

# A Review of Distributed Algorithms for Principal Component Analysis

*This paper discusses distributed PCA algorithms that are amenable when data are distributively acquired without communicating and accessing the entire data set locally.*

By SISSI XIAOXIAO WU<sup>✉</sup>, Member IEEE, HOI-TO WAI<sup>✉</sup>, Member IEEE, LIN LI, Member IEEE, AND ANNA SCAGLIONE, Fellow IEEE

**ABSTRACT** | Principal component analysis (PCA) is a fundamental primitive of many data analysis, array processing, and machine learning methods. In applications where extremely large arrays of data are involved, particularly in distributed data acquisition systems, distributed PCA algorithms can harness local communications and network connectivity to overcome the need of communicating and accessing the entire array locally. A key feature of distributed PCA algorithm is that they defy the conventional notion that the first step toward computing the principal vectors is to form a sample covariance. This paper is a survey of the methodologies to perform distributed PCA on different data sets, their performance, and of their applications in the context of distributed data acquisition systems.

**KEYWORDS** | Clustering algorithms; data mining; distributed algorithms; principal component analysis; radar signal processing

## I. OVERVIEW

Distributed algorithms have a long history. In recent years, they have gained prominence in light of the end of Moore's law scaling and the seemingly exponential growth in data to analyze, due to the latest incarnation of networked technologies from social mobile media to the Internet of Things. This backdrop has sparked significant advances over the last decades on multiagent signal processing algorithms. In contrast to their centralized counterparts, these algorithms require the participating agents, i.e., nodes in the network, to make use of their local processing power and the ability to communicate with each other by message passing, with the goal of tackling a common optimization problem.

In this context, a pervasive primitive in machine learning and sensor array processing is the computation of the principal components from the covariance matrix of several data streams. The goal of this paper is to survey distributed algorithms in principal component analysis (PCA), which has wide applications in areas of communications, network, data mining, and machine learning [1]–[3]. Generally speaking, PCA is a statistical procedure that converts a set of high-dimensional samples into a set of features that represent the data in a lower dimensional space spanned by the principal components. Since its first appearance in the seminal 1901 paper by Karl Pearson [4], PCA has evolved into various forms that fit different applications. The distributed PCA algorithms we review in this paper leave a residual relative to the original data set that is minimum in the least square sense, i.e., with the minimum Euclidean norm. This is the most common version of PCA employed in signal processing and data science. The solution found by the PCA problem(s) is widely applied in dimensionality reduction and to cluster large amount of

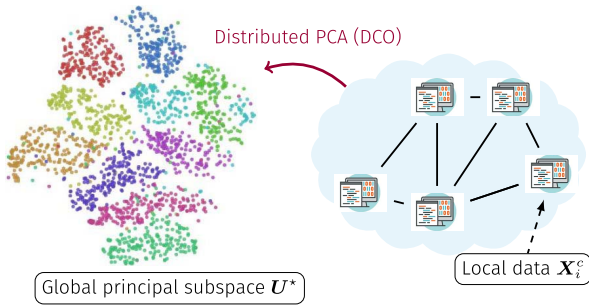
Manuscript received February 24, 2018; revised May 27, 2018; accepted June 4, 2018. Date of current version August 2, 2018. This work was supported in part by the U.S. Air Force under Contracts FA8721-05-C-0002 and/or FA8702-15-D-0001; by the National Natural Science Foundation of China under Grant 61701315; by Shenzhen Technology R&D Fund JCYJ20170817101149906 and JCYJ20170302145906843; by Shenzhen University Launch Fund 2018018; and by the U.S. National Science Foundation under Grants EAGER CCF 1553746, NSF CCF-BSF 1714672, and BSF 2016660. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Government. (Corresponding author: Sissi Xiaoxiao Wu.)

**S. X. Wu** is with the Department of Communication and Information Engineering, Shenzhen University, China (e-mail: xxwu.eesissi@szu.edu.cn).

**H.-T. Wai** and **A. Scaglione** are with the Ira A. Fulton School of Electrical Computer and Energy Engineering, Arizona State University, USA (e-mail: htwai.Scaglione@asu.edu; Anna.Scaglione@asu.edu).

**L. Li** is with the Massachusetts Institute of Technology Lincoln Laboratory, USA (e-mail: lin.li@ll.mit.edu).

Digital Object Identifier 10.1109/JPROC.2018.2846568



**Fig. 1.** Example of distributed PCA with DCO data. In this illustration, each agent is a computer server gathering data locally from the users that it serves, organized through partitions by the columns.

data into groups via spectral clustering [5], to classify word documents [6], and for beam-forming in array processing [1], [2]. We refer the reader to [3] for a comprehensive survey of PCA applications.

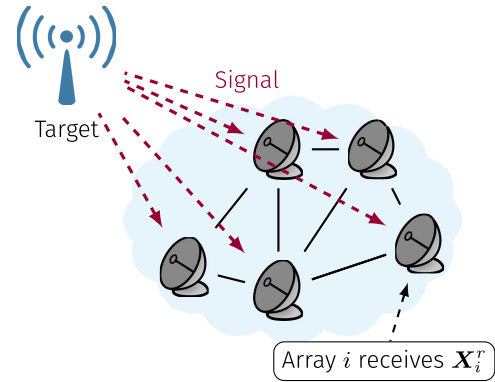
The main goal of this review is to explain how distributed storage and computation systems can be used to implement PCA. Next, we classify these algorithms based on the way the data are partitioned and the communications are structured in the network.

### A. The Family of Distributed PCA Algorithms

The distributed PCA algorithms in the literature fall into different design classes, that respond to: 1) how the data are divided in the network, i.e., the data model in real applications; and 2) how the communication and computations among the different agents (or workers) are structured, namely a hierarchical architecture versus a totally flat architecture based on message passing. We discuss where these different settings arise in the next two subsections, differentiating between the data partition and communication model, respectively.

1) *Data Model*: The design of distributed PCA algorithms depends heavily on the data model; i.e., the way that data are partitioned and stored in the agents on the network in real applications. Below, we survey two of the main classes of data partitions.

In the first class of data partition, each agent has access to a different subset of samples of the data set. This type of partition occurs in applications where a large amount of high-dimensional data is stored across different sites in a network. In this case, distributed PCA allows to learn the important features from these high-dimensional data that help compress, summarize, classify, or rank them, without sharing the data directly. A relevant instance for this setting occurs in document classification (cf., Fig. 1), where the data dimensions are the frequency of a specific word in the document, and each sample corresponds to a different document. As the documents are scattered across various servers, each agent in the network only possesses a subset of the outcomes. Applying distributed PCA in

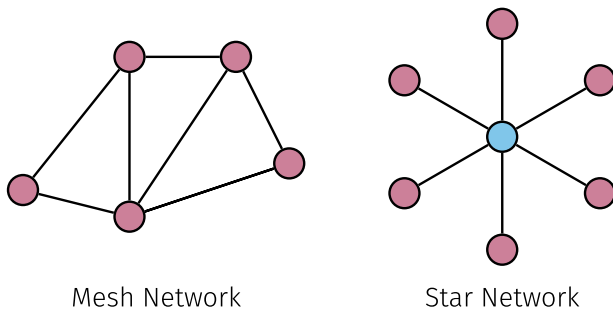


**Fig. 2.** Example of distributed PCA with DRO data. In this example, each array receives its own copy of signal which is partitioned by rows, and all the arrays estimate the subspace of received signals for target tracking.

this case provides a way to denoise the unlabeled data (that can be interpreted as topics) by retaining only the useful subspaces and that can then be used to classify the documents.

Prior work on this class of the distributed PCA algorithms is prevalent in the machine learning community, e.g., [7]–[17]. Specifically, Qu *et al.* [7] and Kargupta *et al.* [14] proposed to compute a set of local principal component vectors at the agents and then have a central coordinator fuse these results into the desired order principal subspace; the performance of this approach (and its variants) has been studied in, e.g., [8]–[11]. In [12], a multiround distributed PCA algorithm was proposed to better balance communication and computation costs. In [13], the principal components are found by performing a sequence of local QR decomposition at the agents. In [15], the principal components are computed using the aggregated eigenvectors and eigenvalues of the local covariance matrix. It is worthwhile to point out that except for [17], the works cited above consider a star network architecture, with a hub that fuses the results of several servers iteratively. This is different from the setting where the servers communicate through a meshed network, which is the primary focus of this paper in Section IV.

The second class of data partition corresponds to having a multidimensional time series and having the entries of each sample distributed across the agents, on an entry-by-entry (or a block-by-block) basis. This situation naturally arises in distributed sensor deployments where the sensors collect samples simultaneously of a continuous field that evolves in time and space. For example, the field could be the signal emitted or backscattered by a moving target (cf., Fig. 2). What motivates the use of PCA is the underlying assumption that the field has few active signal components, and therefore spanning an unknown but low-dimensional signal subspace. Each sensor collects one (or a subset) of the projections of such vector field at a given time, i.e., a spatial sample of the field.



**Fig. 3.** Mesh and star networks. For star networks, the nodes are divided into slave nodes and a master node, where the latter receives data from all the slave nodes.

This class of PCA problems has inspired a number of distributed PCA methods developed by the signal processing community [18]–[29] which we review in Sections IV and V. For instance, Scaglione *et al.* [18], Le Borgne *et al.* [19], and Yildiz *et al.* [20] considered extending the classical power method [30] to the distributed setting. The proposed distributed power method was later analyzed in [21] and extended to an asynchronous setting in [22] and [29]. Note that the power method is a batch processing method with fast convergence, yet the method is nonadaptive and has high latency. When the observations are nonstationary, it is natural to seek adaptive solutions; in the centralized setting, this is achieved by the Oja's method [31], and the distributed setting can be found in [23] and [25], which developed the decentralized subspace tracking algorithms; the asynchronous version has been discussed in [23] and [28]. Interestingly, these classes of algorithms are all for meshed networks.

Before we delve into the mathematical details in Section II, we will fix some terminology by describing how the data samples are arranged. In particular, we define a data matrix which takes each sample of data as a column. In this way, the two main situations described above correspond, respectively, to what we will refer as the distributed columns observations (DCOs) and the distributed rows observations (DROs), respectively. In both cases, the PCA problem aims at retrieving the principal subspace for the column span of the data matrix, yet in the DRO case each node computes only a subset of the coordinates (one entry or one block) of the principal subspace vectors, while in the DCO case all nodes achieve consensus on the entire principal subspace basis. The way the knowledge about the principal subspace is shared in the DCO and DRO cases affects how the distributed PCA primitive can be used for different applications.

2) *Communication and Computation Architecture:* The designs of distributed PCA algorithms also differ in terms of the type of communications they require: one variant uses a hierarchical master–slave-type architecture and one uses a flat message passing architecture (cf., Fig. 3).

The master–slave approach for distributed PCA conforms to a hierarchical division of tasks, and employs a central coordinator that also acts as a fusion node [7]–[16]. In this case, the agents form a star network topology, where the computation tasks of the master node at the center are different from those of the servers. These models conform to the typical architecture for parallel computation in multicore processors, where the aim of applying distributed PCA algorithms is to accelerate PCA computation by utilizing local computations and also local memory resources. It is also worthwhile to point out that the algorithms above fall into the category of the DCO-type distributed PCA.

The algorithms developed for this architecture usually consist of two stages: a local stage and a global stage. In the local stage, each agent/slave performs its own local optimization, e.g., by solving a local PCA problem, and sends the results to the central coordinator. In the global stage, the central coordinator then computes the global PCA from the aggregated data.

The second class of distributed PCA algorithms works on arbitrarily meshed networks and conforms to the type of parallel processing that is performed in graphic processing units (GPUs), or distributed storage systems. They have the following features: 1) all nodes and links must perform the same function and run the same procedures; and 2) the nodes exchange partial computations but not the data, thus privacy is respected. This architecture can be seen as a generalization of the master–slave one, where the agents form a topology described by a general graph, and it arises from applications involving computer or wireless sensor networks. In particular, while the agents are still connected to each other, transmitting information from one agent to another may require multihop communications. In addition to accelerating PCA computations, this class of algorithms aims at offering better resiliencies in hostile environments such as random failures of agents. Examples of these algorithms can be found in [18]–[25]. To adapt to the mesh network setting, these algorithms are often developed by reinterpreting classical numerical methods for PCA such as power and Oja's method as a sequence of computation steps for averaging a set of local values across the agents. The distributed averaging can then be realized by resorting to instantiations of the average consensus (AC) subroutine; see [32] and [33]. Their accuracy rests on the performance of the AC subroutine, which is known to converge exponentially at rate that depends on the algebraic connectivity of the communication network; see Section III-B.

We observe that the majority of distributed PCA algorithms belong to either one of the two types: 1) DCO data on master–slave architectures; and 2) DRO data on arbitrary mesh networks. Such dichotomy is due to the different types of applications that are prevalent in the machine learning and signal processing communities, respectively. Throughout this paper, we focus on surveying algorithms for the settings with arbitrarily meshed networks, though

some of the representatives working for the master–slave architecture will also be summarized in Sections IV-A and V-A. Furthermore, we will emphasize that the tailor-made master–slave distributed PCA algorithms can be more effective in handling specific big data problems, and they are preferable in contexts where one can choose to build such an architecture.

*Notation.* We use boldfaced lower case letters (e.g.,  $\mathbf{x}$ ) to denote vectors and boldfaced upper case letters (e.g.,  $\mathbf{X}$ ) to denote matrices. While this convention is prevalent throughout, occasionally roman capitals  $\mathbf{X}$  will be used to denote the random vectors whose outcomes are denoted with the lower case boldface notation  $\mathbf{x}$  used for deterministic vectors. For a square matrix  $\mathbf{R}$ ,  $\lambda_i(\mathbf{R})$  denotes its  $i$ th largest eigenvalue, while for a rectangular matrix  $\mathbf{X}$ ,  $\sigma_i(\mathbf{X})$  is its  $i$ th largest singular value. The operator  $(\cdot)^H$  [resp.,  $(\cdot)^T$ ] denotes the Hermitian transpose (resp., the standard transpose). Unless otherwise specified,  $\|\mathbf{x}\|$  is the standard Euclidean norm for the vector  $\mathbf{x}$ .

## B. Paper Organization

In the next section, we formulate mathematically the distributed PCA problem and its two instantiations, for the DRO and DCO scenarios, respectively. Then, in Section III, we provide background on the building blocks that are at the basis of the distributed PCA algorithms: specifically, the AC subroutine, the power method, and the Oja’s method. Sections IV and V review the distributed PCA methods for the DRO setting and for the DCO setting, respectively; within each of them, some salient applications are also reviewed. Possible extensions of the work cited and the related problems are presented in Section VI, which are followed by the conclusions in Section VII.

## II. THE PCA PROBLEM AND ALGORITHMS

The original motivation for the PCA problem provided by Karl Pearson in his seminal 1901 paper [4] was to generalize the notion of linear regression

*“to represent a system of points in a plane by the best fitting straight line or plane.”*

Since the notion of the best fitting hyperplane was associated with the least square criterion, the PCA formulation naturally tied into the algebraic notions eigenvalue decomposition (EVD), of singular value decomposition (SVD), spectral decomposition, subspace estimation, orthogonal iteration, as well as orthogonal–triangular (QR) factorization. In statistics, the solution can be interpreted as a minimum mean squared error solution that is based on an empirical estimate of the data covariance. Asymptotically, for a stationary and ergodic process for which the sample covariance converges almost surely to its ensemble, PCA is equivalent to the discrete Karhunen–Loève transform (KLT) [34]–[36], i.e., a linear mapping of random data from a higher dimensional ensemble such that: 1) the projection coefficients are uncorrelated; and 2) relative to the

original high-dimensional representation, the representation through these coefficients in the PC subspace achieves minimum mean square error in expectation. This paper is particularly concerned with those forms in signal processing, communications, and data mining where the quest for this low-dimensional linear approximation applies to a distributed measurement setting, in which multiple data repositories have to agree on what is the appropriate approximating hyperplane, or low-dimensional coordinate system, where the data should be represented.

