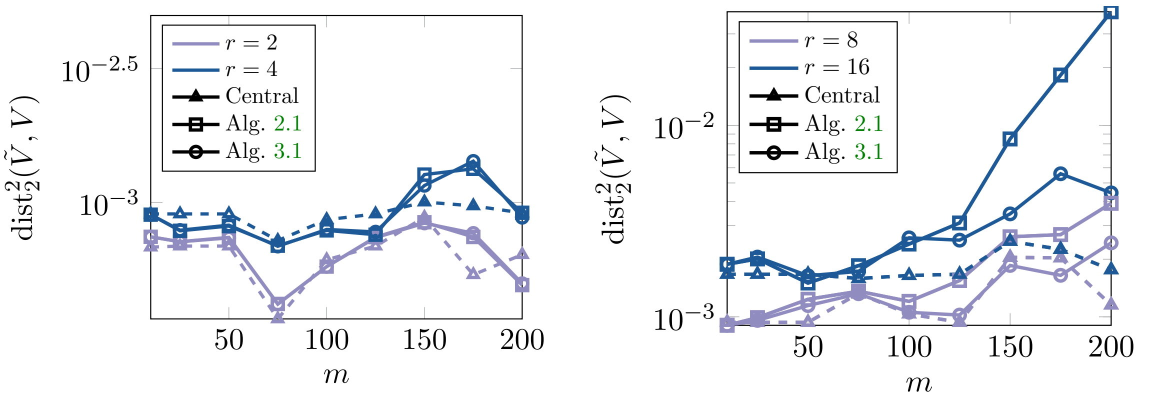


Figure 3.2. *Performance of centralized versus distributed PCA with Procrustes fixing for fixed $n \cdot m = 20000$ with $\delta = 0.2$ and varying m; here, Algorithm 3.1 is invoked with `n_iter` $= 2$. When the number of machines is too large, the accuracy of each individual solution degrades, leading to loss of accuracy in the averaged solution itself.*

be inaccurate. Additionally, we compare Algorithms 2.1 and 3.1 using synthetic instances generated by model (M2) to examine the effect of additional rounds of iterative refinement on the accuracy of solutions; the results can be found in Figure 3.3. Therein, we report the subspace distance from the ground truth by varying `n_iter` $\in \{2, 5, 15\}$. For small n, iterative refinement can decrease the subspace distance significantly, at the cost of a few extra rounds of computation. Moreover, just a few rounds of refinement are sufficient; the difference between 5 and 15 refinement steps is negligible in all configurations of $(n, r_\star)$.

3.3. Performance as a function of intrinsic dimension.

To examine the influence of $r_\star = \mathsf{intdim}(X)$ on the performance of the algorithm, we generate a family of covariance matrices following the model (M2) by letting

$$r_\star \in \left\{ r + 2^k \;\middle|\; k = 2, \ldots, 6 \right\},$$

where r is the dimension of the principal subspace, while keeping $d = 250$, $n = 2 \cdot d$, and $m = 100$ fixed. Figure 3.4 depicts the results for Gaussian samples generated according to (M2) and $r \in \{2, 5, 10\}$. In all cases, the performance of Algorithm 2.1 is competitive with centralized PCA, as well as with [18, Algorithm 1], its error being at worst a constant multiplicative factor larger than centralized PCA. In addition, the performance of Algorithm 3.1, the iteratively refined variant of Algorithm 2.1 introduced in subsection 3.2, is on par with both centralized PCA as well as [18, Algorithm 1]. In all cases, an increase in the intrinsic dimension $r_\star$ implies an increase in the error of all estimators.

Moreover, we set up an additional experiment where the intrinsic dimension $r_\star$ was kept fixed while varying the target subspace dimension r, for the same choice of parameters $d = 250$, $n = 2 \cdot d$, $m = 100$ and eigengap equal to $\delta = 0.25$, using Gaussian samples. In particular, we tried 3 sets of instances where $r_\star \in \{16, 24, 32\}$ and r is varied between 1 and 10, and compared Algorithms 2.1 and 3.1 with the centralized estimator as well as [18, Algorithm 1]. The experimental findings are depicted in Figure 3.5; we observe that the error follows an

