The Best of Both Worlds: Distributed PCA That is Both Exact and Communication Efficient

Arpita Gang and Waheed U. Bajwa
Department of Electrical and Computer Engineering, Rutgers University–New Brunswick, NJ USA
Emails: arpita.gang@rutgers.edu, waheed.bajwa@rutgers.edu

***Abstract*—The effectiveness of machine learning algorithms largely depends on the *goodness* of the representation of data. While the massiveness in dimension and amount of modern day data requires dimension reduction and feature extraction for efficient use of available computational resources, the use of uncorrelated features is known to enhance the performance of such machine learning algorithms. Thus, an efficient representation learning approach should focus on dimension reduction as well as uncorrelated feature extraction. Even though Principal Component Analysis (PCA) and linear autoencoders are fundamental data processing tools largely used for dimension reduction, they can also be used to extract uncorrelated features when engineered properly. At the same time, factors like ever-increasing volume of data or inherently distributed data generation impede the use of existing centralized solutions for representation learning that require availability of data at a single location. This paper proposes two variants of an algorithm called *FAST-PCA (Fast and exAct diSTributed PCA)* based on a feedforward neural network-based system that learn data representations in a distributed setting such that they are reduced in dimension as well as have uncorrelated features. The proposed variants are meant to curb the communication overheads prevalent in the existing solutions and are shown to converge to the exact solutions at a linear rate. These claims are further supported by extensive numerical experiments.**

***Index Terms*—Exact convergence, Krasulina's method, Oja's rule, principal component analysis, representation learning**

I. Introduction

The modern world is data driven as massive, high-dimensional datasets are becoming an increasingly essential part of nearly every aspect of our lives ranging from healthcare to finance and from social media to the Internet-of-Things (IoT). In a related trend, machine learning algorithms are finding their applications in every possible domain because of their data-driven approaches and the ability to generalize to new unseen data. These algorithms require considerable amount of data preprocessing for their efficient and effective use. Some of the most used techniques for dimension reduction and feature learning are principal component analysis (PCA) [1], linear discriminant analysis (LDA), autoencoders [2], etc. The performance of machine learning methods is heavily dependent on the choice of data representation (or features) on which they are applied. Learning features from the data that embody the most important, explanatory and distinguishing information is an essential requirement of representation learning of the data.

This work was supported in part by the National Science Foundation under Awards CCF-1907658, and OAC-1940074, and by the Army Research Office under Awards W911NF-17-1-0546 and W911NF-21-1-0301.

It has been argued that one of the factors that makes a representation (of a data sample) "good" is having uncorrelated features in the learned representation [3]. This is because if the learned representations have uncorrelated features, changes or noise in one will not affect the others. Since PCA focuses on both the goals of dimension reduction as well as feature decorrelation, this paper focuses on PCA.

Another aspect of data now-a-days is it being inherently distributed geographically across locations in cases such as IoT, smart cities, autonomous vehicles, etc., where existing ways of representation learning are not directly applicable. Data tends to be distributed for a multitude of reasons; it can be inherently distributed (e.g., in IoT) where privacy or communication bottleneck prevents collation of data at a single location, or it can be distributed due to storage and/or computational limitations. Distributed setups can be largely classified into two types: i) those having a central entity/server that coordinates with the other nodes in a master-slave architecture, and ii) those lacking any central entity, in which the nodes are connected in an arbitrary network. Both these scenarios are prevalent in the world, but the latter case is more general as it encompasses all arbitrary network structures. The absence of a central server has further advantages like absence of single point of failure, no communication bottleneck at the central server, etc. Motivated by these reasons, this paper focuses on the problem of dimension reduction and uncorrelated representation learning in a distributed network with no central server.

A. Relation to Prior Work

PCA [1], [4] dates back to early 1900's and was proposed for decorrelating and compressing a set of data points by finding their principal components. Since then, many iterative methods like power method, orthogonal iterations [5], and Lanczos method [6] have been proposed to estimate eigenvectors or low-dimensional subspaces of symmetric matrices. In a different vein, a stochastic approximation algorithm was proposed by Krasulina in [7] for the estimation of the dominant eigenvector in the streaming data case. Another similar algorithm was later proposed by Oja [8] based on the Hebbian learning rule [9], which was then extended for multiple eigenvector estimation by Sanger [10]. Krasulina's method was also generalized for the estimation of a subspace of dimension greater than one in [11], although it only guarantees convergence to the principal subspace, instead of

principal components, under the low-rank matrix assumption. Although Oja's and Krasulina's methods are similar and there is no clear conclusion as to which is better, Oja's rule is more popular for the estimation of dominant eigenvector.