Mathematically, we consider a set of observations given as an  $N \times T$  (potentially complex) matrix  $\mathbf{X}$ , i.e.,

$$\mathbf{X} := (\mathbf{x}(1) \mathbf{x}(2) \cdots \mathbf{x}(T)) \in \mathbb{C}^{N \times T}. \quad (1)$$

The rows and columns of  $\mathbf{X}$  represent the feature/spatial dimension and the sample/time dimension, respectively. For PCA what matters is the sample correlation of  $\mathbf{X}$ , namely  $\frac{1}{T} \mathbf{X} \mathbf{X}^H$ . Let us define the EVD of  $\frac{1}{T} \mathbf{X} \mathbf{X}^H = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^H$  where  $\mathbf{U}$  is an  $N \times N$  unitary matrix and  $\mathbf{\Lambda} = \text{Diag}([\lambda_1; \dots; \lambda_R; 0; \dots; 0])$  is a diagonal matrix of the eigenvalues, with  $R = \text{rank}(\mathbf{X})$ . Throughout the paper we make the canonical assumption that all the eigenvalues have multiplicity of one, i.e.,  $\lambda_1 > \dots > \lambda_R > 0$ . This assumption can be relaxed, e.g., when one is interested in only finding the top  $p$  principal subspaces, then we have Assumption 1.

**Assumption 1:** The  $p$ th eigenvalue is strictly greater than the  $p + 1$ th one, i.e.,  $\lambda_1 \geq \dots \geq \lambda_p > \lambda_{p+1} \geq \dots \geq \lambda_R > 0$ .

In many practical scenarios, the data samples  $\mathbf{x}(t)$  are correlated with each other. When the data set is large with  $N \gg 1$  and  $T \gg 1$ , an option to reduce the size of the data is to project  $\mathbf{X}$  onto its  $p$ -dimensional ( $p$ -D) principal subspace, which can be represented by an orthogonal transformation  $\mathbf{U}_p$ —a submatrix of the unitary matrix  $\mathbf{U}$  consisting of only its left  $p$  column vectors. In particular, given the orthogonal projection vector  $\mathbf{z}(t) = \mathbf{U}_p^H \mathbf{x}(t) \in \mathbb{C}^p$  the corresponding low-dimensional approximation of  $\mathbf{x}(t)$  is given by  $\hat{\mathbf{x}}(t) = \mathbf{U}_p \mathbf{z}(t)$ . It is also convenient to write the  $t$ th sample as

$$\mathbf{x}(t) = \hat{\mathbf{x}}(t) + \mathbf{e}(t) = \mathbf{U}_p \mathbf{z}(t) + \mathbf{e}(t) \quad (2)$$

where  $\mathbf{e}(t)$  represents the modeling error. PCA is effective when the Frobenious norm of  $\mathbf{e}(t)$  is small. Based on (2), the PCA problem amounts to learning the orthogonal transformation  $\mathbf{U}_p$  from the data  $\mathbf{X}$  that would allow a lossy mapping of  $\mathbf{x}(t) \mapsto \mathbf{z}(t)$ . Under the standard assumption that  $T \geq p$ , the PCA problem can be represented as the following optimization:

$$\begin{aligned} \mathbf{U}^* \in \arg \min_{\mathbf{U} \in \mathbb{C}^{N \times p}} \|\mathbf{I} - \mathbf{U} \mathbf{U}^H\|_{\text{F}}^2 \\ \text{s.t. } \mathbf{U}^H \mathbf{U} = \mathbf{I}. \end{aligned} \quad (3)$$



As seen, the solution of (3) minimizes the residual for the signal reconstructed from  $\{z(t)\}_{t=1}^T$  in the mean square sense. The problem may also be extended to a stochastic and dynamic setting as follows:

$$\begin{aligned} U^*(t) \in \arg \min_{U \in \mathbb{C}^{N \times p}} \mathbb{E}[\|(I - UU^H)X(t)\|^2] \\ \text{s.t. } U^H U = I \end{aligned} \quad (4)$$

where we have denoted  $X(t) \in \mathbb{C}^N$  as a vector-valued random process and the expectation is taken with respect to the distribution of  $X(t)$ . Note that the realization of  $X(t)$  is denoted as  $x(t)$ . To account for the possible nonstationarity in  $X(t)$ , the solution of (4) depends on the sample/time index  $t$ . We will refer to (3) as the batch/static PCA problem, and (4) as the dynamic/stochastic PCA problem. Note that the solution to the PCA problems is intrinsically ambiguous subject to rotations, as we observe by first defining the equivalence class  $[U]$  with  $U \in \mathbb{C}^{N \times p}$

$$[U] := \{\hat{U} \in \mathbb{C}^{N \times p} \mid \hat{U} = UQ, Q \in \mathbb{C}^{p \times p} \text{ is unitary}\}. \quad (5)$$

Importantly, any matrix in  $[U^*]$  (resp.,  $[U^*(t)]$ ) will also be an optimal solution to (3) [resp., (4)].

In the batch PCA setting, a centralized PCA algorithm [37] solves (3) by performing an SVD on  $X$ , where we can simply set the optimal solution as  $U^* = S_p$ . Note that this gives the optimal objective value of  $\sum_{r=p+1}^R \sigma_r^2$ .

### A. Power Method for PCA

Motivated by the fact that an optimal solution to the static PCA problem (3) can be obtained from evaluating the top eigenvectors of the autocorrelation  $XX^H$  [or equivalently the top (left) singular vectors of the data matrix  $X$ ], a natural idea is to apply the power method for solving the PCA problem, which is a well-known numerical method [30] for computing the top eigenvectors of a symmetric matrix. In light of this, here we briefly review the power method and provide insights that will be instrumental to the development of distributed PCA methods. Consider the PCA problem (3) and observe that an optimal solution can be found by retrieving the top- $p$  eigenvectors corresponding to the largest  $p$  eigenvalues of the sampled correlation matrix

$$\hat{R}_x := \frac{1}{T} XX^H = \frac{1}{T} \sum_{t=1}^T x(t)x^H(t). \quad (6)$$

Note that  $x(t)$  is zero mean, then  $\hat{R}_x$  is also the sampled covariance. Alternatively, one could remove the mean in the data by a simple preprocessing step. To compute the top eigenvector of  $\hat{R}_x$ , the power method is initialized by  $u_1[1] \sim \mathcal{CN}(0, I)$ , and it adopts the following recursion:

$$\bar{u}_1[k] = \frac{u_1[k]}{\|u_1[k]\|}, u_1[k+1] = \hat{R}_x \bar{u}_1[k], \forall k \geq 1. \quad (7)$$

It can be shown that  $\bar{u}_1[k]$  converges to  $u_1^*$  as  $k \rightarrow \infty$ , where  $u_1^*$  is the top eigenvector of  $\hat{R}_x$  [30].

From an optimization perspective, the power method can be seen as a fixed-point iteration method which solves the nonlinear system arising from the optimality condition of (3). Its rate of convergence is exponential as it computes an  $\epsilon$ -accurate eigenvector<sup>1</sup> with  $k = \Omega((\log(\sigma_1(\hat{R}_x)/\sigma_2(\hat{R}_x)))^{-1} \log(1/\epsilon))$  power iterations [30]. Note that the ratio  $\sigma_1(\hat{R}_x)/\sigma_2(\hat{R}_x)$  is known as the spectral gap of  $\hat{R}_x$  and it is a key factor determining the convergence speed of the power method. Moreover, as we will reveal later, the computations above can be performed distributively with the help of the average consensus subroutine, by exploiting the relationship between  $\hat{R}_x$  and the data.

To find the second eigenvector of  $\hat{R}_x$ , denoted by  $u_2^*$ , we observe that  $(I - u_1^*(u_1^*)^H)\hat{R}_x$  is also Hermitian and positive semidefinite, and this matrix's top eigenvector is the sought  $u_2^*$ . Naturally, we can apply the same power method procedure to compute  $u_2^*$ . Repeating the same procedures we can find  $u_3^*, \dots, u_p^*$ , to complete the principal subspace  $U^* = (u_1^*, \dots, u_p^*)$ .

### B. Oja-Based Methods for PCA

Originally proposed by Oja *et al.* [31] in 1985, the Oja's method for PCA was developed from a different philosophy than the power method. In particular, the method focuses on tackling the optimization problem (4) using a stochastic gradient descent (SGD) method. Note that (4) is a non-convex and stochastic optimization problem which can be difficult to handle. As a remedy, we consider the special case of  $p = 1$  and the following form of (3):

$$u^*(t) \in \arg \max_{u \in \mathbb{C}^N} f_t(u) := \frac{u^H \mathbb{E}[x(t)x^H(t)]u}{\|u\|^2} \quad (8)$$

where the objective function is also known as the Rayleigh coefficient of the correlation matrix  $R_x(t) := \mathbb{E}[x(t)x^H(t)]$ . The following function is a stochastic approximation of  $f_t(u)$ :

$$\hat{f}(u; \{t\}) := \frac{u^H x(t)x^H(t)u}{\|u\|^2} \approx f_t(u). \quad (9)$$

Since only one sample is used in the above, we say that the batch size used is 1. We observe that the approximation is unbiased as  $\mathbb{E}[\hat{f}(u; \{t\})] = f_t(u)$ . Consequently, its gradient is also unbiased since  $\mathbb{E}[\nabla \hat{f}(u; \{t\})] = \nabla f_t(u)$ .

The Oja's method in [31] is essentially an SGD method for (8) with a batch size of 1, i.e., let  $u^{\text{Oja}}(t) \in \mathbb{C}^N$  be the estimated principal component at iteration  $t$ , then we have

$$\begin{aligned} u^{\text{Oja}}(t+1) &= u^{\text{Oja}}(t) + \tilde{\gamma}_t \nabla \hat{f}(u^{\text{Oja}}(t); \{t\}) \\ &= u^{\text{Oja}}(t) + \gamma_t \left( x(t)x^H(t) - \frac{x^H(t)u^{\text{Oja}}(t)|^2}{\|u^{\text{Oja}}(t)\|^2} I \right) \\ &\quad \times u^{\text{Oja}}(t) \end{aligned} \quad (10)$$

<sup>1</sup>The estimated vector  $\bar{u}_1[k]$  is  $\epsilon$ -accurate if  $\sqrt{1 - |\bar{u}_1[k]^H u_1^*|^2} \leq \epsilon$ .

where  $\tilde{\gamma}_t > 0$  is the SGD step size and  $\gamma_t := 2\tilde{\gamma}_t / \|\mathbf{u}^{\text{Oja}}(t)\|^2$ . It is worth noting that, due to the nonconvex nature of (8), the global convergence of Oja's learning rule has remained elusive. In fact, we focus on the special cases with stationary  $\mathbf{x}(t)$ . For example, Oja *et al.* in [31, Th. 2] proved that if the step sizes are nonsummable diminishing, i.e.,  $\sum_t \tilde{\gamma}_t = \infty$  and  $\sum_t \tilde{\gamma}_t^2 < \infty$ , then (10) converges almost surely to the principal subspace, yet the convergence rate is not given. Balasubramani *et al.* [38] proved that when  $p = 1$ , with diminishing step sizes  $\tilde{\gamma}_t = c/t$ , the system in (10) returns the top eigenvector of  $\mathbf{R}_x$  as  $t \rightarrow \infty$  at a sublinear rate of  $\mathcal{O}(1/t)$ . Nevertheless, the Oja's learning rule is often used even when the process  $\mathbf{x}(t)$  is nonstationary. To avoid getting stuck at a solution equal to the previous subspaces, an effective heuristic is to set  $\tilde{\gamma}_t$  (or  $\gamma_t$ ) to be a small constant such that the newly observed samples are sufficiently represented. Various forms of the Oja-based method have also been proposed. In general, their philosophy is to apply different relaxations to the PCA problem (4) and then to apply SGD on the relaxed problems. Examples are NOja and NOOja studied in [39]. Their convergence has been studied in [40].

### III. SETTINGS FOR DISTRIBUTED PCA

In this section, we first review the distributed data model in real applications and then describe the average consensus algorithm, which will serve as the building block for the distributed PCA methods reviewed in the paper.

#### A. The Types of Data Partitions

In a distributed or dynamic setting, the entries of  $\mathbf{X}$ , instead of being processed at a central machine, are scattered on  $S$  different machines/sensors, which we will refer to as agents later on. We further provide the following definitions to distinguish the types of distributed algorithms.

- DCOs: The DCO setting assumes that each agent observes a subset of columns of  $\mathbf{X}$ . We partition  $\mathbf{X}$  by its columns such that

$$\mathbf{X} = (\mathbf{X}_1^c \mathbf{X}_2^c \cdots \mathbf{X}_S^c) \quad (11)$$

where  $\mathbf{X}_i^c \in \mathbb{C}^{N \times T_i}$  is the column-partitioned submatrix kept by agent  $i$  and  $T = \sum_{i=1}^S T_i$ . Alternatively, the data available at agent  $i$  can also be represented as  $\mathbf{x}(t) \in \mathbb{C}^N$  for  $t \in \mathcal{T}_i$  with  $\mathcal{T}_1 \cup \cdots \cup \mathcal{T}_S = \{1, \dots, T\}$ ,  $\mathcal{T}_i \cap \mathcal{T}_j = \emptyset$ , and  $|\mathcal{T}_i| = T_i$ .

- DROs: The DRO setting assumes that each agent observes only a subset of rows of  $\mathbf{X}$ . We partition  $\mathbf{X}$  by its rows such that

$$\mathbf{X} = ((\mathbf{X}_1^r)^\top (\mathbf{X}_2^r)^\top \cdots (\mathbf{X}_S^r)^\top)^\top \quad (12)$$

where agent  $i$  keeps the row-partitioned submatrix  $\mathbf{X}_i^r \in \mathbb{C}^{N_i \times T}$  with  $N = \sum_{i=1}^S N_i$ . Alternatively, the data available at agent  $i$  can also be represented as  $\mathbf{x}_i(t) \in \mathbb{C}^{N_i}$  for  $t = 1, \dots, T$ .

See Fig. 4 for an illustration on the types of data structure considered. A key feature that distinguishes distributed PCA methods from its centralized counterpart is that the agents have to solve (3) by cooperating with their neighbors in the network, given that each agent has only access to the partial observation matrix. Based on the settings we just discussed, next we describe the goals of the distributed PCA methods.

The DCO setting is encountered primarily in big data mining applications. Each agent in this case can be a computer server that gathers data samples from a set of users that it is serving. For instance, the  $i$ th agent obtains samples  $\{\mathbf{x}(t)\}_{t \in \mathcal{T}_i}$  from a group of users. In this case, our goal is as follows:

Goal (DCO): Agent  $i$  learns a common  $p$ -D principal subspace of  $\mathbf{X}$  shared by the other agents, i.e., to learn  $\mathbf{U}^*$ , in a distributed fashion.

Note that the goal here is similar to solving a consensus problem requiring the agents to agree with each other. Using the data structure in the DCO setting, we observe that (3) can be written as

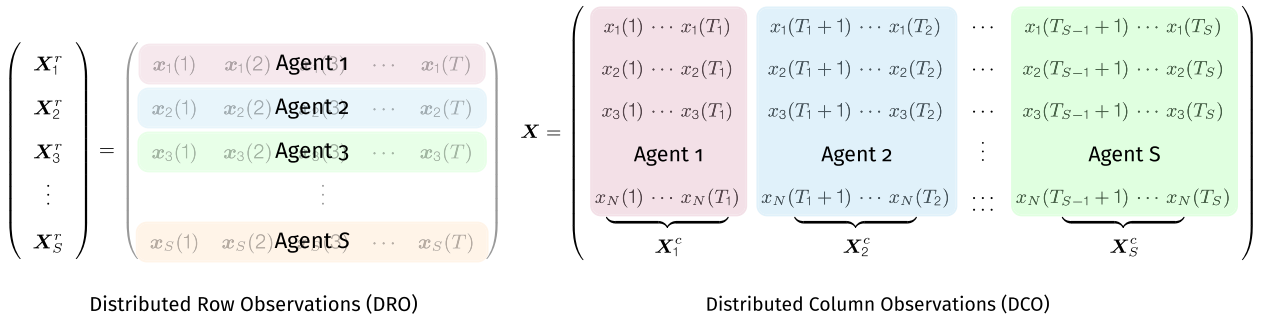
$$\min_{\mathbf{U} \in \mathbb{C}^{N \times p}} \sum_{i=1}^S \underbrace{\|(I - \mathbf{U}\mathbf{U}^H)\mathbf{X}_i^c\|_F^2}_{:=f_i(\mathbf{U})} \quad \text{s.t.} \quad \mathbf{U}^H \mathbf{U} = \mathbf{I} \quad (13)$$

which has a separable objective function similar to the consensus optimization problem tackled in [41]. In other words, our aim is to find a common dictionary based on all the data accrued across the network. Furthermore, we remark that the DCO data structure is only relevant in the static/batch PCA setting.