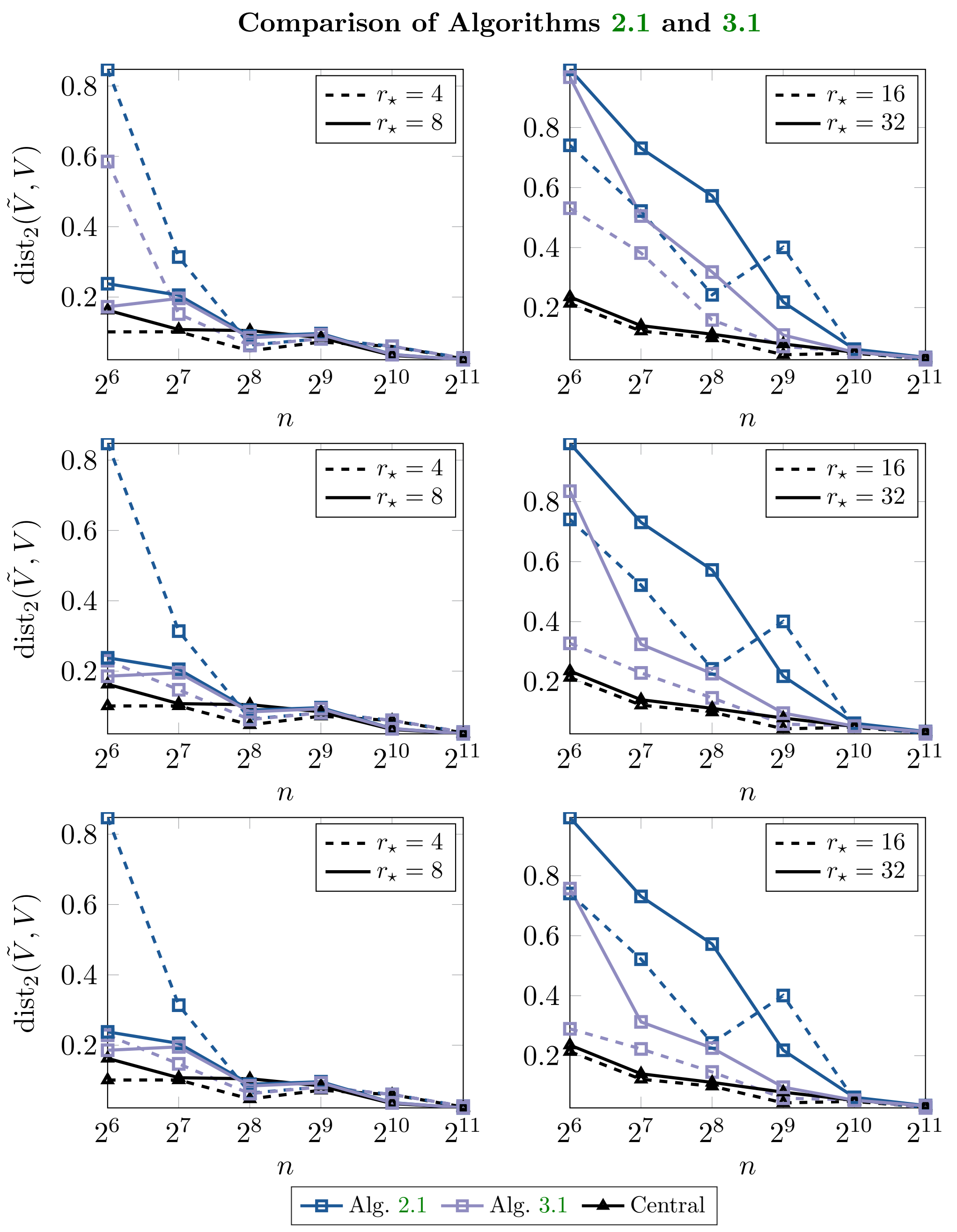


Figure 3.3. *Empirical error of Algorithm 2.1 versus Algorithm 3.1 for a problem with $d = 300$, $m = 50$, and $\delta = 0.1$ for a variety of n and $r_\star := \mathit{intdim}(X)$. $\mathtt{n\_iter} = 2$ (top), $\mathtt{n\_iter} = 5$ (middle), and $\mathtt{n\_iter} = 15$ (bottom). The problem instances generated across rows are identical. In the challenging regime where n is small, iterative refinement can significantly reduce the estimation error. A smaller number of samples per machine n always leads to more significant gaps between central and distributed solutions.*

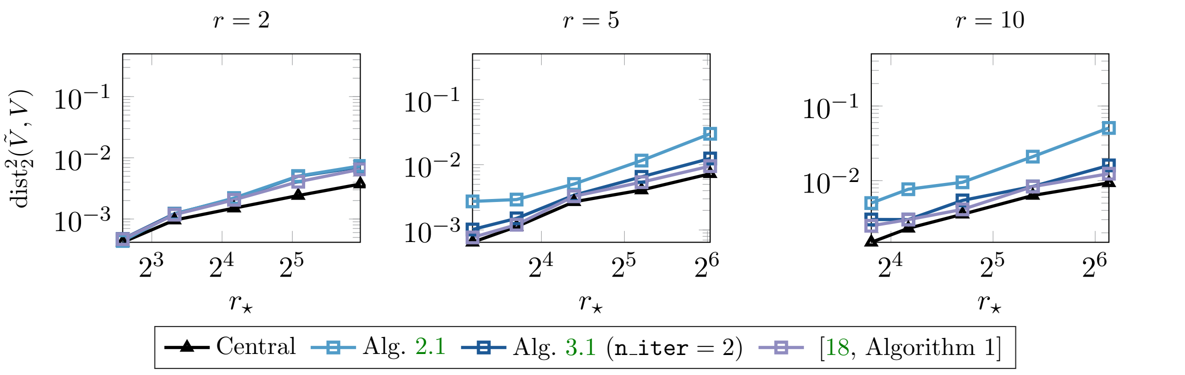


Figure 3.4. *Performance of Algorithms* 2.1 *and* 3.1 *compared to centralized PCA and* [18, Algorithm 1] *for $d = 250$, $n = 500$, $m = 100$, given a covariance matrix X of varying intrinsic dimension $r_\star = \mathsf{intdim}(X)$. Here, $\delta = 0.25$ and $r \in \{2, 5, 10\}$ (left to right). In all cases, the errors of Algorithms* 2.1 *and* 3.1 *are at worst within a constant factor of the error of the centralized algorithm.*

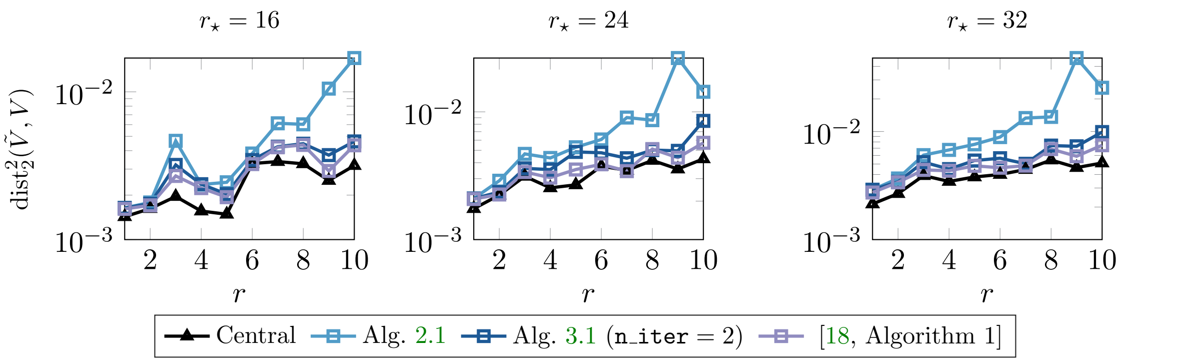


Figure 3.5. *Performance of Algorithms* 2.1 *and* 3.1 *compared to centralized PCA and* [18, Algorithm 1] *for $d = 250$, $n = 500$, $m = 100$ for varying ranks r over instances of fixed intrinsic dimension. Here, $\delta = 0.25$ and $r_\star \in \{16, 24, 32\}$ (left to right). In all cases, the errors of Algorithms* 2.1 *and* 3.1 *are at worst within a constant factor of the error of the centralized algorithm.*

increasing trend as r varies. However, this does not contradict our theoretical results for the following reasons:

- Despite the increasing trend, there are cases where an increase in r does not lead to an increase in error, in contrast to Figure 3.4.
- The centralized estimator, whose theoretical performance depends on $r_\star$ (see, e.g., [53, Theorem 9.2.4]), follows the same trend. Indeed, the dependence on $r_\star$ characterizes the worst-case error of the estimator as quantified by concentration bounds, while the per-instance dependence on r and $r_\star$ may be more nuanced.

3.4. Experiments with non-Gaussian measurements. So far, we have noticed that Algorithm 2.1 achieves comparable yet lower accuracy than Algorithm 1 in [18]; Algorithm 3.1 closes this gap but still does not achieve strictly better estimation error. An interpretation of this is due to the fact that the Procrustes-based algorithm always introduces some bias to the resulting estimator, whereas the algorithm of [18] is unbiased for Gaussian data (which has

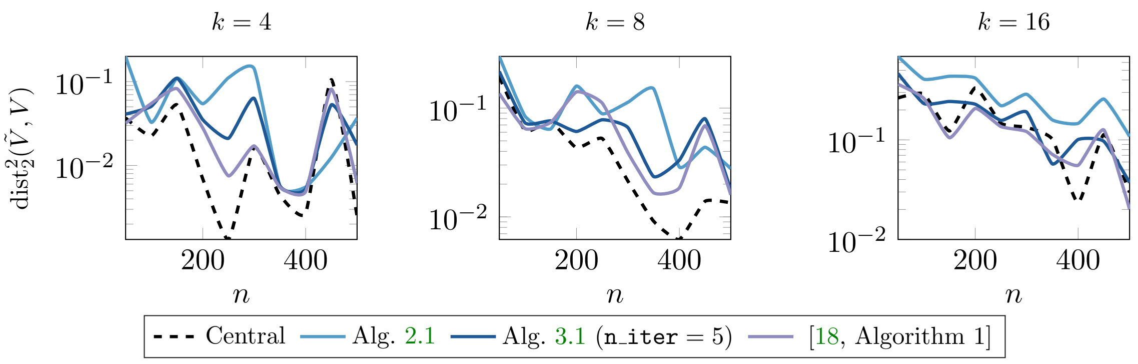


Figure 3.6. *Performance of Algorithms 2.1 and 3.1 compared to centralized PCA and* [18, Algorithm 1] *for samples drawn from $\mathcal{D}_k$ in (3.2) for varying $k \in \{4, 8, 16\}$ (left to right). In all cases, we compute the leading eigenspace of dimension $r = \frac{k}{2}$. Note that Algorithm 1 from* [18] *achieves the lowest error in most, but not all, instances.*

been the focus of our synthetic experiments so far).

Here, we show that this phenomenon persists empirically even when moving to non-Gaussian data. In particular, we generate our samples from the following distribution:

$$\mathcal{D}_k = \mathrm{Unif}\{y_1, \dots, y_k\}, \quad \text{where} \quad y_i \in \sqrt{d}\mathbb{S}^{d-1}. \tag{3.2}$$

The distribution $\mathcal{D}_k$ is not Gaussian and, by the remarks in [53, section 5.6], is heavy-tailed unless the number of vectors k grows exponentially in d. To avoid having to deal with centering issues, we estimate the leading eigenspace of the second moment matrix (instead of the covariance matrix) of $\mathcal{D}_k$. Figure 3.6 depicts the results for an experiment with $m = 25$ machines, number of samples $n \in \{50, 100, \dots, 500\}$, and $k \in \{4, 8, 16\}$. Remarkably, we find that Algorithm 1 from [18] results in lower estimation error in most, but not all, instances generated.

3.5. Consistency of theoretical predictions. In this section, we examine how the empirical error compares with the theoretically prescribed rate from Theorem 2.13, which provides a refined bound using the intrinsic dimension. We compare $\mathrm{dist}_2(V, \tilde{V})$ with the following (simplified) bound $f(r_\star, n)$, for various configurations of $r_\star, n$ and fixing $(d, m) = (300, 100)$, $\delta = 0.2$:

$$f(r_\star, n) := \frac{r_\star + \log m}{\delta^2 n} + \sqrt{\frac{r_\star + 2\log(n)}{\delta^2 mn}}. \tag{3.3}$$

Notice that, due to our construction of the eigenvalues of the covariance matrix shown in (M1), higher values for $\dim(V_1)$ increase $\mathsf{intdim}(\Sigma)$. In the figures below, we report the median over 10 independent runs of the algorithm.

Figure 3.7 reveals that the empirical error rate of Algorithm 2.1 is well below the theoretically prescribed rate, as $f(r_\star, n)$ is an order of magnitude loose compared to the subspace distance $\mathrm{dist}_2(\tilde{V}, V_1)$. This is not too surprising, since some of the intermediate results used to arrive at Proposition 2.8 depend on worst-case perturbation bounds which may not materialize in practice.