The problem of PCA in the distributed setting is relatively recent. A detailed review of various distributed PCA algorithms is done in [12]. Data can be distributed by either features or by samples, and in this paper we focus on the case of sample-wise data distribution, where each node estimates the entire basis and consensus in the network is a necessary condition. The sample-wise data distribution was considered in [13]–[15], where a power method-based approach was proposed for estimation of the dominant eigenvector ($K = 1$). Estimation of K eigenvectors would require running these algorithms K times. This dependence on K was done away with in an orthogonal iteration-based solution that was proposed in [16] for the general case of $K \geq 1$. Although this method estimates the K-dimensional subspace simultaneously, its convergence guarantees are in terms of subspace angles and thus it proves convergence to the principal subspace, making it a principal subspace analysis (PSA) method. Moreover, the aforementioned methods require an explicit consensus loop [17] in every iteration and the final error is a function of the number of consensus iterations, making them inefficient in terms of communication overhead. A recent one time-scale method for distributed optimization based on the Picard iteration was proposed in [18] and [19] demonstrated the application of this method to distributed PCA, but it could only prove local convergence, i.e., if the estimate is already "close enough" to the optimal solution, then it converges to it at a linear rate. A distributed algorithm for PCA based on generalized Hebbian algorithm using a combine-and-adapt strategy called distributed Sanger's algorithm (DSA) was developed and analyzed in our previous work [20]. Though it is a linearly convergent one-time scale algorithm, it only reaches to a neighborhood of the optimal solution for a fixed step size. To overcome the limitations of simple gradient descent-based distributed algorithms, new methods have been proposed recently that deploy a technique called "gradient-tracking" [21]–[23]. A recent paper on distributed PCA [24] used this gradient-tracking idea to develop a two-time scale algorithm for subspace estimation. In this paper, we use this gradient-tracking idea to develop a one-time scale algorithm that linearly converges to the eigenvectors of the covariance matrix, not just the subspace spanned by them.

B. Our Contributions

This paper introduces two variants of a novel algorithm for distributed PCA called *Fast and exAct diSTributed PCA* (FAST-PCA) based on Oja's and Krasulina's method. Theorems describing the convergence guarantees of both versions are presented along with experimental results that further demonstrate the efficiency of our solution for both synthetic and real-world datasets.

Notation: Scalars and vectors are denoted in the paper by lower-case and lower-case bold letters, respectively, while matrices are denoted by upper-case bold letters. The superscript in $\mathbf{a}^{(t)}$ denotes time (or iteration) index, $\|\cdot\|_F$ denotes the Frobenius-norm of matrices, while $\|\cdot\|$ denote the ℓ_2-norm of vectors.

II. Problem Description

For data samples $\mathbf{y} \in \mathbb{R}^d$ sampled from a zero-mean distribution with covariance matrix $\mathbf{\Sigma}$, PCA finds a low-dimensional subspace $\mathbf{X} \in \mathbb{R}^{d\times K}, d \gg K$ that encapsulates maximum information of the data while decorrelating the features $\tilde{\mathbf{y}} = \mathbf{X}^{\mathrm{T}}\mathbf{y} \in \mathbb{R}^K$, i.e., $\left(\mathbb{E}\left[\tilde{\mathbf{y}}\tilde{\mathbf{y}}^{\mathrm{T}}\right]\right)_{lq} = \left(\mathbb{E}\left[\mathbf{X}^{\mathrm{T}}\mathbf{y}\mathbf{y}^{\mathrm{T}}\mathbf{X}\right]\right)_{lq} = 0, \forall l \neq q$. It is evident that $\mathbb{E}\left[\mathbf{X}^{\mathrm{T}}\mathbf{y}\mathbf{y}^{\mathrm{T}}\mathbf{X}\right]$ will be a diagonal matrix if and only if $\mathbf{X}$ contains the eigenvectors of $\mathbf{\Sigma}$. Thus, the true solution of PCA is the eigenvectors of $\mathbf{\Sigma}$, not just any subspace spanned by them. In practice, however, the distribution of the samples and hence $\mathbf{\Sigma}$ is unknown, and is instead approximated using the samples. For a set of samples $\{\mathbf{y}_t\}_{t=1}^N$, the sample covariance matrix is given by $\mathbf{C} = \frac{1}{N-1}\sum_{t=1}^N (\mathbf{y}_t - \bar{\mathbf{y}})(\mathbf{y}_t - \bar{\mathbf{y}})^T$, where $\bar{\mathbf{y}} = \frac{1}{N}\sum_{t=1}^N \mathbf{y}_t$ is the sample mean. Henceforth, we shall assume $\bar{\mathbf{y}} = 0$ without loss of generality. The empirical formulation of the PCA problem in terms of samples is thus