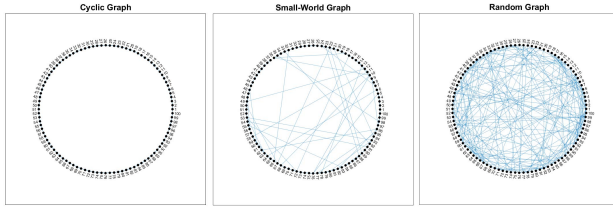
As mentioned before, the DRO setting is typically encountered in sensor networks where each agent, depending on its siting, captures components of the vector field that corresponds to the block  $\mathbf{x}_i(t)$ . For instance, the  $i$ th agent may have access to the  $i$ th antenna that received the signal  $\mathbf{x}_i(t)$ . In this case, we define our goal as follows:

Goal (DRO): Agent  $i$  learns the  $i$ th partition of the  $p$ -D principal subspace of  $\mathbf{X}$ , i.e., to learn  $\mathbf{U}_i^{r,*}$  in the partition  $\mathbf{U}^* = (\mathbf{U}_1^{r,*}; \dots; \mathbf{U}_S^{r,*})$ , in a distributed fashion.

This is a reasonable setting as  $\mathbf{U}_i^{r,*}$  keeps the components of  $\mathbf{U}^*$  that are related to the observations made at the  $i$ th agent. A typical example for DRO is the subspace estimation and tracking in a sensor or radar network. The main setting is an information fusion, which consists of many nodes that could be sensor nodes, arrays, and radars. Each node can be seen as a 1-D unit for receiving the signal, aiming at cooperatively estimating and tracking the principal subspace of the signals. Once the nodes have retrieved the PCA, the distributed projection on the principal subspace can also be distributed and all nodes



**Fig. 4.** Types of data partition employed in distributed PCA. Each data partition type will involve a different solution technique for distributed computation of the PCA.



**Fig. 5.** Examples of graph topology. (Left) A cyclic graph with  $S=100$  nodes and node degree 4. (Middle) A small-world graph with  $S=100$ , average degree 4, and rewiring probability 0.2. (Right) A random graph with  $S=100$  nodes and node degree 4.

will achieve a consensus on the coordinates of the PCA approximation  $z(t)$ .

The important observation here is that in the DRO setting the PCA problem is not separable like it is for the DCO setting, as seen in (13), and there is no consensus condition that ties the results the agents obtain.

Prior to reviewing the DCO and DRO algorithms in Sections IV and V, respectively, in the next section, we briefly introduce a building block that these algorithms rely on.

## B. Average Consensus Algorithm (Gossip Algorithm)

An important step taken by most distributed algorithms is to aggregate and share information across different agents. In the case for master-slave architecture, this can be done easily by a transmitting information directly from/to the agents with the central node. For meshed network, this has to be done with a sequence of computation steps adaptable to the network structure. In particular, in this section, we provide a brief introduction to the so-called average consensus algorithm (also known as the gossip algorithm) [32], [33], which is a key subroutine used in distributed PCA methods.

To proceed, it is necessary to define some terminology to refer to the communication network that connects the agents. As explained in the Introduction, we focus on the setting where the agents are connected on an arbitrarily meshed network. Specifically, the network used can be mapped onto a connected, undirected, and simple graph  $G = (\mathcal{V}, \mathcal{E})$ , where  $\mathcal{V} = \{1, \dots, S\}$  and  $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$  is the

set of communication links between the agents. See Fig. 5 for examples of a few common models to generate graph topologies.

Suppose that the  $i$ th agent holds a certain vector/matrix  $\mathbf{Y}_i$  and he/she wishes to compute the global average  $\mathbf{Y}_{\text{avg}} := S^{-1} \sum_{i=1}^S \mathbf{Y}_i$ . A distributed algorithm suitable for the task is the classical average consensus algorithm [32], [33], [42], also known as the gossip algorithm. In particular, for all  $\ell \geq 1$ , let  $\mathbf{W}[\ell]$  be a symmetric and doubly stochastic matrix (i.e., such that  $\mathbf{W}[\ell]\mathbf{1} = \mathbf{1}$  and  $\mathbf{1}^T \mathbf{W}[\ell] = \mathbf{1}^T$ ) whose sparsity matches that of adjacency matrix for the graph  $G$ , that is,  $W_{ij}[\ell] = W_{ji}[\ell] > 0$  only if  $(i, j) \in \mathcal{E}$ , and  $\mathbf{W}[\ell]\mathbf{1} = \mathbf{W}^T[\ell]\mathbf{1} = \mathbf{1}$ . We can compute  $\mathbf{Y}_{\text{avg}}$  distributively as follows.

### Average Consensus (AC) Algorithm:

- 1) Initialize as  $\mathbf{Z}_i[0] = \mathbf{Y}_i$  for all  $i$ .
- 2) For all agent  $i \in \{1, \dots, S\}$ , perform the recursion

$$\mathbf{Z}_i[\ell] = \sum_{j=1}^S W_{ij}[\ell] \mathbf{Z}_j[\ell-1] \quad (14)$$

for all  $\ell \geq 1$  and we terminate after  $\ell \geq L$ . The collection  $\{\mathbf{Z}_i[L]\}_{i=1}^S$  is retrieved as the output of the AC subroutine.

For simplicity, let us denote  $\mathbf{Z}_i[L]$ , i.e., the variable after the  $L$ th update, as output of the above subroutine

$$\{\mathbf{Z}_i[L]\}_{i=1}^S := \text{AC}(\{\mathbf{Y}_i\}_{i=1}^S; L) \quad (15)$$

where the first argument to the subroutine is the initialization given to the network and the second argument specifies the number of average consensus updates required. We also use  $\mathbf{Z}_i[L] := \text{AC}_i(\{\mathbf{Y}_i\}_{i=1}^S; L)$  to denote the output of the AC subroutine stored at the  $i$ th agent as indicated by the subscript. We remark that the AC subroutine defined in the above is applicable for computing the averages of scalar, vector, and matrices.

Above, we described the most general case where  $\mathbf{W}[\ell]$  changes over time, e.g., it models the scenario when some links in the network may be inactive at times. Formally, for every  $\ell \geq 1$ , this matrix is drawn from a distribution

with  $\mathbb{E}[\mathbf{W}[\ell]] = \bar{\mathbf{W}}$ . The following assumption states conditions for the AC subroutine to achieve asymptotic average consensus.

**Assumption 2:** For a given undirected graph  $G = (\mathcal{V}, \mathcal{E})$  with  $|\mathcal{V}| = S$  nodes and its associated weight matrix  $\bar{\mathbf{W}} \in \mathbb{R}^{S \times S}$ , the convergence conditions imposed on the network are as follows: 1) the graph  $G$  is connected; 2)  $\bar{\mathbf{W}}$  is doubly stochastic; and 3) the connectivity parameter  $\lambda_{\text{conn.}} < 1$ , where

$$\lambda_{\text{conn.}} := \max\{\lambda_2(\bar{\mathbf{W}}), -\lambda_S(\bar{\mathbf{W}})\} < 1. \quad (16)$$

An example is represented by the pairwise gossiping protocol introduced in [42], where an edge  $(i_\ell, j_\ell)$  is selected from  $\mathcal{E}$  uniformly at random and the agents compute a convex combination of their current state values which then becomes their new state. In this case

$$\mathbf{W}[\ell] = \mathbf{I} - \frac{1}{2}(\mathbf{e}_{i_\ell} - \mathbf{e}_{j_\ell})(\mathbf{e}_{i_\ell} - \mathbf{e}_{j_\ell})^\top \quad (17)$$

and the condition (16) will be satisfied as long as  $G$  is a connected graph.

An important feature of the average consensus algorithm is that its convergence is exponentially fast, i.e., we have  $\mathbb{E}[\|\mathbf{Z}_i[L] - \mathbf{Y}_{\text{avg}}\|] = \mathcal{O}(\lambda_{\text{conn.}}^L)$  for all  $i$  where the expectation is taken with respect to the realizations of  $\mathbf{W}[\ell]$ . To compute an  $\epsilon$ -accurate average, one only requires  $\Theta(\log \epsilon^{-1})$  average consensus updates.

## IV. DISTRIBUTED PCA METHODS FOR DCO

In this section, we survey some of the representative algorithms for distributed PCA in the DCO setting. As mentioned before, a distinguishing feature of the DCO scenario is that each agent keeps a (sub)set of the observed samples  $\{\mathbf{x}(t)\}_{t \in \mathcal{T}_i}$ , grouped into an  $N \times T_i$  matrix  $\mathbf{X}_i^c$ . When  $\mathbf{x}(t)$  are generated i.i.d. and  $T_i$  is large, each agent possesses sufficient data to compute the PCA locally. As such, we focus on the more restrictive case when  $T_i$  is not large and discuss the strategies in which the agents can leverage on additional information from other agents in the same network.

### A. Strategies for Master-Slave Architectures

A number of papers [7]–[17] tackled the distributed PCA problem (with DCO data) when the agents are organized according to a master-slave architecture, or equivalently a star network. As mentioned in the Introduction, these works keep the sets of data samples at the different servers sites and focus on the issue of reducing the overall computation complexity and communication cost required by the distributed methods.

As an example, Qu et al. [7] proposed to compute a local PCA from the partial data by having agent  $i$  perform an SVD of its own local subset of data matrix, deprived of

their mean, i.e.,

$$\mathbf{X}_i^c(\mathbf{I} - |\mathcal{T}_i|^{-1}\mathbf{1}\mathbf{1}^\top) = \mathbf{U}_i\mathbf{\Lambda}_i\mathbf{V}_i^H. \quad (18)$$

Let  $\tilde{\mathbf{U}}_i \in \mathbb{C}^{N \times p_i}$  be the matrix for the top  $p_i$  singular vector and  $\tilde{\mathbf{\Lambda}}_i$  be the principal  $p_i \times p_i$  submatrix of  $\mathbf{\Lambda}_i$ , where  $p_i \geq p$ . Agent  $i$  then transmits  $\tilde{\mathbf{U}}_i\tilde{\mathbf{\Lambda}}_i^2\tilde{\mathbf{U}}_i^H$  and the local mean  $\bar{\mathbf{x}}_i := |\mathcal{T}_i|^{-1}\mathbf{X}_i^c\mathbf{1}\mathbf{1}^\top$  to the central server, which estimates the global PCA  $\hat{\mathbf{U}} \in \mathbb{C}^{N \times p}$  from

$$\tilde{\mathbf{S}} = \sum_{i=1}^S \tilde{\mathbf{U}}_i\tilde{\mathbf{\Lambda}}_i^2\tilde{\mathbf{U}}_i^H + \sum_{i=1}^S |\mathcal{T}_i|(\bar{\mathbf{x}}_i - \bar{\mathbf{x}})(\bar{\mathbf{x}}_i - \bar{\mathbf{x}})^H. \quad (19)$$

We observe that this algorithm can be completed in one communication and computation round and its accuracy depends on the choice of  $p_i$ . On the other hand, it is clear that  $\hat{\mathbf{U}} \neq \mathbf{U}^*$  as  $\tilde{\mathbf{S}}$  does not share the principal subspaces as  $\mathbf{S}$  in general. In fact, the accuracy of the algorithm above can only be guaranteed when the data observed are homogeneous, e.g., for all  $i$ , each column of  $\mathbf{X}_i^c$  is generated from (2) with independently distributed modeling error  $e(t)$ . The merits of these approaches rest on the savings in computation cost when implemented in a parallel computer system. A similar approach can be found in [8] which combines the distributed PCA method with distributed  $K$ -means algorithms for further processing of the dimension reduced data.

### B. Strategies for Mesh Networks

From now on we focus on the mesh networks case. As pointed out in [17], a naïve distributed PCA method can be obtained by simply approximating the global correlation matrix via the AC subroutine, i.e., for some  $L \geq 1$ , we have

$$\hat{\mathbf{R}}_{x,i} = \frac{S}{T} \cdot \text{AC}_i(\{\mathbf{X}_j^c(\mathbf{X}_j^c)^H\}_{j=1}^S; L) \approx \hat{\mathbf{R}}_x. \quad (20)$$

In other words, each agent obtains an approximate of the global correlation matrix and the desired PCA can be then computed from  $\hat{\mathbf{R}}_{x,i}$  using, for example, the power method. The drawback, however, lies on the communication and computation cost entailed in this approach, which is particularly onerous when considering high-dimensional data with  $N \gg 1$ . This is because (20) requires computing the average of an  $N \times N$  matrix.

As an improvement over the correlation averaging method, next we describe a natural extension of the power method in (7) to the distributed (mesh network) setting with DCO data. In this setting, the distributed power method (DistPM) was introduced as a subroutine in [43] and [44]. Observe that the first expression in (7) can be written as

$$\begin{aligned} \mathbf{u}_1[k] &= \frac{1}{T} \mathbf{X} \mathbf{X}^H \bar{\mathbf{u}}_1[k-1] \\ &= \frac{1}{T} \sum_{i=1}^S \left( \mathbf{X}_i^c (\mathbf{X}_i^c)^H \bar{\mathbf{u}}_1[k-1] \right). \end{aligned} \quad (21)$$



If a copy of  $\bar{\mathbf{u}}_1[k-1]$  is also known to the agents, computing each of the terms requires data known to the  $i$ th agent and, therefore,  $\mathbf{u}_1[k]$  can simply be computed adding up the  $S$  terms—a computation that can be accomplished using the average consensus subroutine. This will be the case if the nodes start with the same initial vector  $\mathbf{u}_1[0]$  and achieve asymptotic convergence to the average. In reality, both conditions can be relaxed.

Specifically, for the DistPM with DCO data, we denote  $\mathbf{u}_1^i[k] \in \mathbb{C}^N$  (resp.,  $\bar{\mathbf{u}}_1^i[k]$ ) as the estimate of the first (resp., normalized) eigenvector of  $\mathbf{R}_x$  at the  $k$ th power iteration, kept by the  $i$ th agent. The DistPM (DCO) proceeds as follows.

#### Distributed Power Method (DistPM) for DCO:

- 1) Initialize for each agent an independent random vector, i.e.,  $\mathbf{u}_1^i[0] \sim \mathcal{CN}(\mathbf{0}, \mathbf{I})$  for all  $i$ .
- 2) For all agent  $i \in \{1, \dots, S\}$ , perform the recursion

$$\begin{aligned} \mathbf{u}_1^i[k] &= S \cdot \text{AC}_i(\{\mathbf{X}_j^c(\mathbf{X}_j^c)^H \bar{\mathbf{u}}_1^j[k-1]\}_{j=1}^S; L), \\ \bar{\mathbf{u}}_1^i[k] &= \mathbf{u}_1^i[k] / \|\mathbf{u}_1^i[k]\| \end{aligned}$$

for all  $k \geq 1$  and we terminate after  $k \geq K$ .

- 3) Denote  $\hat{\mathbf{u}}_1^i := \mathbf{u}_1^i[K]$  as the solution kept by the  $i$ th agent for all  $i \in \{1, \dots, S\}$ .

The subsequent eigenvectors  $\mathbf{u}_2^*, \dots, \mathbf{u}_p^*$  can be found by using a similar strategy as in the centralized power method.

*Theoretical Guarantee:* The convergence of DistPM (DCO) has been analyzed in [44] for real-valued data matrices and static network between the agents, i.e.,  $\mathbf{W}[\ell] = \bar{\mathbf{W}}$  for all  $\ell$ . For simplicity, we present the result for estimating the top eigenvector of  $\hat{\mathbf{R}}_x$ .