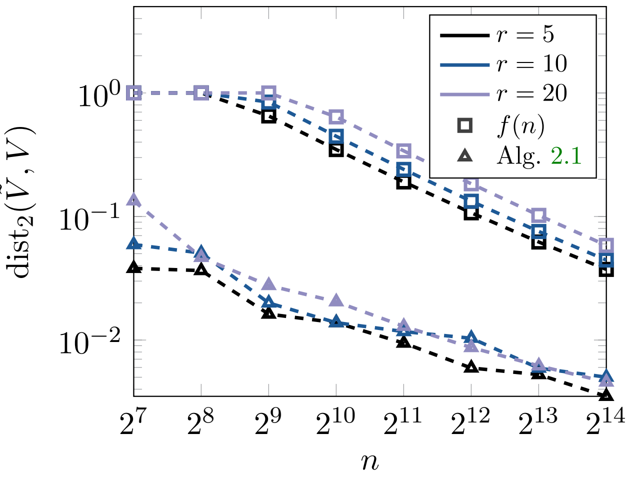


Figure 3.7. *Empirical error of Algorithm 2.1 versus theoretically prescribed error $f(r_\star, n)$ from Theorem 2.13, the latter of which is loose by an order of magnitude. Here, $(d, m) = (300, 100)$ and $\delta = 0.2$.*

3.6. Application: Distributed node embeddings. We apply our algorithm to the task of generating *node embeddings* for undirected graphs. Given a graph $G = (V, E)$, methods for generating node embeddings generate a d-dimensional vector $z_i \in \mathbb{R}^d$ for all nodes $v_i \in V$. One broad category of unsupervised methods for this task is so-called *implicit factorization methods* [25], which attempt to directly minimize a loss of the form

$$\sum_{(v_i,v_j)\in\mathcal{D}} \|\langle z_i, z_j\rangle - s_G(v_i, v_j)\|^2 \approx \left\|ZZ^\mathsf{T} - S_G\right\|_F^2, \tag{3.4}$$

where $\mathcal{D}$ is a set of node pairs (e.g., all connected nodes) and $s_G(v_i, v_j)$ is a proximity measure between nodes v_i, v_j (for example, $s_G(v_i, v_j) = \mathbf{1}\{(v_i, v_j) \in E\}$). Since the loss is invariant to arbitrary orthogonal transforms of the node embeddings, our algorithmic framework is also applicable to combining node embeddings in a distributed environment, in the setting described below.

Experimental setup. Consider a graph $G = (V, E)$, and let m denote the number of machines. We assume that machine i observes a "censored" version $G^{(i)} = (V, E^{(i)})$ of the graph G, where edges are "hidden" independently with probability p, so that $\mathbb{P}(E_{j,k}^{(i)} = 1) = (1 - p) \cdot \mathbf{1}\{E_{j,k} = 1\}$, leading to $\mathbb{E}[E^{(i)}] = (1 - p)E \ \forall i$. Each machine then applies a node embedding algorithm to G^i. In our experiments, we set the edge failure probability $p = 0.1$ and use the `HOPE` method [43] with embedding dimension $d = 64$ and path decay $\beta = 0.1$.

Letting $Z^{(i)} \in \mathbb{R}^{|V|\times d}$ denote the matrix of node embeddings for the ith compute node, we define the following solutions:

- $Z_{\mathrm{avg}} := \frac{1}{m}\sum_{i=1}^m Z^{(i)}Q^{(i)}$ for the Procrustes-aligned estimate, where
$$Q^{(i)} := \operatorname*{argmin}_{U\in\mathbb{O}_d} \left\|Z^{(i)}U - Z^{(1)}\right\|_F.$$
- Z_{cnt} for the "central" estimate, which is the node embedding generated by applying the method on the "uncensored" version of G.

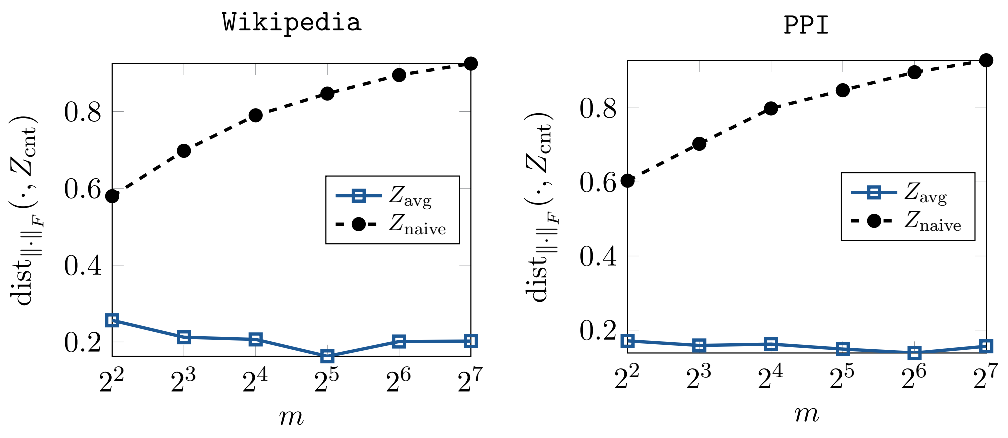


Figure 3.8. *Distance of solutions produced by naive averaging and Algorithm 2.1 from the centralized solution for the* `Wikipedia` *and* PPI *datasets. The quality of the solution constructed by Algorithm 2.1 does not degrade with m, as opposed to naive averaging.*

Table 3.1

*Relative decrease in macro-*F1 *score using Z_{avg} (Algorithm 2.1 averaging on several censored graphs) instead of Z_{cnt} (central solution on entire graph) for one versus rest logistic regression. In most cases, the distributed solution does not reduce the predictive performance (in several instances, using the averaged solution actually leads to a better macro-*F1 *score).*

Dataset	$m =$	2^2	2^3	2^4	2^5	2^6	2^7
`Wikipedia`		0.0	0.0	0.84%	0.0	0.0	0.38%
`PPI`		27.01%	0.0	0.0	0.0	0.0	0.0

- Z_{naive} for the vanilla averaged estimate $\frac{1}{m}\sum_i Z^{(i)}$.