$$\mathbf{X} = \underset{\mathbf{X}\in\mathbb{R}^{d\times K}, \mathbf{X}^{\mathrm{T}}\mathbf{X}=\mathbf{I}}{\arg\min} \sum_{t=1}^N \|\mathbf{y}_t - \mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{y}_t\|^2 \quad \text{such that} \quad \forall l \neq q, \left(\mathbf{X}^{\mathrm{T}}\left(\sum_{t=1}^N \mathbf{y}_t\mathbf{y}_t^{\mathrm{T}}\right)\mathbf{X}\right)_{lq} = 0. \tag{1}$$

In a distributed setting, the samples are unavailable at a single location. Consider a undirected and connected network of M nodes whose topology is described by a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{1, \ldots, M\}$ is the set of nodes and $\mathcal{E}$ is the set of edges between the nodes. In the sample-wise partitioned case considered in this paper, each node $i \in \{1, \ldots, M\}$ has access to n_i samples $\mathbf{Y}_i \in \mathbb{R}^{d\times n_i}$ such that $\sum_i n_i = n$, where n is the total number of samples in the network. Here, even though node i has access to a local covariance matrix $\mathbf{C}_i = \mathbf{Y}_i\mathbf{Y}_i^{\mathrm{T}}$, the goal is to find the eigenvectors of the global covariance matrix $\mathbf{C} = \sum_i \mathbf{C}_i$ at every node, with each node maintaining its own copy $\mathbf{X}_i$ of the variable $\mathbf{X}$. The goal of distributed PCA is for each node to eventually reach the same solution, i.e., achieve network consensus, given by the eigenvectors of $\mathbf{C}$. Thus, the actual PCA objective for the distributed case is

$$\underset{\mathbf{X}_i\in\mathbb{R}^{d\times K}, \mathbf{X}_i^{\mathrm{T}}\mathbf{X}_i=\mathbf{I}}{\arg\min} \sum_{i=1}^M \|\mathbf{Y}_i - \mathbf{X}_i\mathbf{X}_i^{\mathrm{T}}\mathbf{Y}_i\|_F^2 \quad \text{such that}$$
$$\forall j \in \mathcal{N}_i, \mathbf{X}_i = \mathbf{X}_j \quad \text{and} \quad \forall l \neq q, \left(\mathbf{X}_i^{\mathrm{T}}\left(\sum_{i=1}^M \mathbf{Y}_i\mathbf{Y}_i^{\mathrm{T}}\right)\mathbf{X}_i\right)_{lq} = 0. \tag{2}$$

Here, the objective functions $\arg\min \|\mathbf{Y}_i - \mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{Y}_i\|_F^2$ are local to the node i, but the constraint is global, that is, shared among all the nodes. This major difference between the formulations (1) and (2) makes it impossible for the centralized PCA solutions to be directly applicable to the distributed case.

To satisfy the requirement for all nodes to reach a common solution, which is the eigenvectors of the global covariance matrix $\mathbf{C}$, without sharing the raw local data $\mathbf{Y}_i$, we need some collaboration between the nodes.

The constraint in (2) makes the problem nonconvex since the solution lies on the Stiefel manifold and particularly, it is a specific element of the manifold. Convexification of the problem like in [25] will result in $\mathcal{O}(d^2)$ computational and memory cost since it approximates the projection matrix of the $d \times K$ dimensional subspace, which is restrictive in the case of high dimensional data. Also, such convexification relaxes the problem to only find the subspace spanned by the eigenvectors. Since neither of these limitations are desirable for us, we choose algebraic methods that can lead to the true solution in a computationally efficient manner. In [20], we showed that a simple method based on the generalized Hebbian rule [10] leads to the inexact convergence, i.e., the solution only reaches to a neighborhood of the true eigenvectors of $\mathbf{C}$. In this paper, we ask the question: is it possible to have both fast and exact convergence for distributed PCA? The short answer is: Yes! We propose a solution based on the gradient-tracking technique [21]–[23] that converges exactly and at a linear rate to the true eigenvectors of the global covariance matrix $\mathbf{C}$ in the distributed setting.