**Theorem IV.1:** Suppose that Assumptions 1 and 2 hold for  $p = 1$ . Let  $\hat{\mathbf{R}}_x = \frac{1}{T} \mathbf{X} \mathbf{X}^T$  be the sampled correlation matrix of the data matrix  $\mathbf{X} \in \mathbb{R}^{N \times T}$ . For a given failure probability  $c \in (0, 1)$  and an  $\epsilon$ -accuracy with  $\epsilon \in (0, 1/2)$ , if the number of AC iterations  $L$  and the number of power iterations  $K$  of the DistPM (DCO) algorithm satisfy

$$\begin{aligned} L &= \Omega \left( \frac{\log \epsilon^{-1} + \log(\lambda_1(\hat{\mathbf{R}}_x) - \lambda_2(\hat{\mathbf{R}}_x))^{-1}}{\log(\lambda_{\text{conn}}^{-1})} \right) \\ K &= \Omega \left( \frac{\lambda_1(\hat{\mathbf{R}}_x)}{\lambda_1(\hat{\mathbf{R}}_x) - \lambda_2(\hat{\mathbf{R}}_x)} \cdot \log \left( \frac{N}{c \cdot \epsilon} \right) \right) \end{aligned} \quad (22)$$

and the initialization satisfies  $(\mathbf{u}_1^*)^T \mathbf{u}_1^i[0] \neq 0$ , then with probability at least  $1 - Sc$  where  $S$  is the number of nodes in the network, the estimated top eigenvector  $\hat{\mathbf{u}}_1^i$  at each node  $i$  is  $\epsilon$ -accurate

$$\sqrt{1 - ((\mathbf{u}_1^*)^T \hat{\mathbf{u}}_1^i)^2} \leq \epsilon \quad (23)$$

and  $\min_{j=2, \dots, N} ((\mathbf{u}_j^*)^T \hat{\mathbf{u}}_1^i)^2 \leq \epsilon^2$  for all  $i \in \{1, \dots, S\}$ . Additionally, the eigenvectors computed at each node are in consensus

$$\|\hat{\mathbf{u}}_1^i(\hat{\mathbf{u}}_1^{i'})^T - \hat{\mathbf{u}}_1^{i'}(\hat{\mathbf{u}}_1^i)^T\| = \mathcal{O}(\epsilon), \quad \forall i, i' \in \{1, \dots, S\}. \quad (24)$$

The randomness in the algorithm is due to the random initializations. The theorem above is taken from [44, Prop. 1] and was partially inspired by [45]. Theorem IV.1 gives a nonasymptotic bound on the accuracy achieved by the DistPM (DCO) for a given  $K$ . In addition, Theorem IV.1 depicts a communication/performance tradeoff for the DistPM (DCO) method. Specifically, the total communication cost for the method is given by  $LK$ —when the desired accuracy  $\epsilon$  is small, the required communication cost increases in the order of  $\log(1/\epsilon)$  as seen in (22).

It is worth noting that an asynchronous version of the DistPM (DCO) method has been studied in [29], which showed that the method converges asymptotically to the correct solution even when the agents communicate in a pairwise gossip exchange manner.

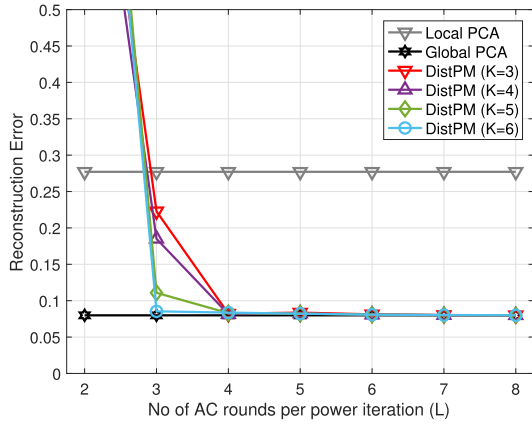
## C. Application Examples

We consider applying PCA to dimensionality reduction and clustering in a distributed setting. Particularly, we assume that the  $i$ th machine/agent observes a data set which can be described as

$$\mathbf{X}_i^c = \mathbf{A} \mathbf{Z}_i^c + \mathbf{E}_i \in \mathbb{R}^{N \times T_i} \quad (25)$$

where each row of  $\mathbf{X}_i^c$  represents a feature and each column is a data point. In the above,  $\mathbf{A} \in \mathbb{R}^{N \times p}$  is a common “dictionary” with  $p \ll N$ ,  $\mathbf{Z}_i^c \in \mathbb{R}^{p \times T_i}$  is the latent parameters for the observations, and  $\mathbf{E}_i$  represents the modeling error. Furthermore,  $\mathbf{X}_i^c$  constitutes a set of training data, and our aim is to learn the subspace spanned by the columns of  $\mathbf{A}$  so that we can compress some unseen data  $\mathbf{X}_{\text{test}}$ , which are also generated by the same model, using their  $p$ -D latent parameters. Specifically, if  $\mathbf{U}^*$  is the learned  $p$ -D subspace from the collection of training data  $\{\mathbf{X}_i^c\}_{i=1}^S$ , then  $\mathbf{X}_{\text{test}}$  can be compressed as  $\mathbf{Z}_{\text{test}} := (\mathbf{U}^*)^H \mathbf{X}_{\text{test}}$ . We assume that the modeling errors  $\mathbf{E}_i$  are independent, and the dimensionality reduction problem can now be solved as a static PCA problem (3). Furthermore, our target problem corresponds to the DCO setting considered in (13) as the data points are collected locally by the machines.

To illustrate the performance of the distributed PCA method (DistPM), we consider the data model of (25) with  $N = 1000$ ,  $p = 2$ , and  $T_i = 2$ , while the SNR is fixed at 25 dB. We consider a network with  $S = 100$  nodes, connected through a mean degree-10 small world graph with rewiring probability of 0.2. Fig. 6 shows the reconstruction error in different settings for the DistPM (DCO) with a different number of AC rounds.



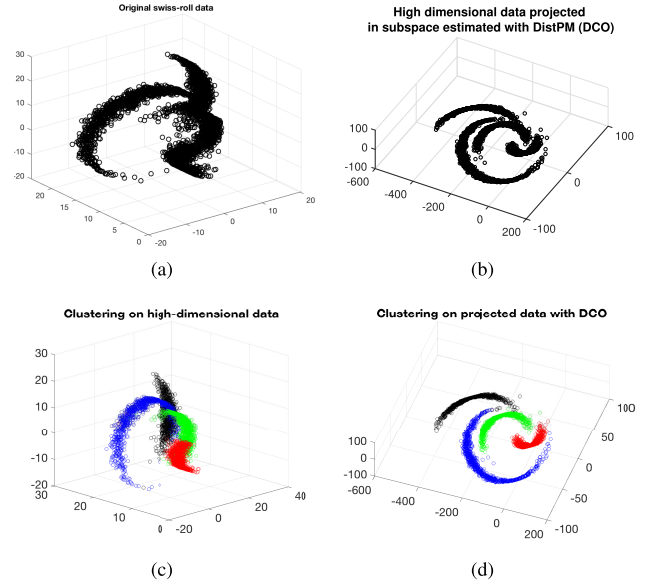
**Fig. 6.** Applying DistPM (DCO) on the dimensionality reduction problem. The reconstruction error is shown for agent  $i=1$  on the testing data, which is generated according to a similar model as (25). In the legend, “local PCA” (resp., “global PCA”) refers to using only the local data (resp., all available data) to estimate the principal subspace.

The development of distributed dimensionality reduction techniques provides both efficient storage and pre-processing (denoising) of the data for machine learning. Fig. 7 shows an application of dimensionality reduction in data clustering. We consider a set of high-dimensional data distributed over  $S = 100$  nodes. Specifically, the data consist of four separated clusters. Each cluster is a high-dimensional embedding of a swiss roll using the model in (25) with  $N = 200$ . Fig. 7(a) shows the original clusters of swiss roll data that are used to generate the high-dimensional data. Fig. 7(b) shows the result of dimensionality reduction using the estimated subspace with DistPM (DCO). We then perform Ward’s hierarchical clustering algorithm [46] on both the original high-dimensional data and the projected data.<sup>2</sup> The normalized mutual information (NMI) is used to measure the clustering performance. Fig. 7(b) and (c) shows the detected clusters with  $\text{NMI} = 0.9797$  and  $0.9785$ , respectively.

## V. DISTRIBUTED PCA METHODS FOR DRO

In this section, we focus on the DRO setting and survey the available distributed PCA strategies in the literature. An interesting feature under this setting is that the goal of each agent is to compute a portion of the orthogonal transformation  $U^*$  required. This requires one to design algorithms carefully in order to take advantage of these requirements while respecting the DRO data structure. To the best of our knowledge, most of the prior work for the DRO data structure has been designed for the mesh network scenario. Therefore, it will be the main focus of the current section.

<sup>2</sup>Here we let the nodes to broadcast their local projected data to the network.



**Fig. 7.** Clustering on the swiss roll data. (a) Original swiss roll data before high-dimensional embedding. (b) High-dimensional data projected in 3-D subspace estimated with DistPM (DCO). (c) Results of hierarchical clustering on the original high-dimensional data. (d) Results of hierarchical clustering on the projected data with DistPM (DCO).

## A. Strategy for Master–Slave Architecture

Consider the case when the communication network follows a master–slave architecture. As the data from each agent are necessary in this case and there is no direct agent-to-agent link, aggregating all data (incrementally or in batches) at the master node becomes an unavoidable step for solving the PCA problem. Acknowledging this, the simplest approach is to directly apply the centralized PCA methods (e.g., power method) after the data aggregation at the master node, and then transmit the required portion of  $U^*$  back to the agents. Such approach works off-the-shelf and the convergence properties follow directly from the analysis on centralized methods (cf., Section II).

## B. Distributed Power Method (DRO)

The distributed power method (DistPM) was developed in [18] for the DRO setting. In a nutshell, the method combines average consensus and the power method to estimate the principal subspace distributively. Note that this is a batch method that is applied to stationary data.

We first derive the DistPM for just the top eigenvector of  $\hat{R}_x$ , denoted as  $u_1^*$ . To set up the stage, we define  $u_{1,i}[k]$  as the local portion of the top eigenvector kept by the  $i$ th agent at iteration  $k$  and  $u_1[k] := (u_{1,1}[k]; \dots; u_{1,S}[k])$  as the global eigenvector estimate. To initialize, each agent generates a random vector  $u_{1,i}[0] \sim \mathcal{CN}(0, I) \in \mathbb{C}^{N_i}$  independently.

Consider the  $k$ th power iteration in (7); we see that the normalization step may be relegated to the end of the iterations as it only involves a scalar multiplication.

Instead, we focus on the second step in (7). Observe that the  $i$ th row partition in  $\mathbf{u}_1[k+1]$  can be written as

$$\frac{1}{T} \sum_{t=1}^T \mathbf{x}_i(t) \mathbf{x}^H(t) \mathbf{u}_1[k] = \frac{1}{T} \sum_{t=1}^T \mathbf{x}_i(t) \left( \sum_{j=1}^S \mathbf{x}_j^H(t) \mathbf{u}_{1,j}[k] \right). \quad (26)$$

Note that we are using the unnormalized vector  $\mathbf{u}_{1,j}[k]$  on the right-hand side of the equation. The vectors  $\mathbf{x}_j(t)$  and  $\mathbf{u}_{1,j}[k]$  are known to the  $j$ th agent. Therefore, the inner products inside the summation are locally computable by the  $j$ th agent. This inspires us to replace the summation with the previously introduced AC subroutine. Specifically, the required computation can be performed by the following pseudocode.

**Distributed Power Method (DistPM) for DRO:**

- 1) Initialize for each agent an independent random vector, i.e.,  $\mathbf{u}_{1,i}[0] \sim \mathcal{CN}(0, \mathbf{I}) \in \mathbb{C}^{N_i}$  for all  $i$ .
- 2) For all agent  $i \in \{1, \dots, S\}$ , perform the recursion

$$\mathbf{u}_{1,i}[k+1] = \frac{S}{T} \sum_{t=1}^T \mathbf{x}_i(t) \text{AC}_i(\{\mathbf{x}_j^H(t) \mathbf{u}_{1,j}[k]\}_{j=1}^S; L)$$

for all  $k \geq 0$  and we terminate after  $k \geq K-1$ .

- 3) Denote  $\mathbf{u}_{1,i}[K]$  as the unnormalized top eigenvector kept by the  $i$ th agent for all  $i \in \{1, \dots, S\}$ .

Note that we have replaced the inner product  $\mathbf{x}^H(t) \mathbf{u}_1[k]$  by its distributed average approximation  $\text{AC}_i(\{\mathbf{x}_j^H(t) \mathbf{u}_{1,j}[k]\}_{j=1}^S; L)$ .

The procedure above finds the  $i$ th block of the unnormalized top eigenvector of  $\hat{\mathbf{R}}_x$ . As a final step, we will take care of the normalization factor which we have ignored. Observe that

$$\|\mathbf{u}_1[K]\| = \sqrt{\sum_{i=1}^S \|\mathbf{u}_{1,i}[K]\|^2}. \quad (27)$$

Once again we observe that the terms inside the summation are merely the squared norm of the local estimate  $\mathbf{u}_{1,i}[K]$ , which is obviously known to the  $i$ th agent. This can be replaced by another AC subroutine. Let us define

$$\text{norm}_i := \sqrt{S \cdot \text{AC}_i(\{\|\mathbf{u}_{1,j}[K]\|^2\}_{j=1}^S; L')} \approx \|\mathbf{u}_1[K]\| \quad (28)$$

where we set  $L' \gg L$ . Finally, we denote the obtained partial eigenvector as  $\hat{\mathbf{u}}_{1,i}$  such that

$$\hat{\mathbf{u}}_{1,i} := (\text{norm}_i)^{-1} \mathbf{u}_{1,i}[K] \quad (29)$$

and the concatenated version of it as  $\hat{\mathbf{u}}_1 := (\hat{\mathbf{u}}_{1,1}; \dots; \hat{\mathbf{u}}_{1,S})$ . It is worth noting that the normalization procedure (29) is done at the last step when acquiring each eigenvector. This is done so to reduce the communication cost without sacrificing the performance since the scalar

multiplication does not modify the subspace found by the algorithm.

Moreover, we can also compute the eigenvalue  $\lambda_1$  associated with  $\mathbf{u}_1^*$  by observing

$$\lambda_1 = \hat{\mathbf{u}}_1^H \left( \frac{\frac{1}{T} \sum_{t=1}^T \mathbf{x}(t) \mathbf{x}^H(t) \hat{\mathbf{u}}_1}{\approx (\mathbf{u}_{1,1}[K]; \dots; \mathbf{u}_{1,S}[K])} \right). \quad (30)$$

The above can be approximated by

$$\hat{\lambda}_{1,i} = S \cdot \text{AC}_i(\{\hat{\mathbf{u}}_{1,j}^H \mathbf{u}_{1,j}[K]\}_{j=1}^S; L). \quad (31)$$

However, we note that this step is optional as the eigenvalue is not required in the computation of the second, third, etc., eigenvectors.

To compute the second eigenvector  $\mathbf{u}_2^*$ , we observe that it is equivalent to computing the top eigenvector for  $(\mathbf{I} - \mathbf{u}_1^*(\mathbf{u}_1^*)^H) \hat{\mathbf{R}}_x$ . In light of this, the respective power iteration can be approximated as

$$\mathbf{u}_2[k+1] = \frac{1}{T} \sum_{t=1}^T (\mathbf{I} - \hat{\mathbf{u}}_1 \hat{\mathbf{u}}_1^H) \mathbf{x}(t) \mathbf{x}^H(t) \mathbf{u}_2[k]. \quad (32)$$

Similar to the previous derivations in (26), the  $i$ th row block of the above can be written as

$$\frac{1}{T} \sum_{t=1}^T \left( \mathbf{x}_i(t) - \underbrace{\hat{\mathbf{u}}_{1,i} \hat{\mathbf{u}}_1^H \mathbf{x}(t)}_{\approx \text{prod}_{1,i}} \right) \underbrace{\left( \mathbf{x}^H(t) \mathbf{u}_2[k] \right)}_{\approx \text{prod}_{2,i}(k)}. \quad (33)$$

Importantly, the highlighted inner products are replaced, respectively, by:

$$\begin{aligned} \text{prod}_{1,i} &:= S \cdot \text{AC}_i(\{\hat{\mathbf{u}}_{1,j}^H \mathbf{x}_j(t)\}_{j=1}^S; L) \\ &\approx \hat{\mathbf{u}}_{1,i}^H \mathbf{x}(t) \\ \text{prod}_{2,i}(k) &:= S \cdot \text{AC}_i(\{(\mathbf{x}_j(t))^H \mathbf{u}_{2,j}[k]\}_{j=1}^S; L) \\ &\approx \mathbf{x}^H(t) \mathbf{u}_2[k] \end{aligned} \quad (34)$$

where the right-hand side can be computed distributively using the AC subroutine. Finally, the power iteration is performed by

$$\mathbf{u}_{2,i}[k+1] = \frac{1}{T} \sum_{t=1}^T \left( \mathbf{x}_i(t) - \hat{\mathbf{u}}_{1,i} \cdot \text{prod}_{1,i} \right) \text{prod}_{2,i}[k]. \quad (35)$$

We can compute  $\hat{\mathbf{u}}_{3,i}, \dots, \hat{\mathbf{u}}_{p,i}$  using a similar procedure, again in a distributed fashion.