We evaluate the solutions in a node classification setting based on macro-F1 score, using the `Wikipedia` [40] and `PPI` (Protein-Protein Interaction) [42] datasets. The node embeddings serve as features for a logistic regression classifier (after standardization) with ℓ_2 regularization of inverse strength $C = 0.5$ (`Wikipedia`) and $C = 1.0$ (`PPI`). We use a 75%/25% split of training/test data. Finally, we average classification metrics over 10 random instantiations of such splits. Table 3.1 depicts the relative loss in F1 score when using Z_{avg} instead of Z_{cnt}; we see that the relative loss is minimal (with the exception of a single configuration), with several configurations actually benefitting from using the averaged solution.

In addition, we compare the distances of Z_{avg} and Z_{naive} from the "central" embedding Z_{cnt}, as depicted in Figure 3.8. Unsurprisingly, as the number of machines m grows, Z_{naive} strays further away from the embedding generated from a consistent view of G. On the other hand, averaging using Algorithm 2.1 leads to estimates whose distance from the centralized solution does not increase with m.

3.7. Application: Distributed spectral initializations.

We present an application of our algorithm to a distributed method for initializing local search algorithms for *quadratic sensing*. In this setting, one observes N measurements of the form

$$y_i = \left\|X_\sharp^{\mathsf{T}} a_i\right\|_2^2 + \mathtt{noise}_i, \quad i \in [N], \tag{3.5}$$

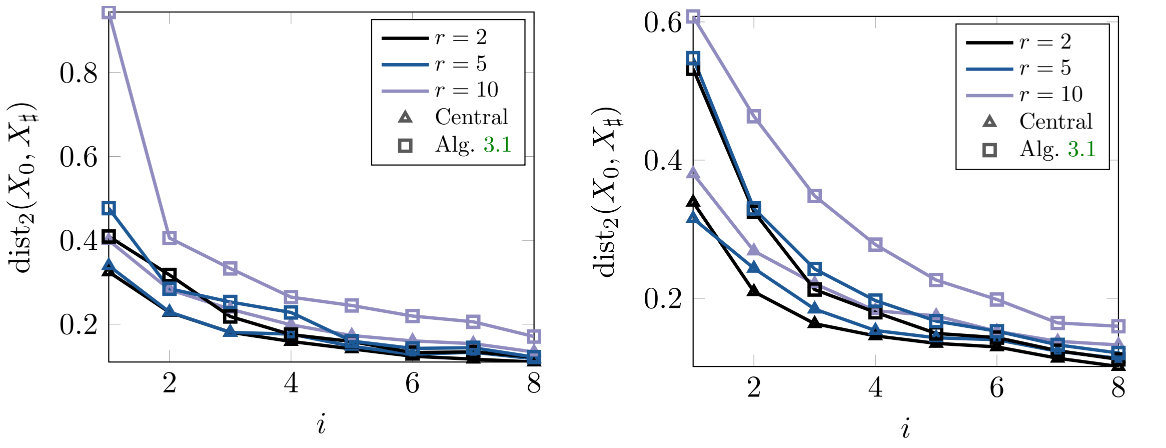


Figure 3.9. *Performance of spectral initialization for quadratic sensing with $X_\sharp \in \mathbb{R}^{d\times r}$ in the centralized and distributed settings. Here, $d = 100$ (left) and $d = 200$ (right), with $n = i \cdot rd$, $m = 30$. Algorithm 3.1 is invoked with `n_iter` $= 10$.*

where $\{a_i\}_{i\in[N]}$ are *known* design vectors, typically satisfying $a_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_d)$, and $X_\sharp \in \mathbb{R}^{d\times r}$ is a matrix to be recovered. Applications of quadratic sensing include covariance sketching [11], learning one-hidden-layer networks with quadratic activations [38], and quantum state tomography [35]. When $r = 1$, quadratic sensing recovers the well-known *phase retrieval* problem, which can be viewed as an instance of the Gaussian *single-index model*, also discussed in the applications of [9]. Even though the natural least-squares formulation of this problem is nonconvex, there are provable algorithms in the literature (see, e.g., [8,10,13] for a nonexhaustive list) that retrieve a global solution (up to rotation) by performing a carefully initialized local search. Assuming for simplicity that $X_\sharp \in \mathbb{O}_{d,r}$, a natural spectral initialization for such methods works as follows: we form the positive semidefinite matrix D_N, given by

$$D_N := \frac{1}{N}\sum_{i=1}^{N} \mathcal{T}(y_i) a_i a_i^{\mathsf{T}}, \quad \mathcal{T}(\cdot) : \mathbb{R} \to \mathbb{R}_+, \tag{3.6}$$

where $\mathcal{T}$ is commonly chosen as a truncation operator, e.g., $\mathcal{T}(y) := y\mathbb{1}\{y \le \tau\}$ for some threshold τ. Standard arguments then show that the leading r-dimensional eigenspace (denoted by X_0) of D_N forms a nontrivial estimate of $X_\sharp$ with high probability, as long as the number of samples $N \gtrsim r^4 d \log(d)$ [8].

Our alignment framework can be applied to the problem at hand when the measurements y_i are collected in several different machines. To initialize, e.g., stochastic local search algorithms, each machine can form its "local" D_N matrix and compute a weak estimate X_0 of $X_\sharp$; a central coordinator can then refine the weak estimate by aggregation and redistribution to local machines. Even though our theory is not directly transferable, we nevertheless demonstrate that Procrustes fixing can be applied for this more general problem.