III. Proposed Algorithm: FAST-PCA

Iterative methods like power method [5] have proven to be very effective solutions for PCA because of their effectiveness and ease of implementations. For the case of estimating the top eigenvector ($K = 1$) of data covariance matrix in centralized setting for the streaming data case, two elegant, similar and widely studied algorithms were proposed by Oja [8] and Krasulina [7] whose update equations go as follows:

$$\textbf{Oja:} \quad \mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \alpha_t \big(\mathbf{C}_t \mathbf{x}^{(t)} - (\mathbf{x}^{(t)})^{\mathrm{T}} \mathbf{C}_t \mathbf{x}^{(t)} \mathbf{x}^{(t)}\big)$$

$$\textbf{Krasulina:} \quad \mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \alpha_t \big(\mathbf{C}_t \mathbf{x}^{(t)} - \frac{(\mathbf{x}^{(t)})^{\mathrm{T}} \mathbf{C}_t \mathbf{x}^{(t)}}{\|\mathbf{x}^{(t)}\|^2} \mathbf{x}^{(t)}\big),$$

where $\mathbf{C}_t = \mathbf{y}_t \mathbf{y}_t^{\mathrm{T}}$ is the sample covariance matrix and α_t is the step size at time t. Both these methods were shown to converge to the dominant eigenvector of $\mathbf{\Sigma} = \mathbb{E}\left[\mathbf{C}_t\right]$ under the requirement that $\sum_t \alpha_t^2 \to 0$. From an autoencoder training point of view, the simple update based rules can be very easily implemented in neural networks where the $\mathbf{x}'s$ denote the weights of the network. The resultant "encoding" or representation learned from such network will have uncorrelated features. Oja's rule was generalized for the case of multiple eigenvector estimation ($K > 1$) by Sanger [10] using the generalized Hebbian rule, whereas Krasulina's method was extended for the estimation of a higher dimensional ($K > 1$) subspace spanned by the eigenvectors of $\mathbf{\Sigma}$ [11].

In the distributed setting considered in this paper, samples are not streaming but split across nodes and $\mathbf{C} = \sum_i \mathbf{C}_i$ where $\mathbf{C}_i$ is the sample covariance matrix at node i. As mentioned before, the data is fixed at each node and the goal is to find the top K eigenvectors of $\mathbf{C}$ at each node i.e., have consensus in the network, and thus naive implementation of generalized Hebbian rule (GHA) or Krasulina's method would not accomplish our goal. Instead, we propose a gradient-tracking [22], [23] based solution called *Fast and exAct diSTributed PCA* (FAST-PCA) that converges exactly and at a linear rate to the top K eigenvectors of the global covariance matrix $\mathbf{C}$. We propose two variants of FAST-PCA based on the Hebbian (Oja's) and Krasulina's rule and call them FAST-PCA-O and FAST-PCA-K, respectively.

Let $\mathbf{x}_{i,k}^{(t)}$ be the estimate of the $k^{th}, k = 1, \ldots, K$, eigenvector at node i after t iterations of the algorithm. Then we define two pseudo gradients $\mathbf{h}_i^O(\mathbf{x}_{i,k}^{(t)})$ and $\mathbf{h}_i^K(\mathbf{x}_{i,k}^{(t)})$ motivated, respectively, from the Oja's and Krasulina's rule as follows:

$$\mathbf{h}_i^O(\mathbf{x}_{i,k}^{(t)}) = \mathbf{C}_i \mathbf{x}_{i,k}^{(t)} - (\mathbf{x}_{i,k}^{(t)})^{\mathrm{T}} \mathbf{C}_i \mathbf{x}_{i,k}^{(t)} \mathbf{x}_{i,k}^{(t)} - \sum_{p=1}^{k-1} (\mathbf{x}_{i,p}^{(t)})^{\mathrm{T}} \mathbf{C}_i \mathbf{x}_{i,k}^{(t)} \mathbf{x}_{i,p}^{(t)}$$

$$\mathbf{h}_i^K(\mathbf{x}_{i,k}^{(t)}) = \mathbf{C}_i \mathbf{x}_{i,k}^{(t)} - \frac{(\mathbf{x}_{i,k}^{(t)})^{\mathrm{T}} \mathbf{C}_i \mathbf{x}_{i,k}^{(t)}}{\|\mathbf{x}_{i,k}^{(t)}\|^2} \mathbf{x}_{i,k}^{(t)} - \sum_{p=1}^{k-1} \frac{(\mathbf{x}_{i,p}^{(t)})^{\mathrm{T}} \mathbf{C}_i \mathbf{x}_{i,k}^{(t)}}{\|\mathbf{x}_{i,p}^{(t)}\|^2} \mathbf{x}_{i,p}^{(t)},$$