**Theoretical Guarantees:** The asymptotic convergence of DistPM (DRO) for the static network, i.e.,  $\mathbf{W}[\ell] = \overline{\mathbf{W}}$  for all  $\ell$ , has been analyzed in [21]. Here, we present a simplified version of their result on seeking for the top eigenvector exclusively i.e., when  $p = 1$ . Importantly, we observe that when  $L \rightarrow \infty$  such that the AC subroutine computes the exact averages, the DistPM method will be equivalent to the original/centralized power method.

Focusing on the error performance with a finite  $L$  for the DistPM method, the analysis in [21, Th. 3] shows the following.

**Theorem V.1:** Under Assumptions 1 and 2 with  $p = 1$ , assume that the initialization satisfies  $|(\mathbf{u}_1^*)^H \mathbf{u}_1[0]| > 0$ . Let  $\hat{\mathbf{R}}_x = \frac{1}{T} \mathbf{X} \mathbf{X}^H$  be the sampled correlation matrix of the data matrix  $\mathbf{X} \in \mathbb{C}^{N \times T}$ . For a given number of AC iterations  $L$ , as the number of power iterations  $K \rightarrow \infty$ , we have

$$\|\hat{\mathbf{u}}_1 - \mathbf{u}_1^*\| = \mathcal{O} \left( \frac{\lambda_1(\hat{\mathbf{R}}_x)}{\lambda_1(\hat{\mathbf{R}}_x) - \lambda_2(\hat{\mathbf{R}}_x)} \cdot \lambda_{\text{conn.}}^L \right). \quad (36)$$

Theorem V.1 bounds the difference between the principal eigenvector  $\mathbf{u}_1^*$  and the estimated principal eigenvector  $\hat{\mathbf{u}}_1$ . Note that the DistPM's performance depends on the number of AC iterations  $L$  and in particular, the performance improves exponentially with  $L$ . Moreover, the error also depends on the spectral gap in  $\hat{\mathbf{R}}_x$  such that the error is reduced when  $\lambda_1(\hat{\mathbf{R}}_x) \gg \lambda_2(\hat{\mathbf{R}}_x)$ .

### C. Distributed Oja's Method

As opposed to the distributed power method, the Oja's method can be applied to the dynamic PCA problem (4) which is a nonconvex, time-varying, stochastic optimization problem. A distributed version of the Oja's method, named D-Oja, was developed in [23] for the DRO setting. Here, we focus on the setting with  $p = 1$ . The first step is to approximate the update (10) as

$$\begin{aligned} \mathbf{u}^{\text{Oja}}(t+1) &= \mathbf{u}^{\text{Oja}}(t) + \gamma_t \left( \mathbf{x}(t) \mathbf{x}^H(t) - |\mathbf{x}^H(t) \mathbf{u}^{\text{Oja}}(t)|^2 \mathbf{I} \right) \\ &\quad \times \mathbf{u}^{\text{Oja}}(t). \end{aligned} \quad (37)$$

Similar to the development of DistPM, the D-Oja method relies on interpreting the Oja's updates as a combination of several computations that are replaceable by distributed algorithms. In particular, consider the  $i$ th block of  $\mathbf{u}^{\text{Oja}}(t+1)$  in (37)

$$\begin{aligned} \mathbf{u}_i^{\text{Oja}}(t+1) &= \mathbf{u}_i^{\text{Oja}}(t) + \gamma_t \left( \mathbf{x}_i(t) \cdot \frac{\mathbf{x}^H(t) \mathbf{u}^{\text{Oja}}(t) - \mathbf{u}_i^{\text{Oja}}(t)}{|\mathbf{x}^H(t) \mathbf{u}^{\text{Oja}}(t)|^2} \right) \\ &\quad \cdot \frac{|\mathbf{x}^H(t) \mathbf{u}^{\text{Oja}}(t)|^2}{|\mathbf{x}^H(t) \mathbf{u}^{\text{Oja}}(t)|^2}. \end{aligned} \quad (38)$$

In particular, for the  $i$ th agent, it suffices to compute the complex-valued inner product  $\mathbf{x}^H(t) \mathbf{u}^{\text{Oja}}(t)$  in order to complete the update. To distribute the computation, again we observe that this inner product is computed from a number of local terms available at each agent

$$\begin{aligned} \mathbf{x}^H(t) \mathbf{u}^{\text{Oja}}(t) &= \sum_{i=1}^S \mathbf{x}_i^H(t) \mathbf{u}_i^{\text{Oja}}(t) \\ &\approx S \cdot \text{AC}_i(\{\mathbf{x}_j^H(t) \mathbf{u}_j^{\text{Oja}}(t)\}_{j=1}^S; L). \end{aligned} \quad (39)$$

The D-Oja's method for DRO is summarized in the following pseudocode.

#### Distributed Oja's (D-Oja) Method for DRO:

- 1) Initialize for each agent an independent random vector, i.e.,  $\mathbf{u}_i^{\text{Oja}}(0) \sim \mathcal{CN}(0, \mathbf{I}) \in \mathbb{C}^{N_i}$  for all  $i$ .
- 2) For all agent  $i \in \{1, \dots, S\}$ , perform the recursion

$$\begin{aligned} \text{prod}_i(t) &:= S \cdot \text{AC}_i(\{\mathbf{x}_j^H(t) \mathbf{u}_j^{\text{Oja}}(t)\}_{j=1}^S; L), \\ \mathbf{u}_i^{\text{Oja}}(t+1) &= \mathbf{u}_i^{\text{Oja}}(t) + \gamma_t \left( \text{prod}_i(t) \cdot \mathbf{x}_i(t) \right. \\ &\quad \left. - |\text{prod}_i(t)|^2 \cdot \mathbf{u}_i^{\text{Oja}}(t) \right) \end{aligned} \quad (40)$$

for all  $t \geq 0$ .

For general  $p > 1$ , Li et al. [23] also proposed an alternative method that solves a generalized objective function given in (4).

*Theoretical Guarantees:* The asymptotic (i.e.,  $t \rightarrow \infty$ ) convergence of the D-Oja method has been analyzed in [23] under the assumption that  $\mathbf{x}(t)$  is stationary. Similar to the convergence analysis of the Oja's algorithm, Li et al. [23] investigated the convergence property by deriving the associated ordinary differential equation (ODE) of the stochastic approximation in (40). That is, for  $\gamma_t$  sufficiently small,<sup>3</sup> the associated ODE of the stochastic approximation in (40) is given by

$$\begin{aligned} \frac{d\hat{\mathbf{u}}_1}{dt} &= \mathbb{E}([\mathbf{d}_1(t); \mathbf{d}_2(t); \dots; \mathbf{d}_S(t)]) \\ &= \left\{ S \bar{\mathbf{W}}^L \odot \mathbf{R} - S \bar{\mathbf{W}}^L \odot \left[ (N \bar{\mathbf{W}}^L \odot \hat{\mathbf{u}}_1 \hat{\mathbf{u}}_1^H) \mathbf{R} \right] \right\} \hat{\mathbf{u}}_1 \end{aligned} \quad (41)$$

where  $\mathbf{R} = \mathbb{E}[\mathbf{x}(t) \mathbf{x}(t)^H]$ . The following results state the stability and convergence property of the ODE (41).

**Theorem V.2:** Suppose that Assumptions 1 and 2 hold for  $p = 1$ . For any fixed number of AC iterations  $L$  satisfying  $\{L \in \mathbb{N}_+ | \lambda_{\text{conn.}}^L \ll 1/(2S)\}$ , the equilibrium point of the norm  $\|\hat{\mathbf{u}}_1\|_F^2$  is contained in the interval

$$\|\hat{\mathbf{u}}_1\|^2 \in \left( 1 - 3N\lambda_{\text{conn.}}^K, 1 + 3N\lambda_{\text{conn.}}^K \right). \quad (42)$$

**Theorem V.3:** Suppose that Assumptions 1 and 2 hold for  $p = 1$ . The ODE in (41) has a stable equilibrium point (i.e., a sink point)  $\hat{\mathbf{u}}_1$  that satisfies

$$\|\hat{\mathbf{u}}_1 - \mathbf{u}_1^*\| \leq 3S\lambda_1(\hat{\mathbf{R}}_x) \cdot \lambda_{\text{conn.}}^L + \mathcal{O}(S^2 \lambda_{\text{conn.}}^{2L}). \quad (43)$$

The proof can be found in [23, Lemma 4.1 and 4.2]. Similar to the results for DistPM (DRO), Theorem 5.3 also shows an exponential dependence on the number of AC steps  $L$  in the error produced by the D-Oja method.

The asynchronous version of D-Oja has been studied in [23] and [29]. In particular, Li et al. [23] showed the

<sup>3</sup>Example step sizes include but are not limited to diminishing step sizes (i.e.,  $\gamma_t = c/t$  for a constant  $c$ ), nonsummable diminishing step sizes (i.e.,  $\sum_t \gamma_t = \infty, \sum_t \gamma_t^2 < \infty$ ), and a small constant based on Euler's method.



asymptotic convergence of the method, while Fellus *et al.* [29] have extended the method with a sparsification step to reduce the communication overhead and showed the extended method converges with probability 1.

#### D. Power-Oja's Method

As discussed earlier on, the Oja's method (as well as the D-Oja method) is known to converge slowly, i.e., at most at the rate of  $\mathcal{O}(1/t)$ . To accelerate the convergence rate, Wu *et al.* [25] proposed a Power-Oja's (P-Oja) method that combines the fast convergence of power method with the adaptivity of the Oja's method. Note that the power method is a batch method that requires the random process  $x(t)$  to be stationary. As a compromise, we assume quasi-stationarity for  $x(t)$  such that the process is stationary over the period  $\mathcal{I}_k$  in which  $x(\tau)$  has the same distribution for all  $\tau \in \mathcal{I}_k$ . Furthermore, in contrast to the Oja's method, the P-Oja's method tackles the following dynamic PCA problem:

$$U^*(k) \in \arg \min_{U \in \mathbb{C}^{N \times p}} f_\tau(U) \text{ s.t. } U^H U = I \quad (44)$$

where  $\tau \in \mathcal{I}_k$  and

$$f_\tau(U) := \mathbb{E} \left[ \text{Tr} \left( (U U^H U U^H - 2 U U^H) x(\tau) x^H(\tau) \right) \right]. \quad (45)$$

It can be shown that the above problem is equivalent to (4).

We apply stochastic approximation for (44) to obtain a tractable objective function. Let  $\mathcal{B}_{k,s} \subseteq \mathcal{I}_k$  be the  $s$ th batch in period  $\mathcal{I}_k$ , then the objective function in the above can be approximated by

$$\hat{f}(U; \mathcal{B}_{k,s}) := \text{Tr} \left( (U U^H U U^H - 2 U U^H) \hat{R}(\mathcal{B}_{k,s}) \right) \quad (46)$$

where

$$\hat{R}(\mathcal{B}_{k,s}) := \frac{1}{|\mathcal{B}_{k,s}|} \sum_{\tau \in \mathcal{B}_{k,s}} x(\tau) x^H(\tau) \quad (47)$$

which is a mini-batch sampled correlation matrix.

From here, one may apply the SGD method and update the principal subspaces by taking the gradient of  $\hat{f}(U; \mathcal{B}_{k,s})$ . Doing so results in a method similar to Oja's which may suffer from the same slow convergence rate. Instead, let  $P$  be a predefined integer constant, then we consider a modified stochastic approximation function

$$\hat{f}^{\text{POja}}(U; \mathcal{B}_{k,s}) := \text{Tr} \left( (U U^H U U^H - 2 U U^H) \hat{R}^P(\mathcal{B}_{k,s}) \right). \quad (48)$$

Importantly, the optimal solution set to the modified stochastic approximation function is the same as the original one

$$\begin{aligned} \arg \min_{U \in \mathbb{C}^{N \times p}} \hat{f}(U; \mathcal{B}_{k,s}) &= \arg \min_{U \in \mathbb{C}^{N \times p}} \hat{f}^{\text{POja}}(U; \mathcal{B}_{k,s}) \\ \text{s.t. } U^H U &= I \quad \text{s.t. } U^H U = I. \end{aligned} \quad (49)$$

The benefit of considering  $\hat{f}^{\text{POja}}(\cdot)$  in lieu of  $\hat{f}(\cdot)$  is that the former admits a better spectral gap for the sampled correlation since

$$\frac{\sigma_p(\hat{R}^P) - \sigma_{p+1}(\hat{R}^P)}{\sigma_p(\hat{R}^P)} > \frac{\sigma_p(\hat{R}) - \sigma_{p+1}(\hat{R})}{\sigma_p(\hat{R})} \quad (50)$$

as we note that  $\sigma_p(\hat{R}^P(\mathcal{B}_{k,s})) = \sigma_p(\hat{R}(\mathcal{B}_{k,s}))^P$ , where  $\sigma_p(\mathbf{R})$  denotes the  $p$ th largest singular value of  $\mathbf{R}$ . As shown in the analysis of [38], the size of the spectral gap  $\sigma_p(\hat{R}^P(\mathcal{B}_{k,s})) - \sigma_{p+1}(\hat{R}^P(\mathcal{B}_{k,s}))$  is an important factor in determining the convergence speed of subspace estimation/tracking algorithms. Therefore, we anticipate that using  $\hat{f}^{\text{POja}}(\cdot)$  should yield a faster PCA algorithm than using  $\hat{f}^{\text{Oja}}(\cdot)$ . Finally, observe that the gradient of the new stochastic approximation function is

$$\begin{aligned} \nabla \hat{f}^{\text{POja}}(U; \mathcal{B}_{k,s}) &= -2 \hat{R}^P(\mathcal{B}_{k,s}) U + \hat{R}^P(\mathcal{B}_{k,s}) U U^H U \\ &\quad + U U^H \hat{R}^P(\mathcal{B}_{k,s}) U. \end{aligned} \quad (51)$$

Remarkably,  $\hat{R}^P(\mathcal{B}_{k,s}) U$  is similar to running a power method for  $P$  steps with the columns of  $U$  as initialization vectors (an insight that motivated this distributed PCA approach). Last, relaxing the Grassmanian manifold constraint  $U^H U = I$  yields the P-Oja method

$$U(k, s+1) = U(k, s) - \gamma_s \nabla \hat{f}^{\text{POja}}(U(k, s); \mathcal{B}_{k,s}) \quad (52)$$

which is similar to the NOja's method considered in [39].

Note that the P-Oja method requires a nontrivial batch size  $|\mathcal{B}_{k,s}| \geq 2$  to show its benefits, which depend on a sufficient increase in the spectral gap.