To that end, we design the following experiment: we set $d \in \{100, 200\}$, $m = 30$ and vary $r \in \{2, 5, 10\}$. We generate a set of instances with $X_\sharp \sim \mathbb{O}_{d,r}$, let n take on values $n \in \{i \cdot r \cdot d, i \in \{1, \dots, 8\}\}$, and apply Algorithm 3.1 for better estimates. In Figure 3.9, we plot the distance $\left\|(I_d - X_\sharp X_\sharp^\mathsf{T})X_0\right\|_2$ as a function of i. The problem becomes more difficult as the rank r increases; nevertheless, the distributed initialization scheme weakly recovers $X_\sharp$ as long as $n \gtrsim 2rd$ on each machine. In contrast, naive averaging of the local solutions produces an estimate that is nearly orthogonal to $X_\sharp$ (we omit its depiction).

4. Discussion.

We presented a communication-efficient algorithm for distributed subspace estimation, which averages the local estimates in a principled manner and achieves comparable performance to a purely centralized estimator using the same amount of data for distributed PCA. Our algorithm generalizes that of [22], which only addresses the rank-1 case, and requires a single round of communication between compute nodes and a central coordinator.

From a theoretical perspective, it is straightforward to adapt our analysis to the setting where a fixed set of vectors is distributed across machines in an i.i.d. fashion, instead of assuming that all vectors are drawn from the same distribution; in that case, the objective is to approximate the centralized *empirical* covariance matrix, which can be viewed as the expectation of each local empirical covariance matrix (this is in fact the setting depicted in Figure 1.1). It may also be possible to leverage the insight and analysis of [14] to prove improved rates for distributed covariance estimation with error measured in the $\ell_{2,\infty}$ norm (which yields entrywise bounds in the rank-1 case). Nonetheless, this will require a more careful analysis of the local expansions of each local solution, as well as $\ell_{2,\infty}$ concentration bounds for empirical covariance matrices which are asymptotically tighter than the corresponding spectral norm bounds.

From a practical perspective, our algorithm is not necessarily confined to distributed covariance estimation. Instead, Procrustes fixing local solutions before averaging can be employed in any estimation problem where local estimates can be rotated arbitrarily, as we demonstrated with distributed node embeddings and spectral initialization.

The setting of this paper also gives rise to a few questions to explore in future work. One example is the following: what if *some of the machines are compromised* and can return an arbitrary matrix with orthonormal columns instead of an unbiased estimate of the leading eigenspace? In this case, we believe it may be possible to adapt the proposed method using a robust distance estimator to choose an appropriate reference solution for Algorithm 2.1 with high probability and treating the averaging step as a robust mean estimation problem.[4] Finally, it would be interesting to identify other classes of learning problems for which our deterministic result can be used in a black-box fashion to yield competitive estimation rates in the distributed setting.

Acknowledgment. We wish to thank the anonymous referees for their careful reading of this work. Their invaluable feedback and suggestions have greatly improved the paper.

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar,

[4]This model of failure is an instance of the so-called *Byzantine generals* problem [36].

P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*, preprint, https://arxiv.org/abs/1603.04467, 2016.

[2] Z. Allen-Zhu and Y. Li, *Doubly accelerated methods for faster CCA and generalized eigendecomposition*, in Proceedings of the 34th International Conference on Machine Learning (Sydney, Australia), D. Precup and Y. W. Teh, eds., Proc. Mach. Learn. Res. 70, PMLR, 2017, pp. 98–106, http://proceedings.mlr.press/v70/allen-zhu17b.html.

[3] Z. Allen-Zhu and Y. Li, *First efficient convergence for streaming k-PCA: A global, gap-free, and near-optimal rate*, in Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS), IEEE, Washington, DC, 2017, pp. 487–492, https://doi.org/10.1109/focs.2017.51.

[4] M. F. Balcan, Y. Liang, L. Song, D. Woodruff, and B. Xie, *Communication efficient distributed kernel principal component analysis*, in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, ACM, New York, 2016, pp. 725–734, https://doi.org/10.1145/2939672.2939796.

[5] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling Up Machine Learning*, Cambridge University Press, Cambridge, UK, 2009, https://doi.org/10.1017/cbo9781139042918.

[6] A. Bhaskara and P. M. Wijewardena, *On distributed averaging for stochastic k-PCA*, in Advances in Neural Information Processing Systems 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds., Curran Associates, Red Hook, NY, 2019, pp. 11026–11035.

[7] R. Bro, E. Acar, and T. G. Kolda, *Resolving the sign ambiguity in the singular value decomposition*, J. Chemometrics, 22 (2008), pp. 135–140, https://doi.org/10.1002/cem.1122.

[8] E. J. Candes, X. Li, and M. Soltanolkotabi, *Phase retrieval via Wirtinger flow: Theory and algorithms*, IEEE Trans. Inform. Theory, 61 (2015), pp. 1985–2007, https://doi.org/10.1109/tit.2015.2399924.

[9] X. Chen, J. D. Lee, H. Li, and Y. Yang, *Distributed estimation for principal component analysis: An enlarged eigenspace analysis*, J. Amer. Statist. Assoc., (2021), https://doi.org/10.1080/01621459.2021.1886937.

[10] Y. Chen and E. Candes, *Solving random quadratic systems of equations is nearly as easy as solving linear systems*, in Advances in Neural Information Processing Systems 28, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds., Curran Associates, Red Hook, NY, 2015, pp. 739–747.

[11] Y. Chen, Y. Chi, and A. Goldsmith, *Exact and stable covariance estimation from quadratic sampling via convex programming*, IEEE Trans. Inform. Theory, 61 (2015), pp. 4034–4059, https://doi.org/10.1109/tit.2015.2429594.

[12] Y. Chen, L. Su, and J. Xu, *Distributed statistical machine learning in adversarial settings*, Proc. ACM Meas. Anal. Comput. Syst., 1 (2017), pp. 1–25, https://doi.org/10.1145/3154503.