Let $\mathbf{X}_i^{(t)} = \left[\mathbf{x}_{i,1}^{(t)}, \ldots, \mathbf{x}_{i,K}^{(t)}\right] \in \mathbb{R}^{d \times K}$ be the estimate of the K eigenvectors of the global covariance matrix $\mathbf{C}$ after t iterations. Along with $\mathbf{X}_i^{(t)}$, FAST-PCA also updates a second variable in every iteration that essentially tracks the average of the pseudo-gradients at the nodes. Let us define a *pseudo-gradient tracker* matrix $\mathbf{S}_i^{(t)} = \left[\mathbf{s}_{i,1}^{(t)}, \ldots, \mathbf{s}_{i,K}^{(t)}\right] \in \mathbb{R}^{d \times K}$ that tracks the average of the pseudo-gradients at each node. These $\mathbf{S}_i^{(t)}$ are updated along with the eigenvector estimates $\mathbf{X}_i^{(t)}$ in each iteration of FAST-PCA. The entities $\mathbf{h}_i^O(\mathbf{X}_i^{(t)})$ and $\mathbf{h}_i^K(\mathbf{X}_i^{(t)})$ in the algorithm are the matrices of the psuedo-gradients, i.e., $\mathbf{h}_i^{O/K}(\mathbf{X}_i^{(t)}) = \left[\mathbf{h}_i^{O/K}(\mathbf{x}_{i,1}^{(t)}), \ldots, \mathbf{h}_i^{O/K}(\mathbf{x}_{i,K}^{(t)})\right] \in \mathbb{R}^{d \times K}$. Both versions of the algorithm, namely FAST-PCA-O and FAST-PCA-K, involve same steps, except the difference in the pseudo-gradients as explained earlier and are formally described in Algorithm 1. The weight matrix $\mathbf{W} = [w_{ij}]$ is a doubly stochastic matrix that conforms to the underlying graph topology [26], i.e., $w_{ij} \neq 0$ if $(i,j) \in \mathcal{E}$ or $i = j$ and 0 otherwise. A necessary assumption for convergence of the algorithm here is the graph connectivity. In the next section, we provide convergence results of both versions of the proposed algorithm FAST-PCA.

Algorithm 1 Fast and exAct diSTributed PCA (FAST-PCA)

Input: $\mathbf{Y}_1, \mathbf{Y}_2, \ldots \mathbf{Y}_M, \mathbf{W}, \alpha, K$

Initialize: $\forall i, \mathbf{X}_i^{(0)} \leftarrow \mathbf{X}_{\text{init}} : \mathbf{X}_{\text{init}} \in \mathbb{R}^{d \times K}, \mathbf{X}_{\text{init}}^T \mathbf{X}_{\text{init}} = \mathbf{I}$; $\mathbf{S}_i^{(0)} \leftarrow \mathbf{h}_i(\mathbf{X}_i^{(0)})$

for $t = 0, 1, \ldots$ **do**

Communicate $\mathbf{X}_i^{(t)}$ from each node i to its neighbors

Subspace estimate at node i: $\mathbf{X}_i^{(t+1)} \leftarrow \frac{1}{2}\mathbf{X}_i^{(t)} + \sum_{j \in \mathcal{N}_i} \frac{w_{ij}}{2} \mathbf{X}_j^{(t)} + \alpha \mathbf{S}_i^{(t)}$

Psuedo-gradient estimate at node i: $\mathbf{S}_i^{(t+1)} \leftarrow \frac{1}{2}\mathbf{S}_i^{(t)} + \sum_{j \in \mathcal{N}_i} \frac{w_{ij}}{2} \mathbf{S}_j^{(t)} + \mathbf{h}_i^{O/K}(\mathbf{X}_i^{(t+1)}) - \mathbf{h}_i^{O/K}(\mathbf{X}_i^{(t)})$

end for

Return: $\mathbf{X}_i^{(t+1)}, i = 1, 2, \ldots, M$

TABLE I: Comparison of Communication and Iteration Cost

	Comm./Iteration	No. of Iterations	PCA/PSA
DistSeqPM	$\mathcal{O}(\frac{K}{\log gap_r^{-1}} \log \frac{1}{\epsilon})$	$\mathcal{O}(\frac{K}{\log gap_r^{-1}} \log \frac{1}{\epsilon})$	PCA
S-DOT	$\mathcal{O}(\frac{1}{\log gap_r^{-1}} \log \frac{1}{\epsilon})$	$\mathcal{O}(\frac{1}{\log gap_r^{-1}} \log \frac{1}{\epsilon})$	PSA
DeEPCA	$\mathcal{O}(\log \frac{1}{gap})$	$\mathcal{O}(\frac{1}{gap} \log \frac{1}{\epsilon})$	PSA
DSA	1	$\mathcal{O}(\frac{1}{\log(1+gap)} \log \frac{1}{\epsilon})$ up to $\epsilon = \mathcal{O}(\alpha)$	PCA
FAST-PCA	2	$\mathcal{O}(\frac{1}{\log(1+gap)} \log \frac{1}{\epsilon})$	PCA

IV. Convergence Guarantees

In this section we provide the convergence results of both versions of the FAST-PCA algorithm. Proof of convergence of FAST-PCA-K can be found in [27], while proof for FAST-PCA-O follows in a similar way with some minor changes.