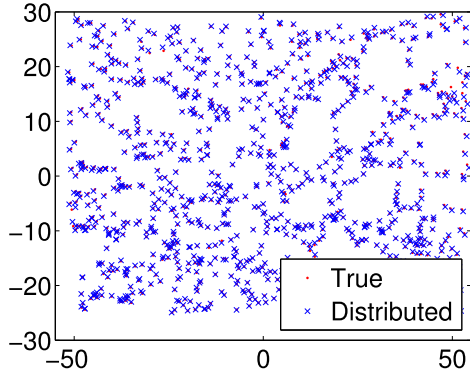
*Distributed Method:* Wu *et al.* [25] considered a distributed extension for the P-Oja method. Their philosophy is similar to that employed by the DistPM and D-Oja methods, as we observe that the update of the  $i$ th block [cf., (51)] is

$$U_i(s) - \gamma_s [\nabla \hat{f}^{\text{POja}}(U(s); \mathcal{B}_s)]_i \quad (53)$$

where we have dropped the index of  $k$  for simplicity and

$$\begin{aligned} &[\nabla \hat{f}^{\text{POja}}(U(s); \mathcal{B}_s)]_i \\ &= -2 \underbrace{[\hat{R}^P(\mathcal{B}_s) U(s)]_i}_{\approx \text{prod}_{1,i}(s)} + \underbrace{[\hat{R}^P(\mathcal{B}_s) U(s)]_i U^H(s) U(s)}_{\approx \text{prod}_{1,i}(s)} \\ &\quad + U_i(s) \underbrace{U^H(s) \hat{R}^P(\mathcal{B}_s) U(s)}_{\approx \text{prod}_{3,i}(s)}. \end{aligned} \quad (54)$$

The required computation boils down to evaluating the highlighted terms in the above. First, we observe that  $\hat{R}^P(\mathcal{B}_s) U(s)$  is the result of applying  $P$  power iterations with the sampled correlation matrix  $\hat{R}(\mathcal{B}_s)$ , initialized on  $U(s)$ . Each of the power iterations can be handled in a similar fashion as in the DistPM. In particular, let  $\hat{U}(s, q)$



**Fig. 8.** Distributed distance matrix estimation. Comparing the true and estimated distance matrix on a sensor network with  $S=1000$  sensors. We set  $K=50$ ,  $L=500$  for the DistPM and reconstructed the distance matrix through finding its rank-2 approximation.

be the intermediate variable at the  $q$ th power iteration,  $\hat{U}_i(s, q)$  be its  $i$ th row block, and  $\hat{U}(s, 1) = U(s)$ , then the power iteration can be approximated as

$$\begin{aligned} \hat{U}_i(s, q+1) &= \frac{S}{|\mathcal{B}_s|} \sum_{\tau \in \mathcal{B}_s} \mathbf{x}_i(\tau) \cdot \text{AC}_i(\{\mathbf{x}_j^H(\tau) \hat{U}_j(s, q)\}_{j=1}^S; L) \\ &\approx \frac{1}{|\mathcal{B}_s|} \sum_{\tau \in \mathcal{B}_s} \mathbf{x}_i(\tau) \mathbf{x}^H(\tau) \hat{U}(s, q). \end{aligned} \quad (55)$$

Repeating the above for  $P$  iterations yields

$$\text{prod}_{1,i}(s) := \hat{U}_i(s, P+1) \approx [\hat{\mathbf{R}}^P(\mathcal{B}_s)U(s)]_i. \quad (56)$$

For the remaining terms, we can handle them easily by

$$\begin{aligned} \text{prod}_{2,i}(s) &:= S \cdot \text{AC}_i(\{\mathbf{U}_j^H(s) \mathbf{U}_j(s)\}_{j=1}^S; L) \\ &\approx \sum_{j=1}^S \mathbf{U}_j^H(s) \mathbf{U}_j(s), \\ \text{prod}_{3,i}(s) &:= S \cdot \text{AC}_i(\{\mathbf{U}_j^H(s) \text{prod}_{1,j}(s)\}_{j=1}^S; L) \\ &\approx \sum_{j=1}^S \mathbf{U}_j^H(s) \text{prod}_{1,j}(s) \approx \sum_{j=1}^S \mathbf{U}_j^H(s) [\hat{\mathbf{R}}^P U(s)]_j. \end{aligned} \quad (57)$$

Finally, the P-Oja update for agent  $i$  can be approximated as

$$\begin{aligned} U_i(s+1) &= U_i(s) - \gamma_s \left( U_i(s) \text{prod}_{3,i}(s) \right. \\ &\quad \left. + \text{prod}_{1,i}(s) \text{prod}_{2,i}(s) - 2\text{prod}_{1,i}(s) \right). \end{aligned} \quad (59)$$

This demonstrates that the P-Oja update can be replaced by a number of average consensus steps to be computed distributively. We remark that the P-Oja update can be reduced to the multiple subspaces-tracking D-Oja method (i.e., when  $p > 1$ ) by setting  $P = 1$ . As a reminder, herein we restate that unlike DistPM, D-Oja and distributed P-Oja are adaptive methods that are suitable for tracking subspace for streaming data. Currently, the convergence of the P-Oja method is yet to be analyzed theoretically.

## E. Application Examples

1) *Distributed Distance Matrix Estimation:* We demonstrate that the distance matrix estimation problem can be cast as a PCA problem, which naturally admits a DRO data structure when we consider a sensor network setting. Here, agent/sensor  $i$  gathers a  $d$ -dimensional observation  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $i = 1, \dots, S$ . Our goal is to compute the pairwise distances  $\|\mathbf{x}_i - \mathbf{x}_j\|$  which are necessitated by clustering problems or dimensionality reduction with ISOMAP [20]. Let us observe that

$$\|\mathbf{x}_i - \mathbf{x}_j\| = \sqrt{\|\mathbf{x}_i\|^2 + \|\mathbf{x}_j\|^2 - 2\mathbf{x}_i^\top \mathbf{x}_j}. \quad (60)$$

Importantly, the squared norms and inner products in the above can be obtained from the following matrix:

$$\begin{aligned} \mathbf{A} &:= \mathbf{X} \mathbf{X}^\top \in \mathbb{R}^{S \times S}, \text{ where} \\ \mathbf{X} &:= (\mathbf{x}_1 \cdots \mathbf{x}_S)^\top \in \mathbb{R}^{S \times d}. \end{aligned} \quad (61)$$

We have  $\mathbf{x}_i^\top \mathbf{x}_j = A_{ij}$  and therefore the pairwise distance can be computed through  $\|\mathbf{x}_i - \mathbf{x}_j\| = \sqrt{A_{ii} + A_{jj} - 2A_{ij}}$ . To estimate the pairwise distances, we can therefore compute the rank- $p$  approximation of  $\mathbf{A}$

$$\hat{\mathbf{A}}_p = \sum_{r=1}^p \lambda_r \mathbf{u}_r^* (\mathbf{u}_r^*)^\top \approx \mathbf{A} \quad (62)$$

In particular,  $\hat{\mathbf{A}}_p$  is computed from the eigenvalue–vector pairs  $\{\lambda_r, \mathbf{u}_r^*\}_{r=1}^p$ , i.e., it is a PCA problem. We note that the data are structured in a DRO setting since each agent holds a row vector in  $\mathbf{X}$ . This is a static PCA problem as the data matrix does not change over time. Thus, DistPM is a suitable algorithm. Fig. 8 compares the true and estimated distance matrix using DistPM (DRO) on a sensor network with  $S = 1000$  sensors/agents, with the graph topology arranged according to a random geometric graph. The DistPM (DRO) finds the distance matrix accurately.

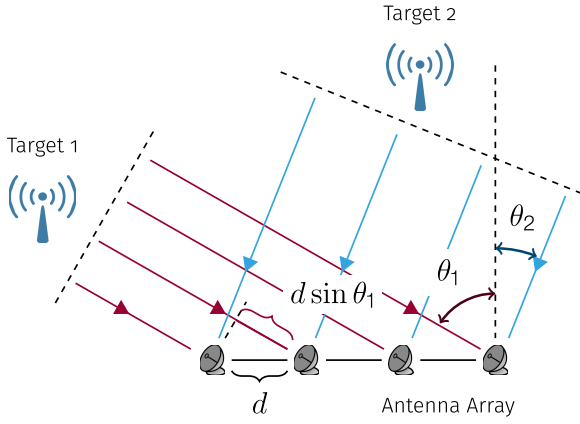
2) *Distributed Direction-of-Arrival Tracking:* We next illustrate how to track the direction-of-arrival (DoA) of multiple objects in a massive antenna array system using the distributed PCA methods. Consider a scenario where the (groups of) antennas are arranged in a uniform linear array (ULA) configuration, receiving signals from a far object. An example is depicted in Fig. 9. When there are  $p$  objects to be tracked, it is well known [1], [2] that the signal received at the antennas is

$$\mathbf{x}(t) = \sum_{q=1}^p \mathbf{a}(d, \theta_q) \mathbf{z}_q(t) + \mathbf{e}(t) \quad (63)$$

where

$$\mathbf{a}(d, \theta_q) := (1, e^{-j\omega_c d \sin \theta_q / \lambda}, \dots, e^{-j\omega_c (S-1)d \sin \theta_q / \lambda})^\top$$

such that  $\lambda$  is the wavelength and  $\omega_c$  is the carrier frequency,  $\mathbf{z}_q(t)$  is the signal transmitted from the  $q$ th object, and  $\mathbf{e}(t)$  is a white additive noise with variance  $\sigma^2$ . Importantly, the vectors  $\{\mathbf{a}(d, \theta_q)\}_{q=1}^p$  are mutually orthogonal



**Fig. 9. Direction-of-arrival estimation.** Our goal is to estimate the directions  $\theta_1, \theta_2, \dots$  of the targets in the figure from the signals received at the uniform linear array (ULA).

when the number of antennas is large, provided that  $\theta_q \neq \theta_{q'}$ , thus the  $p$ -D principal subspace of the signal  $x(t)$  is given by  $\mathbf{A} := (a(d, \theta_1) \cdots a(d, \theta_p))$  and the model fits into the one described by (2). The DoA for multiple objects can thus be found by extracting this principal subspace. This leads to the celebrated MUSIC/ESPRIT methods [2]. For instance, suppose that  $\hat{\mathbf{U}}$  is the estimated principal subspace of  $x(t)$ , the MUSIC method finds the  $p$  largest “peaks” in the following pseudospectrum:

$$P(\theta) = \frac{\mathbf{a}^H(d, \theta) \mathbf{a}(d, \theta)}{\mathbf{a}^H(d, \theta) \mathbf{U}_\perp \mathbf{U}_\perp^H \mathbf{a}(d, \theta)} \quad (64)$$

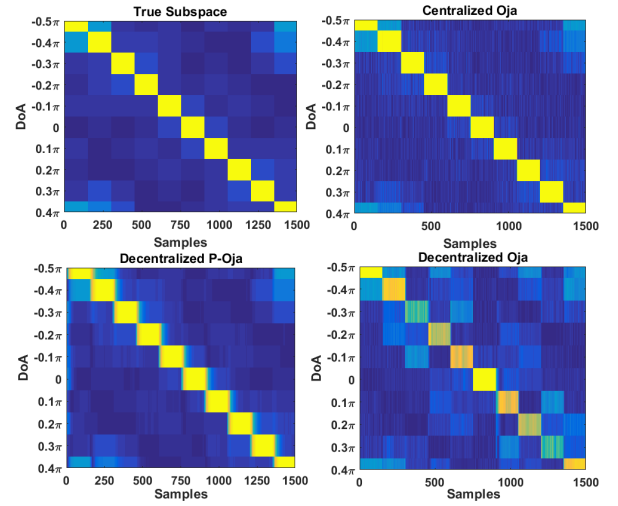
where  $\hat{\mathbf{U}}_\perp$  is an  $N \times (N - p)$  matrix containing  $N - p$  orthonormal vectors that are orthogonal to  $\hat{\mathbf{U}}$  such that  $\mathbf{U}_\perp^H \hat{\mathbf{U}} = \mathbf{0}$ .

In a distributed setting, each element of  $x(t)$  corresponds to a scalar signal received by an antenna operated by a sensor in the network. We have  $S = N$  such sensors/antennas. The problem of determining  $\hat{\mathbf{U}}$ , the  $p$ -D principal subspace of  $x(t)$ , is thus a dynamic PCA problem (4) with DRO data, as the object is moving.

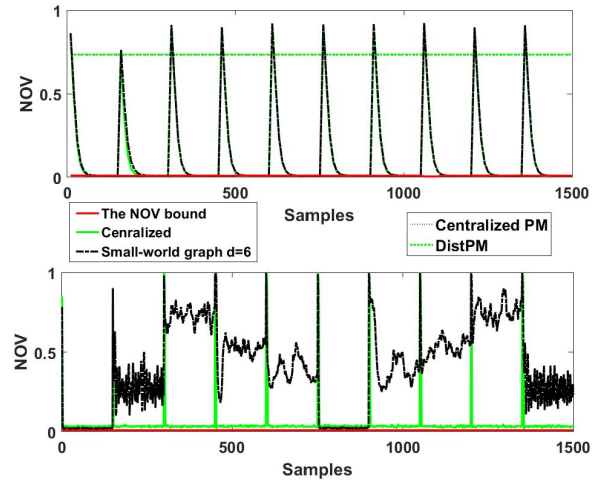
A simulation example of tracking the DoA of a single moving object using a ULA with  $S = N = 30$  antennas is shown below. In the example, the antennas/sensors are connected through a mean degree-4 small world graph with rewiring probability 0.2. Each of the antenna receives a total of  $T = 1500$  samples and the signal-to-noise ratio (SNR) in (63) is 20 dB. The heatmap for evolution of pseudospectrum and the evolution of normalized objective value (NOV) [cf., (45)], i.e.,

$$\text{NOV}(t) := \frac{\mathbb{E} \|x(t) - \mathbf{u}(t) \mathbf{u}^H(t) x(t)\|_2^2}{\mathbb{E} \|x(t)\|_2^2} \quad (65)$$

are shown in Figs. 10 and 11, respectively. Once again, we observe that the decentralized approach produces an accurate estimate of the DoA.

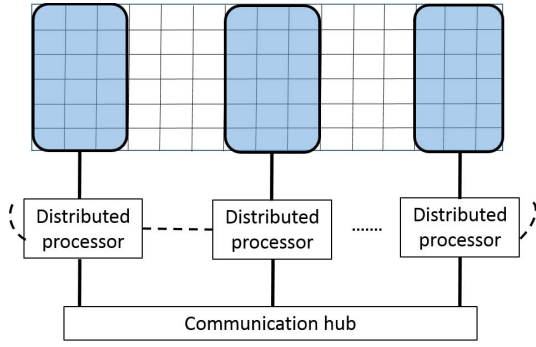


**Fig. 10. DoA tracking with distributed PCA methods.** The top-left plot shows the evolution of DoA against time/batch number, as indicated by the yellow dot. The remaining three plots show the evolution of pseudospectrum (64) using the Oja, D-Oja, and distributed P-Oja methods. Each column in the plot represents a pseudospectrum. The P-Oja method is set with the batch size of  $B=5$  with a step size of  $\gamma_t = 0.25$  while the Oja's method is set with a step size of  $\gamma_t = 0.05$ .

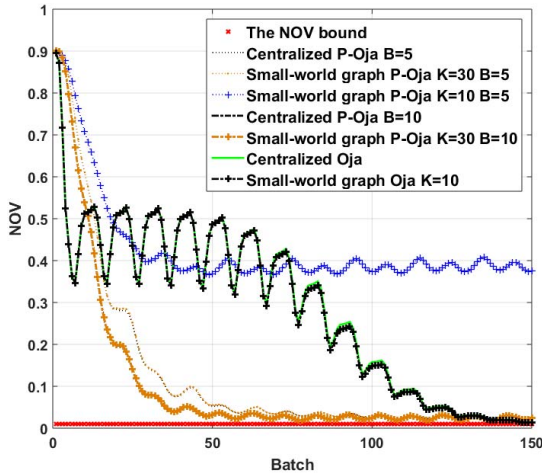


**Fig. 11. Evolution of the NOV against time in DoA tracking.** We apply the centralized and decentralized P-Oja and Oja methods for subspace tracking. The top figure is for the P-Oja method where the batch size is  $B=5$ , the power iteration parameter is set as  $P=20$ , and a step size  $\gamma_t = 0.25$ ; the bottom figure is the Oja method with a step size  $\gamma_t = 0.05$ . For both methods, the number of gossip round is  $L=10$ .

3) *Distributed Beamforming*: A related application is that of distributed PCA methods for beamforming, where the aim is to find the weights of the maximal ratio combiner, to combine the signal at each antenna element in such a way as to maximize the SNR. This problem fits into the setting of a dynamic PCA problem with DRO data as we deploy subarrays of antennas.



**Fig. 12.** Grouping antennas into subarrays with distributed processors. Each distributed processor can be treated as a supernode and communicate with the others.



**Fig. 13.** Evolution of NOV against time in distributed beamforming. For Oja's and P-Oja methods, we set the step size as  $\gamma_t = 5 \times 10^{-4}$  and  $\gamma_t = 0.01B$ , respectively, where  $B$  is the batch size used.

Our physical setup is illustrated in Fig. 12, where we consider grouping  $N = 256$  antennas into  $S = 64$  subarrays, each with  $N_i = 4$  antennas. The processor units form a mean degree-6 small world graph with rewiring probability of 0.2. Each processor unit receives  $T = 1500$  samples with an SNR of 20 dB. The signal  $x(t)$  is generated with  $p = 2$  dimensional principal subspace. In Fig. 13, we compare the NOV [cf., (65)] of running the different PCA methods in this example. The numerical results show that the P-Oja method converges faster than the Oja's method in the (quasi)-static setting, and the decentralized algorithms track the subspaces accurately.