[13] Y. Chi, Y. M. Lu, and Y. Chen, *Nonconvex optimization meets low-rank matrix factorization: An overview*, IEEE Trans. Signal Process., 67 (2019), pp. 5239–5269, https://doi.org/10.1109/tsp.2019.2937282.

[14] A. Damle and Y. Sun, *Uniform bounds for invariant subspace perturbations*, SIAM J. Matrix Anal. Appl., 41 (2020), pp. 1208–1236, https://doi.org/10.1137/19M1262760.

[15] J. Duchi, A. Agarwal, and M. Wainwright, *Dual averaging for distributed optimization: Convergence analysis and network scaling*, IEEE Trans. Automat. Control, 57 (2012), pp. 592–606, https://doi.org/10.1109/tac.2011.2161027.

[16] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and Y. Zhang, *Optimality guarantees for distributed statistical estimation*, preprint, https://arxiv.org/abs/1405.0782, 2014.

[17] N. El Karoui and A. d'Aspremont, *Second order accurate distributed eigenvector computation for extremely large matrices*, Electron. J. Statist., 4 (2010), pp. 1345–1385, https://doi.org/10.1214/10-ejs577.

[18] J. Fan, D. Wang, K. Wang, and Z. Zhu, *Distributed estimation of principal eigenspaces*, Ann. Statist., 47 (2019), pp. 3009–3031, https://doi.org/10.1214/18-aos1713.

[19] D. Feldman, M. Schmidt, and C. Sohler, *Turning Big Data into tiny data: Constant-size coresets for k-means, PCA and projective clustering*, in Proceedings of the 2013 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '13, SIAM, Philadelphia, 2013, pp. 1434–1453, https://doi.org/10.1137/1.9781611973105.103.

[20] J. Feng, H. Xu, and S. Mannor, *Distributed Robust Learning*, preprint, https://arxiv.org/abs/1409.5937, 2014.

[21] D. Garber, E. Hazan, C. Jin, S. Kakade, C. Musco, P. Netrapalli, and A. Sidford, *Faster eigenvector computation via shift-and-invert preconditioning*, in Proceedings of the 33rd International Conference on Machine Learning (New York, NY), M. F. Balcan and K. Q. Weinberger, eds., Proc. Mach. Learn. Res. 48, PMLR, 2016, pp. 2626–2634, https://proceedings.mlr.press/v48/garber16.html.

[22] D. Garber, O. Shamir, and N. Srebro, *Communication-efficient algorithms for distributed stochastic principal component analysis*, in Proceedings of the 34th International Conference on Machine Learning (Sydney, Australia), Proc. Mach. Learn. Res. 70, PMLR, 2017, pp. 1203–1212, https://proceedings.mlr.press/v70/garber17a.html.

[23] M. Ghashami, E. Liberty, J. M. Phillips, and D. P. Woodruff, *Frequent directions: Simple and deterministic matrix sketching*, SIAM J. Comput., 45 (2016), pp. 1762–1792, https://doi.org/10.1137/15m1009718.

[24] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins University Press, Baltimore, MD, 2013.

[25] W. L. Hamilton, R. Ying, and J. Leskovec, *Representation Learning on Graphs: Methods and Applications*, preprint, https://arxiv.org/abs/1709.05584, 2017.

[26] N. J. Higham, *The symmetric Procrustes problem*, BIT, 28 (1988), pp. 133–143, https://doi.org/10.1007/bf01934701.

[27] M. Jaggi, V. Smith, M. Takac, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan, *Communication-efficient distributed dual coordinate ascent*, in Advances in Neural Information Processing Systems 27, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds., Curran Associates, Red Hook, NY, 2014, pp. 3068–3076.

[28] I. Jolliffe, *Principal Component Analysis*, 2nd ed., Springer Ser. Statist., Springer, New York, 2002, https://doi.org/10.1007/978-1-4757-1904-8.

[29] M. I. Jordan, J. D. Lee, and Y. Yang, *Communication-efficient distributed statistical inference*, J. Amer. Statist. Assoc., 114 (2018), pp. 668–681, https://doi.org/10.1080/01621459.2018.1429274.

[30] R. Kannan and S. Vempala, *Spectral algorithms*, Found. Trends Theor. Comput. Sci., 4 (2008), pp. 157–288, https://doi.org/10.1561/0400000025.

[31] R. Kannan, S. Vempala, and D. Woodruff, *Principal component analysis and higher correlations for distributed data*, in Proceedings of the 27th Conference on Learning Theory (Barcelona, Spain), M. F. Balcan, V. Feldman, and C. Szepesvári, eds., Proc. Mach. Learn. Res. 35, PMLR, 2014, pp. 1040–1057, http://proceedings.mlr.press/v35/kannan14.html.

[32] M. Karow and D. Kressner, *On a perturbation bound for invariant subspaces of matrices*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 599–618, https://doi.org/10.1137/130912372.

[33] J. Konečný, H. Brendan McMahan, D. Ramage, and P. Richtárik, *Federated Optimization: Distributed Machine Learning for On-Device Intelligence*, preprint, https://arxiv.org/abs/1610.02527, 2016.

[34] J. Konečný, B. McMahan, and D. Ramage, *Federated Optimization: Distributed Optimization Beyond the Datacenter*, preprint, https://arxiv.org/abs/1511.03575, 2015.

[35] R. Kueng, H. Rauhut, and U. Terstiege, *Low rank matrix recovery from rank one measurements*, Appl. Comput. Harmon. Anal., 42 (2017), pp. 88–116, https://doi.org/10.1016/j.acha.2015.07.007.

[36] L. Lamport, R. Shostak, and M. Pease, *The Byzantine generals problem*, in Concurrency: The Works of Leslie Lamport, ACM, New York, 2019, pp. 203–226, https://doi.org/10.1145/3335772.3335936.

[37] Y. Liang, M.-F. F. Balcan, V. Kanchanapally, and D. Woodruff, *Improved distributed principal component analysis*, in Advances in Neural Information Processing Systems 27, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds., Curran Associates, Red Hook, NY, 2014, pp. 3113–3121.