Theorem 1. *Suppose the estimate $\mathbf{x}_{i,k}^{(t)}$ from FAST-PCA-O remains bounded i.e., $\|\mathbf{x}_{i,k}^{(t)}\| \leq \mu$, $\alpha < \frac{\min_{k=1,\ldots,K}(\lambda_k - \lambda_{k+1})}{(3\mu+K)(3\mu+K-1)}(\frac{1-\beta}{9\lambda_1})^2$ where λ_k, λ_{k+1} are the k^{th} and $(k+1)^{th}$ largest eigenvalues of $\mathbf{C}$ and $\beta = \max\{|\lambda_2(\mathbf{W})|, |\lambda_M(\mathbf{W})|\}$, $\mathbf{q}_k^T \mathbf{x}_{i,k}^{(0)} \neq 0$, and the graph underlying the network is connected. Then the estimate $\mathbf{x}_{i,k}^{(t)}$ from FAST-PCA-O converges to the eigenvector $\pm\mathbf{q}_k$ corresponding to the largest eigenvalue λ_k of $\mathbf{C}$ at each node $i = 1, \ldots, M$ at a linear rate.*

Theorem 1 shows that for certain choices of the step size α and under some assumptions, the estimates given by the FAST-PCA-O algorithm converge linearly to the top K eigenvectors of the global covariance matrix $\mathbf{C}$.

Theorem 2. *Suppose $\alpha < \frac{\min_{k=1,\ldots,K}(\lambda_k - \lambda_{k+1})}{(K+5)(K+6)}(\frac{1-\beta}{9\lambda_1})^2$, $\mathbf{q}_k^T \mathbf{x}_{i,k}^{(0)} \neq 0$, and the graph underlying the network is connected. Then the estimate $\mathbf{x}_{i,k}^{(t)}$ from FAST-PCA-K converges to a multiple of the eigenvector $\pm\mathbf{q}_k$ i.e., to $\pm c_k \mathbf{q}_k$, corresponding to the largest eigenvalue λ_k of $\mathbf{C}$ at each node $i = 1, \ldots, M$ at a linear rate.*

Theorem 2 shows that for certain choices of the step size α and under some milder assumptions, the estimates given by the FAST-PCA-K algorithm converge linearly to a multiple of the top K eigenvectors of the global covariance matrix $\mathbf{C}$. Note that a simple normalization step at the end of the algorithm would give us unit norm eigenvectors of $\mathbf{C}$.

In summary, both variants of FAST-PCA converge exactly to the true eigenvectors whilst completely doing away with the need of explicit consensus loop thereby making our solutions faster. Table I provides a comparison of the communication and iteration complexities of various distributed PCA algorithms in terms of error ϵ and eigengap gap. Here $gap = \lambda_K - \lambda_{K+1}$ for PSA algorithms and $gap = \min_{k=1,\ldots,K} \lambda_k - \lambda_{k+1}$ for PCA algorithms. Since we reduce the dependence of total iteration complexity on gap, our solutions are significantly faster than other algorithms as also shown through numerical experiments in the next section.