## E. Other Methods and Summary

In addition to the surveyed methods above, we note that an ADMM-based algorithm has been developed in [24]. There, the authors proposed to relax (3) into a matrix factorization problem as

$$\min_{U \in \mathbb{C}^{N \times p}, Y \in \mathbb{C}^{p \times T}} \|X - UY\|_F^2 \quad (66)$$

**Table 1** Comparison of the Distributed PCA Methods (DRO). We Focus on the Computation and Communication Complexities per Iteration Under the Special Case of  $p = 1$ .  $T$  Is the Number of Samples,  $N_i$  Is the Number of Observations in Subblock  $i$ ,  $L$  Is the Number of Average Consensus Iterations,  $P$  Is the Number of Power Iterations, and  $B$  Is the Number of Samples in One Batch. Note That the Convergence Speed for D-Oja and P-Oja Is Only Observed Empirically to Be Sublinear. The Computation Complexity Is Counted at a per-Agent Level. For the P-Oja Method, the Batch Size Is Constant With  $B = |B_{k,s}|$  for All  $k, s$

Methods	Comp.	Commun.	Convergence
DistPM [18]	$\mathcal{O}(TN_i)$	$\mathcal{O}(TL)$	linear / geometric
D-Oja [23]	$\mathcal{O}(N_i)$	$\mathcal{O}(L)$	(emp.) sub-linear
P-Oja [25]	$\mathcal{O}(PBN_i)$	$\mathcal{O}(PBL)$	(emp.) sub-linear

where they have relaxed the constraint that  $Y = U^H X$ . The problem (66) is then written as

$$\begin{aligned} \min_{U_i \in \mathbb{C}^{N_i \times p}, Y_i \in \mathbb{C}^{p \times T}, \forall i} \sum_{i=1}^S \|X_i^r - U_i Y_i\|_F^2 \\ \text{s.t. } Y_i = Y_j, \quad \forall (i, j) \in \mathcal{E}. \end{aligned} \quad (67)$$

The optimization problem shown above is then tackled using an alternating minimization approach, in which the update of the coupled variables  $Y_i$  is handled by the celebrated ADMM method [47]. See also [27] for a related study.

To conclude this section, in Table 1, we summarize the PCA strategies described in the above for DRO data by comparing the computation and communication complexities required by each method. Note that we chose not to compare the ADMM approach in [24] as the latter involves multiple nested loops which hinders a fair comparison. As seen from the comparison, even though the DistPM enjoys the fastest convergence rate, its computation and communication costs are also  $T$  times larger than the D-Oja method. Note that typically we have  $T \gg 1$ . The computation and communication costs for D-Oja and P-Oja are independent of  $T$ , yet it is observed (empirically) that the two methods converge at sublinear rates only. Compared to D-Oja, the reformulated objective function (48) used in P-Oja gives an improved condition number and potentially reduces the number of iterations required to reach the same level of accuracy. This demonstrates a tradeoff between the computation/communication cost and performance.

## VI. RELATED AND OPEN PROBLEMS

In this section, first we survey a few related problems and the state of the art on how some of these problems have been solved using distributed computations. Then, we discuss open problems that are relevant to distributed PCA.

### A. Distributed Robust PCA

The robust PCA problem [48], [49] deals with scenarios when the observed data are contaminated with outlier



noise. In particular, we consider a modified observation model from (2)

$$\mathbf{x}(t) = \mathbf{U}_p \mathbf{z}(t) + \mathbf{c}(t) + \mathbf{e}(t) \quad (68)$$

where  $\mathbf{c}(t)$  represents the outlier noise, e.g., a sparse vector. Such model may be used to handle situations of faulty sensors or anomalies in the observed data. In particular, when  $\mathbf{c}(t)$  is sufficiently sparse and the subspace dimension is sufficiently low, it is possible to recover the subspace spanned by  $\mathbf{U}_p$  through solving a convex program [49], [50].

Recently, distributed methods have been developed for the robust PCA problem. For instance, in the DCO setting, Kopsinis *et al.* [51] proposed to tackle a modified problem of (13) with a distributed subgradient descent method. The authors therein essentially solve

$$\min_{\mathbf{U} \in \mathbb{C}^{N \times p}} \sum_{t=1}^T \|(\mathbf{I} - \mathbf{U}\mathbf{U}^H)\mathbf{x}(t)\|_1 \quad \text{s.t. } \mathbf{U}^H \mathbf{U} = \mathbf{I} \quad (69)$$

where we have replaced the Euclidean norm in (3) by the sparsity-inducing  $\ell_1$ -norm; Rahmani and Atia [52] proposed a detection method for samples that are contaminated by outlier noise and prune away those samples. Note that these methods are only compatible with the DCO setting. For the DRO case, Mardani *et al.* [53] proposed an ADMM algorithm for the corresponding robust PCA problem.

## B. Distributed Canonical Correlation Analysis

As a closely related problem to PCA, the canonical correlation analysis (CCA) problem [54] is widely applied in blind source separation, array processing, medical imaging, and word embedding; see [55]–[57] for the applications.

In a nutshell, the CCA problem involves simultaneously analyzing two data sets of paired data,  $\{\mathbf{x}(t)\}_{t=1}^T, \{\mathbf{y}(t)\}_{t=1}^T$ , where  $\mathbf{x}(t) \in \mathbb{C}^{N_1}$  and  $\mathbf{y}(t) \in \mathbb{C}^{N_2}$ , in order to find a low-rank structure in the corresponding cross-correlation matrix  $\mathbf{R}_{xy} := \mathbb{E}[\mathbf{x}(t)\mathbf{y}^H(t)]$ . Each of the two data sets offers a different view of the same latent structure, e.g.,  $\mathbf{x}(t)$  corresponds to the spelling features for document  $t$  while  $\mathbf{y}(t)$  corresponds to the contextual features. Mathematically, similar to the PCA problem with  $p = 1$ , the CCA problem can be given by

$$\max_{\mathbf{u}_1 \in \mathbb{C}^{N_1}, \mathbf{v}_1 \in \mathbb{C}^{N_2}} \frac{\mathbf{u}_1^H \mathbb{E}[\mathbf{x}(t)\mathbf{y}^H(t)] \mathbf{v}_1}{\sqrt{\mathbb{E}[\|\mathbf{u}_1^H \mathbf{x}(t)\|^2] \cdot \mathbb{E}[\|\mathbf{v}_1^H \mathbf{y}(t)\|^2]}}. \quad (70)$$

Efficient algorithms for CCA have been developed, e.g., [58]–[60]. However, to the best of our knowledge, the literature on the distributed CCA problem is lacking except for a recent work in [61]. Therein, Bertrand and Moonen considered a DRO setting and applied the idea of alternating optimization to tackle (70). On the down side, the proposed algorithm only works in a tree-network setting.

## C. Distributed Dictionary Learning

The dictionary learning problem [62]–[64] aims at learning an overcomplete dictionary which can describe the data set with a sparse linear combination of the atoms. Let  $\mathbf{X} \in \mathbb{R}^{N \times T}$  be the given data set, then the dictionary learning problem can be formulated as a matrix decomposition problem

$$\min_{\mathbf{D}, \mathbf{Y}} \|\mathbf{X} - \mathbf{D}\mathbf{Y}\|_F^2 \quad \text{s.t. } \|\mathbf{y}_t\|_0 \leq R_0, \quad t = 1, \dots, T \quad (71)$$

where the  $\ell_0$ -norm constraint on the columns of  $\mathbf{Y}$  and  $\mathbf{y}_t$  ensures that the problem finds a sparse representation, and we have  $\mathbf{D} \in \mathbb{R}^{N \times K}$ ,  $\mathbf{Y} \in \mathbb{R}^{K \times T}$  such that the dictionary consists of  $K$  atoms. Problem (71) is challenging since both of its objective function and constraint are nonconvex. As such, (71) is usually tackled by an alternating optimization procedure which iterates between two steps: a sparse coding step for updating  $\mathbf{Y}$  and a dictionary learning step for updating  $\mathbf{D}$  as well as the nonzero elements of  $\mathbf{Y}$ .

To tackle (71) distributively when the data are organized with the DCO structure [cf., (11)] one can follow the cloud  $K$ -SVD method in [43] to show how the techniques in distributed PCA can be applied. For the sparse coding step, it is obvious that the problem is decomposable due to the data structure. Let us denote the solution obtained at this step by  $\mathbf{Y}^{(s)}$ . For the dictionary learning step, Raja and Bajwa [43] applied a cyclical update to optimize each atom (column) of  $\mathbf{D}$  sequentially. For the  $k$ th atom, we consider

$$\begin{aligned} \min_{\mathbf{d}_k \in \mathbb{R}^N} \|\tilde{\mathbf{X}}^{(s)} - \mathbf{d}_k (\mathbf{y}_k^{(s), \text{row}})^T\|_F^2 \\ \iff \min_{\mathbf{d}_k \in \mathbb{R}^N} \|\tilde{\mathbf{X}}_R^{(s)} - \mathbf{d}_k (\mathbf{y}_{k,R}^{(s), \text{row}})^T\|_F^2 \end{aligned} \quad (72)$$

where  $\tilde{\mathbf{X}}^{(s)}$  is obtained by taking the difference between  $\mathbf{X}$  and the contributions from the atoms other than  $\mathbf{d}_k$ ,  $\mathbf{y}_k^{(s), \text{row}}$  is the  $k$ th row of  $\mathbf{Y}^{(s)}$ ,  $\mathbf{y}_{k,R}^{(s), \text{row}}$  is the subvector of  $\mathbf{y}_k^{(s), \text{row}}$  retaining only the nonzero components of the latter, and  $\tilde{\mathbf{X}}_R^{(s)}$  selects the columns of  $\tilde{\mathbf{X}}^{(s)}$  with respect to  $\mathbf{y}_{k,R}^{(s), \text{row}}$ .

Importantly, the minimization on the right-hand side of (72) can be solved by obtaining the top left and right singular vectors of  $\tilde{\mathbf{X}}_R^{(s)}$ . Note that this is precisely the distributed PCA problem with DCO data. As proposed in [43], a DistPM (DCO) method is then applied to compute the updated atom  $\mathbf{d}_k^*$  as well as updating the nonzero elements in  $\mathbf{y}_k^{(s), \text{row}}$ .

We remark that distributed dictionary learning has also been considered in a number of prior works, e.g., [65]–[68] with a different approach toward the solution.

## D. Distributed Low-Rank Optimization

Distributed PCA methods can also be applied as a subroutine to solve low-rank multiagent optimization problem

of the form

$$\min_{\theta \in \mathbb{R}^{m_1 \times m_2}} \sum_{i=1}^S f_i(\theta) \text{ s.t. } \|\theta\|_* \leq R \quad (73)$$

which includes problems such as matrix completion as special cases [69]. Note that  $\|\theta\|_*$  denotes the nuclear norm of the matrix  $\theta$ . Each of  $f_i(\theta)$  is the loss function attributed to the partition of data available at the  $i$ th agent/machine, which is assumed to be continuously differentiable but possibly nonconvex. Importantly, in the above, the observed data can be split in an arbitrary fashion.

For large-scale problems with  $m_1, m_2 \gg 1$ , a popular method is to apply the so-called projection-free (also known as Frank–Wolfe) algorithms [70], which amounts to finding the top singular vector of the gradient matrix at each iteration. Obviously, the top singular vector can be found using the PCA methods. Furthermore, as the gradient matrix is simply a sum of the local gradients, the associated PCA problem admits a DCO data structure. To this end, Wai et al. [44] applied the DistPM (DCO) with a set of carefully designed, time-varying parameters  $L, K$  as a subroutine to tackle (73) in a distributed fashion, and the authors showed that the resulting decentralized Frank–Wolfe algorithm [71] converges at desirable rates for both convex and nonconvex instances of (73). Similar effort can also be found in [72] for the master–slave architecture.

As an alternative, it is worthwhile mentioning that a heuristic solution to the matrix completion problem [69] [when the loss functions in (73) are squared Euclidean distances between entries of  $\theta$  and the observed data] can be found by iteratively computing the top singular vectors from a set of partially observed data, following the procedure proposed in [73]. Once again, this procedure can be applied to a distributed setting adopting the distributed PCA methods reviewed in this paper.

## E. Open Problems

A problem in which the literature is lacking is in tackling the PCA problem distributively with irregularly partitioned data. The data may conform to neither the DCO nor the DRO type exactly, but be divided in an arbitrary fashion. The resulting PCA problem would essentially require each agent to infer the missing entries simultaneously as it computes the principal components.

Another interesting extension of the distributed PCA algorithms would be the methods for solving the tensor PCA problem (or tensor decomposition in general) [74], which has been rarely studied in the distributed setting. Note that tensor models are effective in modeling high-order relationships between observations, e.g., [75]. Some prior work on parallel tensor decomposition is found in [76]–[78], however, in addition to restricting the types of networks to operate the distributed algorithms on, this work either requires the data to be stored centrally [77], [78], or the same set to be available at multiple agents, at least in part [76]. These settings are more restrictive than the distributed data setting considered in distributed PCA. A fully distributed tensor PCA method has yet to be developed.

Last, thanks to the recent advancements in first-order and nonconvex optimization methods, a number of new computation techniques have been applied to the PCA problem, e.g., [79]–[81]. These new methods offer theoretically proven improvements and can overcome certain limitations in fast convergence for the traditional power and Oja’s methods, which are the backbones for distributed PCA methods discussed in this paper. For instance, Allen-Zhu and Li [80] proposed an PCA algorithm with a convergence rate that is independent of the spectral gap  $\sigma_1(\hat{R}_x) - \sigma_2(\hat{R}_x)$ ; De Sa et al. [81] combined the technique of variance reduction with acceleration to yield better dependence of the convergence rate on the spectral gap. The methods above also demonstrate significant speedup in a centralized setting empirically, and it would be beneficial to explore the possibility of extending these benefits to the distributed PCA algorithms.