[38] R. Livni, S. Shalev-Shwartz, and O. Shamir, *On the computational efficiency of training neural networks*, in Advances in Neural Information Processing Systems 27, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds., Curran Associates, Red Hook, NY, 2014, pp. 855–863.

[39] C. Ma, V. Smith, M. Jaggi, M. Jordan, P. Richtárik, and M. Takac, *Adding vs. averaging in distributed primal-dual optimization*, in Proceedings of the 32nd International Conference on Ma-

chine Learning (Lille, France), F. Bach and D. Blei, eds., Proc. Mach. Learn. Res. 37, PMLR, 2015, pp. 1973–1982.

[40] M. Mahoney, *Large Text Compression Benchmark*, http://www.mattmahoney.net/dc/textdata, 2011.

[41] A. Nedic and A. Ozdaglar, *Distributed subgradient methods for multi-agent optimization*, IEEE Trans. Automat. Control, 54 (2009), pp. 48–61, https://doi.org/10.1109/tac.2008.2009515.

[42] J. Nixon, M. Tyers, T. Reguly, J. Rust, A. Winter, M. Livstone, B.-J. Breitkreutz, C. Stark, L. Boucher, A. Chatr-Aryamontri, K. Dolinski, and R. Oughtred, *The BioGRID interaction database*, Nature Prec., 36 (2011), pp. D637–D640, https://doi.org/10.1038/npre.2011.5627.1.

[43] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, *Asymmetric transitivity preserving graph embedding*, in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, ACM, New York, 2016, pp. 1105–1114, https://doi.org/10.1145/2939672.2939751.

[44] L. Page, S. Brin, R. Motwani, and T. Winograd, *The PageRank Citation Ranking: Bringing Order to the Web*, Tech. report, Stanford InfoLab, Stanford University, Stanford, CA, 1999.

[45] J. D. Rosenblatt and B. Nadler, *On the optimality of averaging in distributed statistical learning*, Inf. Inference, 5 (2016), pp. 379–404, https://doi.org/10.1093/imaiai/iaw013.

[46] K. Scaman, F. Bach, S. Bubeck, Y. T. Lee, and L. Massoulié, *Optimal convergence rates for convex distributed optimization in networks*, J. Mach. Learn. Res., 20 (2019), pp. 1–31, http://jmlr.org/papers/v20/19-543.html.

[47] O. Shamir, *Fast stochastic algorithms for SVD and PCA: Convergence properties and convexity*, in International Conference on Machine Learning, New York, NY, 2016, pp. 248–256.

[48] O. Shamir and N. Srebro, *Distributed stochastic optimization and learning*, in Proceedings of the 52nd Annual Allerton Conference on Communication, Control, and Computing (Monticello, IL), D. Precup and Y. W. Teh, eds., IEEE, Washington, DC, 2014, pp. 1203–1212, https://doi.org/10.1109/allerton.2014.7028543.

[49] O. Shamir, N. Srebro, and T. Zhang, *Communication-efficient distributed optimization using an approximate Newton-type method*, in Proceedings of the 31st International Conference on Machine Learning (Bejing, China), E. P. Xing and T. Jebara, eds., Proc. Mach. Learn. Res. 32, PMLR, 2014, pp. 1000–1008, http://proceedings.mlr.press/v32/shamir14.html.

[50] V. Smith, S. Forte, C. Ma, M. Takáč, M. I. Jordan, and M. Jaggi, *CoCoA: A general framework for communication-efficient distributed optimization*, J. Mach. Learn. Res., 18 (2018), pp. 1–49, http://jmlr.org/papers/v18/16-512.html.

[51] G. Stewart, *Smooth Local Bases for Perturbed Eigenspaces*, Tech. Report TR-5010, Institute for Advanced Computer Studies, University of Maryland, Baltimore, MD, 2012.

[52] C. A. Uribe, S. Lee, A. Gasnikov, and A. Nedic, *A dual approach for optimal algorithms in distributed optimization over networks*, Optim. Methods Softw., 36 (2020), pp. 171–210, https://doi.org/10.1080/10556788.2020.1750013.

[53] R. Vershynin, *High-Dimensional Probability*, Camb. Ser. Stat. Probab. Math. 47, Cambridge University Press, Cambridge, UK, 2018, https://doi.org/10.1017/9781108231596.

[54] U. von Luxburg, *A tutorial on spectral clustering*, Stat. Comput., 17 (2007), pp. 395–416, https://doi.org/10.1007/s11222-007-9033-z.

[55] M. J. Wainwright, *High-Dimensional Statistics*, Camb. Ser. Stat. Probab. Math. 48, Cambridge University Press, Cambridge, UK, 2019, https://doi.org/10.1017/9781108627771.

[56] B. A. y Arcas, *Decentralized machine learning*, in 2018 IEEE International Conference on Big Data, IEEE, Washington, DC, 2018, https://doi.org/10.1109/bigdata.2018.8622078.

[57] Y. Zhang, J. Duchi, M. I. Jordan, and M. J. Wainwright, *Information-theoretic lower bounds for distributed statistical estimation with communication constraints*, in Advances in Neural Information Processing Systems 2013, NeurIPS, San Diego, CA, 2013, pp. 2328–2336.

[58] Y. Zhang, J. C. Duchi, and M. J. Wainwright, *Communication-efficient algorithms for statistical optimization*, J. Mach. Learn. Res., 14 (2013), pp. 3321–3363, http://jmlr.org/papers/v14/zhang13b.html.

[59] Z. Zhang, C. Chang, H. Lin, Y. Wang, R. Arora, and X. Jin, *Is network the bottleneck of distributed training?*, in Proceedings of the Workshop on Network Meets AI & ML, NetAI '20, ACM, New York, 2020, pp. 8–13, https://doi.org/10.1145/3405671.3405810.