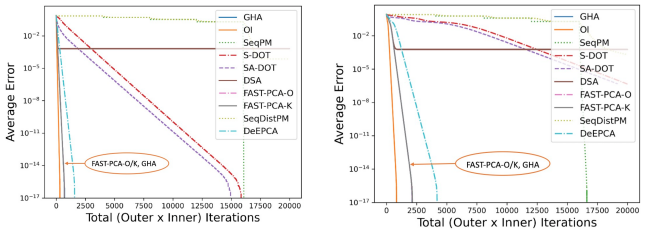
V. Experimental Results

In this section, we compare the performance of the two algorithms with existing algorithms of (centralized) orthogonal iteration (OI), (centralized) generalized Hebbian algorithm (GHA), (centralized) sequential power method (SeqPM), distributed sequential power method (SeqDistPM), distributed orthogonal iteration algorithms (S-DOT, SA-DOT) [16], an orthogonal iteration+gradient tracking-based method DeEPCA [24] and our previously proposed distributed Sanger's algorithm (DSA) [20]. In the case of OI and SeqPM, we assume all the samples are available at a single location and, for the estimation of K dominant eigenvectors of $\mathbf{C}$, SeqPM performs power method K times sequentially, starting from the most dominant eigenvector. SeqDistPM is the distributed version of SeqPM, which uses an explicit consensus loop with a fixed number T_c of consensus iterations per iteration of the power iteration [13], [14], whereas S-DOT and SA-DOT are distributed versions of OI using fixed and increasing number of consensus iterations per orthogonal iteration. The DSA is a distributed generalized Hebbian algorithm that converges linearly to a neighborhood of the true eigenvectors of the global covariance matrix. The x-axes of all the plots indicate the total iteration cost, i.e., total inner and outer loop iterations. In the algorithms with one time scale, this is the same as the number of total outer loop iterations (since inner iterations = 0). The y-axes of the plots is the average angle between the estimated eigenvectors $\mathbf{x}_{i,k}^{(t)}$ and the true eigenvectors $\pm\mathbf{q}_k$ across all the M nodes in the network given by

$$\mathcal{E} = \frac{1}{MK} \sum_{i=1}^{M} \sum_{k=1}^{K} \left(1 - \left(\frac{\mathbf{x}_{i,k}^T \mathbf{q}_k}{\|\mathbf{x}_{i,k}\|}\right)^2\right). \tag{3}$$

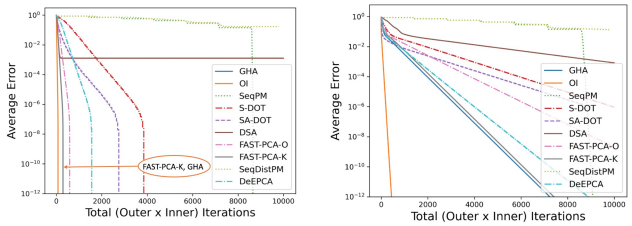
Synthetic Data: We generate Erdos-Renyi graphs ($p = 0.5$) and cyclic graphs to simulate the distributed setup with $M = 20$ nodes. The synthetic data is generated with different eigengaps of $\Delta_K = \frac{\lambda_{K+1}}{\lambda_K} \in \{0.8, 0.97\}$ such that each node has 5000 i.i.d samples, i.e. $N_i = 5000$ with $d = 20$ drawn from a multivariate Gaussian distribution with zero mean and fixed covariance matrix $\mathbf{\Sigma}$. The number of eigenvectors to be estimated is set to $K = 5$. For SeqPM, SeqDistPM and S-DOT, the number of consensus iterations per outer loop iteration is $T_c = 50$ and the number of maximum consensus iterations in the case of SA-DOT is set to 50 as well.

Figure 1 compares the performance of our proposed FAST-PCA algorithm with centralized OI, SeqPM, SeqDistPM, S-DOT, SA-DOT, DeEPCA and DSA. In it, the plots for FAST-PCA-O, FAST-PCA-K and centralized GHA are completely overlapping. This means that when measured in terms of angles, the convergence of our algorithms are same and also same as that of the centralized case, which in turn implies that the performance of Oja's and Krasulina's is same in the distributed setup when using a gradient-tracking technique. Evidently, our algorithms significantly outperform SeqPM, SeqDistPM, S-DOT and SA-DOT since the estimation of one eigenvector at a time and/or using explicit consensus loop slows down



(a) $d = 20, \Delta_K = 0.8$ (b) $d = 20, \Delta_K = 0.97$

Fig. 1: Performance comparison of FAST-PCA with various algorithms for two different eigengaps.



(a) MNIST, $K = 7$ (b) CIFAR10, $K = 7$

Fig. 2: Performance comparison of FAST-PCA with various algorithms for MNIST and CIFAR10

the convergence of these methods. As expected, since DSA converges only to a neighborhood of the true solutions, our new proposed algorithm outperforms it. DeEPCA uses the gradient-tracking technique on orthogonal iterations (OI) with the use of an explicit consensus loop. The dependence of the number of consensus iterations on the eigengap makes it impossible for its performance to match centralized OI.

Real-World Data: We also provide some results for the real-world datasets of MNIST [28] and CIFAR10 [29]. We simulate the distributed setup with an Erdos-Renyi graph with $p = 0.5$ and $M = 20$ nodes. Both these datasets have $N = 60,000$ samples distributed equally among the nodes, making $N_i = 3000$. The data dimensions are $d = 784$ for MNIST and $d = 1024$ for CIFAR10 and $K = 7$ was used for both. Figure 2 shows the comparison of the various PCA algorithms for MNIST ($\alpha = 0.05$) and CIFAR10 dataset ($\alpha = 0.8$) respectively. Due to space constraints, we have included limited experiments here. For more experiments including on heterogenous data, please refer to the thesis [30].