## VII. CONCLUSION

The distributed PCA methods are motivated by the increasing popularity of networked systems, including sensor and computer networks. In this paper, we have surveyed recent advancements in distributed PCA methods, where signal processing strategies have been applied independently depending on how the data are acquired in the network. Compared to centralized PCA, these methods can efficiently harness the computation and storage resources at the distributed agents, as confirmed by the theoretical and empirical analysis presented.  $\square$

## REFERENCES

- [1] H. Krim and M. Viberg, “Two decades of array signal processing research: The parametric approach,” *IEEE Signal Process. Mag.*, vol. 13, no. 4, pp. 67–94, Jul. 1996.
- [2] P. Stoica, *Introduction to Spectral Analysis*. Upper Saddle River, NJ, USA: Prentice-Hall, 1997.
- [3] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006.
- [4] K. Pearson, “On lines and planes of closest fit to systems of points in space,” *Philos. Mag.*, vol. 2, no. 6, pp. 559–572, 1901.
- [5] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 849–856.
- [6] C. Ding and X. He, “K-means clustering via principal component analysis,” in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, p. 29.
- [7] Y. Qu, G. Ostrouchov, N. Samatova, and A. Geist, “Principal component analysis for dimension reduction in massive distributed data sets,” in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Apr. 2002.
- [8] Y. Liang, M.-F. Balcan, V. Kanchanapally, and D. Woodruff, “Improved distributed principal component analysis,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3113–3121.
- [9] J. Fan, D. Wang, K. Wang, and Z. Zhu (2017). “Distributed estimation of principal eigenspaces.” [Online]. Available: <https://arxiv.org/abs/1702.06488>
- [10] R. Kannan, S. Vempala, and D. Woodruff, “Principal component analysis and higher correlations for distributed data,” in *Proc. Conf. Learn. Theory*, 2014, pp. 1040–1057.
- [11] C. Boutsidis, D. P. Woodruff, and P. Zhong, “Optimal principal component analysis in distributed and streaming models,” in *Proc. 48th Annu. ACM Symp. Theory Comput.*, 2016, pp. 236–249.
- [12] D. Garber, O. Shamir, and N. Srebro, “Communication-efficient algorithms for distributed stochastic principal component

- analysis," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1203–1212.
- [13] Z.-J. Bai, R. H. Chan, and F. T. Luk, "Principal component analysis for distributed data sets with updating," in *Proc. Int. Workshop Adv. Parallel Process. Technol.*, 2005, pp. 471–483.
- [14] H. Kargupta, W. Huang, K. Sivakumar, and E. Johnson, "Distributed clustering using collective principal component analysis," *Knowl. Inf. Syst.*, vol. 3, no. 4, pp. 422–448, 2001.
- [15] H. Qi, T.-W. Wang, and J. D. Birdwell, "Global principal component analysis for dimensionality reduction in distributed data mining," *Stat. Data Mining Knowl. Discovery*, pp. 327–342, 2004.
- [16] F. N. Abu-Khzam, N. F. Samatova, G. Ostrouchov, M. A. Langston, and A. Geist, "Distributed dimension reduction algorithms for widely dispersed data," in *IASTED PDSCS*, 2002, pp. 167–174.
- [17] J. Fellus, D. Picard, and P.-H. Gosselin, "Dimensionality reduction in decentralized networks by gossip aggregation of principal components analyzers," in *Proc. Eur. Symp. Artif. Neural Netw. Comput. Intell. Mach. Learn.*, 2014, pp. 171–176.
- [18] A. Scaglione, R. Pagliari, and H. Krim, "The decentralized estimation of the sample covariance," in *Proc. 42nd Asilomar Conf. Signals Syst. Comput.*, 2008, pp. 1722–1726.
- [19] Y.-A. Le Borgne, S. Raybaud, and G. Bontempi, "Distributed principal component analysis for wireless sensor networks," *Sensors*, vol. 8, no. 8, pp. 4821–4850, 2008.
- [20] M. E. Yildiz, F. Caramello, and A. Scaglione, "Distributed distance estimation for manifold learning and dimensionality reduction," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Apr. 2009, pp. 3353–3356.
- [21] W. Suleiman, M. Pesavento, and A. M. Zoubir, "Performance analysis of the decentralized eigendecomposition and ESPRIT algorithm," *IEEE Trans. Signal Process.*, vol. 64, no. 9, pp. 2375–2386, May 2016.
- [22] S. B. Korada, A. Montanari, and S. Oh, "Gossip PCA," in *Proc. ACM SIGMETRICS Joint Int. Conf. Meas. Modeling Comput. Syst.*, 2011, pp. 209–220.
- [23] L. Li, A. Scaglione, and J. H. Manton, "Distributed principal subspace estimation in wireless sensor networks," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 4, pp. 725–738, Aug. 2011.
- [24] I. D. Schizas and A. Aduroja, "A distributed framework for dimensionality reduction and denoising," *IEEE Trans. Signal Process.*, vol. 63, no. 23, pp. 6379–6394, Dec. 2015.
- [25] S. X. Wu, H.-T. Wai, A. Scaglione, and N. A. Jacklin, "The Power-Oja method for decentralized subspace estimation/tracking," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 3524–3528.
- [26] A. Wiesel and A. O. Hero, "Decomposable principal component analysis," *IEEE Trans. Signal Process.*, vol. 57, no. 11, pp. 4369–4377, Nov. 2009.
- [27] A. Bertrand and M. Moonen, "Distributed adaptive estimation of covariance matrix eigenvectors in wireless sensor networks with application to distributed PCA," *Signal Process.*, vol. 104, no. 2, pp. 120–135, Nov. 2014.
- [28] G. Morral, P. Bianchi, and J. Jakubowicz, "Asynchronous distributed principal component analysis using stochastic approximation," in *Proc. IEEE 51st Annu. Conf. Decision Control (CDC)*, Dec. 2012, pp. 1398–1403.
- [29] J. Fellus, D. Picard, and P.-H. Gosselin, "Asynchronous gossip principal components analysis," *Neurocomputing*, vol. 169, pp. 262–271, Dec. 2015.
- [30] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 2013.
- [31] E. Oja and J. Karhunen, "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix," *J. Math. Anal. Appl.*, vol. 106, no. 1, pp. 69–84, 1985.
- [32] J. N. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Dept. Electr. Eng. Comput. Sci., Massachusetts Inst. Technol., Boston, MA, USA, 1984.
- [33] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proc. IEEE*, vol. 98, no. 11, pp. 1847–1864, Nov. 2010.
- [34] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-100, no. 1, pp. 90–93, Jan. 1974.
- [35] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*. New York, NY, USA: Academic, 2014.
- [36] M. Uenohara and T. Kanade, "Optimal approximation of uniformly rotated images: Relationship between Karhunen-Loeve expansion and discrete cosine transform," *IEEE Trans. Image Process.*, vol. 7, no. 1, pp. 116–119, Jan. 1998.
- [37] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*, vol. 46. Hoboken, NJ, USA: Wiley, 2004.
- [38] A. Balsubramani, S. Dasgupta, and Y. Freund, "The fast convergence of incremental PCA," in *Proc. NIPS*, 2013, pp. 3174–3182.
- [39] S. Attallah and K. Abed-Meraim, "Fast algorithms for subspace tracking," *IEEE Signal Process. Lett.*, vol. 8, no. 7, pp. 203–206, Jul. 2001.
- [40] J. H. Manton, I. Y. Mareels, and S. Attallah, "An analysis of the fast subspace tracking algorithm NOja," in *Proc. IEEE Int. Conf. Acoust. Speech, Signal Process. (ICASSP)*, vol. 2, May 2002, pp. 1101–1104.
- [41] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.
- [42] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2508–2530, Jun. 2006.
- [43] H. Raja and W. U. Bajwa, "Cloud K-SVD: A collaborative dictionary learning algorithm for big, distributed data," *IEEE Trans. Signal Process.*, vol. 64, no. 1, pp. 173–188, Jan. 2016.
- [44] H.-T. Wai, A. Scaglione, J. Lafond, and E. Moulines, "Fast and privacy preserving distributed low-rank regression," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 4451–4455.
- [45] M. Hardt and E. Price, "The noisy power method: A meta algorithm with applications," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2861–2869.
- [46] J. H. Ward, Jr., "Hierarchical grouping to optimize an objective function," *J. Amer. Stat. Assoc.*, vol. 58, no. 301, pp. 236–244, 1963. [Online]. Available: <http://www.tandfonline.com/doi/abs/10.1080/01621459.1963.10500845>
- [47] 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.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.
- [48] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *J. ACM*, vol. 58, no. 3, p. 11, May 2011.
- [49] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, "Rank-sparsity incoherence for matrix decomposition," *SIAM J. Optim.*, vol. 21, no. 2, pp. 572–596, 2011.
- [50] A. Agarwal, S. Negahban, and M. J. Wainwright, "Noisy matrix decomposition via convex relaxation: Optimal rates in high dimensions," *Ann. Stat.*, vol. 40, no. 2, pp. 1171–1197, 2012.
- [51] Y. Kopsinis, S. Chouvardas, and S. Theodoridis, "Distributed robust subspace tracking," in *Proc. 23rd Eur. Signal Process. Conf. (EUSIPCO)*, Aug. 2015, pp. 2531–2535.
- [52] M. Rahmani and G. K. Atia, "A decentralized approach to robust subspace recovery," in *Proc. 53rd Annu. Allerton Conf. Commun. Control Comput. (Allerton)*, 2015, pp. 802–807.
- [53] M. Mardani, G. Mateos, and G. B. Giannakis, "Decentralized sparsity-regularized rank minimization: Algorithms and applications," *IEEE Trans. Signal Process.*, vol. 61, no. 21, pp. 5374–5388, Nov. 2013.
- [54] H. Hotelling, "Relations between two sets of variates," *Biometrika*, vol. 28, nos. 3–4, pp. 321–377, 1936.
- [55] N. M. Correa, T. Adali, Y.-O. Li, and V. D. Calhoun, "Canonical correlation analysis for data fusion and group inferences," *IEEE Signal Process. Mag.*, vol. 27, no. 4, pp. 39–50, Jun. 2010.
- [56] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor, "Canonical correlation analysis: An overview with application to learning methods," *Neural Comput.*, vol. 16, no. 12, pp. 2639–2664, 2004.
- [57] P. S. Dhillon, D. P. Foster, and L. H. Ungar, "Eigenwords: Spectral word embeddings," *J. Mach. Learn. Res.*, vol. 16, pp. 3035–3078, Dec. 2015.
- [58] H. Avron, C. Boutsidis, S. Toledo, and A. Zouzias, "Efficient dimensionality reduction for canonical correlation analysis," *SIAM J. Sci. Comput.*, vol. 36, no. 5, pp. S111–S131, 2014.
- [59] R. Ge, C. Jin, S. Kakade, P. Netrapalli, and A. Sidford, "Efficient algorithms for large-scale generalized eigenvector computation and canonical correlation analysis," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2741–2750.
- [60] X. Fu, K. Huang, M. Hong, N. D. Sidiropoulos, and A. M.-C. So, "Scalable and flexible multiview MAXVAR canonical correlation analysis," *IEEE Trans. Signal Process.*, vol. 65, no. 16, pp. 4150–4165, Aug. 2017.
- [61] A. Bertrand and M. Moonen, "Distributed canonical correlation analysis in wireless sensor networks with application to distributed blind source separation," *IEEE Trans. Signal Process.*, vol. 63, no. 18, pp. 4800–4813, Sep. 2015.
- [62] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.
- [63] Q. Zhang and B. Li, "Discriminative K-SVD for dictionary learning in face recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, Jun. 2010, pp. 2691–2698.
- [64] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 689–696.
- [65] P. Chainais and C. Richard, "Learning a common dictionary over a sensor network," in *Proc. IEEE 5th Int. Workshop Comput. Adv. Multi-Sensor Adaptive Process. (CAMSAP)*, Dec. 2013, pp. 133–136.
- [66] J. Chen, Z. J. Towfic, and A. H. Sayed, "Dictionary learning over distributed models," *IEEE Trans. Signal Process.*, vol. 63, no. 4, pp. 1001–1016, Feb. 2015.
- [67] H.-T. Wai, T.-H. Chang, and A. Scaglione, "A consensus-based decentralized algorithm for non-convex optimization with application to dictionary learning," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 3546–3550.
- [68] M. Hong, D. Hajinezhad, and M.-M. Zhao, "Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1529–1538.
- [69] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Found. Comput. Math.*, vol. 9, no. 6, p. 717, 2009.
- [70] M. Jaggi, "Revisiting Frank-Wolfe: Projection-free sparse convex optimization," in *Proc. ICML*, 2013, pp. 427–435.



- [71] H.-T. Wai, J. Lafond, A. Scaglione, and E. Moulines, "Decentralized Frank-Wolfe algorithm for convex and nonconvex problems," *IEEE Trans. Autom. Control*, vol. 62, no. 11, pp. 5522–5537, Nov. 2017.
- [72] W. Zheng, A. Bellet, and P. Gallinari (2017). "A distributed Frank-Wolfe framework for learning low-rank matrices with the trace norm." [Online]. Available: <https://arxiv.org/abs/1712.07495>
- [73] R. Mazumder, T. Hastie, and R. Tibshirani, "Spectral regularization algorithms for learning large incomplete matrices," *J. Mach. Learn. Res.*, vol. 11, pp. 2287–2322, Jan. 2010.
- [74] E. Richard and A. Montanari, "A statistical model for tensor PCA," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2897–2905.
- [75] A. Cichocki et al., "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Process. Mag.*, vol. 32, no. 2, pp. 145–163, Mar. 2015.
- [76] A. L. F. De Almeida and A. Y. Kibangou, "Distributed large-scale tensor decomposition," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, May 2014, pp. 26–30.
- [77] U. Kang, E. Papalexakis, A. Harpale, and C. Faloutsos, "GigaTensor: Scaling tensor analysis up by 100 times—Algorithms and discoveries," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 316–324.
- [78] A. P. Liavas and N. D. Sidiropoulos, "Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers," *IEEE Trans. Signal Process.*, vol. 63, no. 20, pp. 5450–5463, Oct. 2015.
- [79] O. Shamir, "A stochastic PCA and SVD algorithm with an exponential convergence rate," in *Proc. ICML*, 2015, pp. 144–152.
- [80] Z. Allen-Zhu and Y. Li, "LazySVD: Even faster SVD decomposition yet without agonizing pain," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 974–982.
- [81] C. De Sa, B. He, I. Mitliagkas, and C. Ré, and P. Xu (2017). "Accelerated stochastic power iteration." [Online]. Available: <https://arxiv.org/abs/1707.02670>

## ABOUT THE AUTHORS

**Sissi Xiaoxiao Wu** (Member, IEEE) received the B.Eng. degree in electronic information engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2005, the M.Phil. degree from the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, in 2009, and the Ph.D. degree in electronic engineering from the Chinese University of Hong Kong (CUHK), Hong Kong, in 2013.



From December 2013 to November 2015, she was a Postdoctoral Fellow in the Department of Systems Engineering and Engineering Management, CUHK. From December 2015 to March 2017, she was a Postdoctoral Fellow in the Signal, Information, Networks and Energy Laboratory supervised by Prof. A. Scaglione of Arizona State University, Tempe, AZ, USA. Since March 2017, she has been an Assistant Professor at the Department of Communication and Information Engineering, Shenzhen University, Shenzhen, China. Her research interests are in wireless communication theory, optimization theory, stochastic process, and channel coding theory, and with a recent emphasis on the modeling and data mining of opinion diffusion in social networks.

**Lin Li** (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from University of California, Davis, Davis, CA, USA, in 2013.

She is a member of the technical staff in the Human Language Technology (HLT) group at the Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, MA, USA. Before joining MIT Lincoln Laboratory in 2015, she was at the U.S. Army Research Laboratory, Adelphi, MD, USA, as a Postdoctoral Researcher, learning the sparse representations of high-dimensional data, and developing a modeling framework for learning the dynamic patterns in the collective behaviors of social agents. Her current research primarily focuses on advanced graph analytics, adaptation and learning over content and context graphs, and cross-domain tracking and attribution.

Dr. Li received the 2014 Young Author Best Paper Award from the IEEE Signal Processing Society for her work on distributed principal subspace estimation in wireless sensor networks.



**Anna Scaglione** (Fellow, IEEE) received the M.Sc. and Ph.D. degrees from Università di Roma "La Sapienza" in 1995 and 1999, respectively.

She is currently a Professor in Electrical and Computer Engineering at Arizona State University, Tempe, AZ, USA. She was a Professor of Electrical Engineering at the University of California Davis, Davis, CA, USA (2010–2014), an Associate Professor at the University of California Davis (2008–2010) and at Cornell University, Ithaca, NY, USA (2006–2008), and an Assistant Professor at Cornell University (2001–2006) and at the University of New Mexico, Albuquerque, NM, USA (2000–2001). Her expertise is in the broad area of statistical signal processing for communication, electric power systems, and network science. Her current research focuses on studying and enabling decentralized learning and signal processing in networks of sensors.

Dr. Scaglione served in the IEEE in many capacities, including as Associate Editor for the IEEE Transactions on Wireless Communications and the IEEE Transactions on Signal Processing and as the Editor-in-Chief of IEEE Signal Processing Letters. She was a member of the Signal Processing Society Board of Governors from 2011 to 2014. She is currently a Senior Associate Editor for the IEEE Transactions on Control over Networked Systems. She received the 2000 IEEE Signal Processing Transactions Best Paper Award and more recently was honored with the 2013 IEEE Donald G. Fink Prize Paper Award for the best review paper in that year in IEEE publications. Her work with her student earned 2013 IEEE Signal Processing Society Young Author Best Paper Award (with L. Li) and the IEEE ICASSP 2018 Best Conference Paper Award (with H.-T. Wai).



**Hoi-To Wai** (Member, IEEE) received the B.Eng. (with first class honors) and M.Phil. degrees in electronic engineering from The Chinese University of Hong Kong (CUHK), Hong Kong, in 2010 and 2012, respectively, and the Ph.D. degree in electrical engineering from Arizona State University (ASU), Tempe, AZ, USA, in 2017.



He is currently a Postdoctoral Scholar at ASU. He was a Ph.D. student at the University of California Davis, Davis, CA, USA; and a Visiting Scholar at Telecom ParisTech, Paris, France and at the Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA, USA. His research interests are in the broad area of signal processing, machine learning, and optimization theory.

Dr. Wai received the Best Student Paper Award from ICASSP 2018, and the 2017's Dean's Dissertation Award from the Ira A. Fulton Schools of Engineering of ASU.