VI. Conclusion

In this paper, we proposed two variants of a novel algorithm for distributed Principal Component Analysis (PCA) called FAST-PCA (Fast and exAct diSTributed PCA) that converge to the true eigenvectors of the sample covariance matrix when data is distributed across a network. We provided theoretical results that show both versions of FAST-PCA converge linearly, exactly and globally. We also provided experimental results that further validate our claims.

References

[1] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *J. Educational Psychology*, pp. 417–441, 1933.

[2] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Netw.*, vol. 2, no. 1, p. 53–58, Jan. 1989.

[3] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, p. 1798—1828, August 2013.

[4] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Mag.*, vol. 2, pp. 559–572, 1901.

[5] G. H. Golub and C. F. Van Loan, *Matrix Computations (3rd Ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[6] C. Lanczos, "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators," *J. Research Nat. Bureau Standards*, 1950.

[7] T. P. Krasulina, "Method of stochastic approximation in the determination of the largest eigenvalue of the mathematical expectation of random matrices," *Autom. Remote Control*, vol. 1970, pp. 215–221, 1970.

[8] E. Oja and J. Karhunen, "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix," *J. Math. Anal. Applicat.*, vol. 106, no. 1, pp. 69 – 84, 1985.

[9] D. O. Hebb, *The Organization of Behavior : A Neuropsychological Theory*. Wiley New York, 1949.

[10] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," *Neural Netw.*, pp. 459 – 473, 1989.

[11] C. Tang, "Exponentially convergent stochastic k-PCA without variance reduction," in *Proc. NeurIPS*, 2019.

[12] S. X. Wu, H.-T. Wai, L. Li, and A. Scaglione, "A review of distributed algorithms for principal component analysis," *Proc. IEEE*, vol. 106, no. 8, pp. 1321–1340, 2018.

[13] H. Raja and W. U. Bajwa, "Cloud K-SVD: Computing data-adaptive representations in the cloud," in *Proc. Allerton Conf.*, 2013.

[14] 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.

[15] H. Wai, A. Scaglione, J. Lafond, and E. Moulines, "Fast and privacy preserving distributed low-rank regression," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Process., (ICASSP)*, 2017, pp. 4451–4455.

[16] A. Gang, B. Xiang, and W. U. Bajwa, "Distributed principal subspace analysis for partitioned big data: Algorithms, analysis, and implementation," *IEEE Trans. Signal Inform. Process. Netw.*, pp. 699–715, 2021.

[17] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[18] F. L. Andrade, M. A. Figueiredo, and J. Xavier, "Distributed Picard iteration," *arXiv preprint arXiv:2104.00131*, 2021.

[19] ——, "Distributed Picard iteration: Application to distributed EM and distributed PCA," *arXiv preprint arXiv:2106.10665*, 2021.

[20] A. Gang and W. U. Bajwa, "A linearly convergent algorithm for distributed principal component analysis," *Signal Process.*, vol. 193, p. 108408, 2022.

[21] W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: an exact first-order algorithm for decentralized consensus optimization," *SIAM J. Optim.*, vol. 25, no. 2, pp. 944–966, 2015.

[22] P. D. Lorenzo and G. Scutari, "NEXT: In-network nonconvex optimization," *IEEE Trans. Signal Inform. Process. Netw.*, pp. 120–136, 2016.

[23] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Trans. Control Netw. Syst.*, pp. 1245–1260, 2018.

[24] H. Ye and T. Zhang, "DeEPCA: Decentralized exact PCA with linear convergence rate," *J. Mach. Learning Research*, pp. 1–27, 2021.

[25] R. Arora, A. Cotter, and N. Srebro, "Stochastic optimization of PCA with capped MSG," in *Adv. Neural Inform. Process. Syst.*, vol. 26, 2013.

[26] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM REVIEW*, vol. 46, pp. 667–689, 2003.

[27] A. Gang and W. U. Bajwa, "FAST-PCA: A fast and exact algorithm for distributed principal component analysis," *arXiv preprint arXiv:2108.12373*, 2021.

[28] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," *ATT Labs*, vol. 2, 2010.

[29] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[30] A. Gang, "Representation learning in distributed networks," Ph.D. dissertation, Rutgers University-New Brunswick, 2022.