C-DIEGO: An Algorithm with Near-Optimal Sample Complexity for Distributed, Streaming PCA

Muhammad Zulqarnain, Arpita Gang, Waheed U. Bajwa
Department of Electrical and Computer Engineering
Rutgers University, New Brunswick, NJ 08854, USA
m.zulqarnain@rutgers.edu, arpita.gang@rutgers.edu, waheed.bajwa@rutgers.edu

***Abstract*—The accuracy of many downstream machine learning algorithms is tied to the training data having uncorrelated features. With the modern-day data often being streaming in nature, geographically distributed, and having large dimensions, it is paramount to apply both uncorrelated feature learning and dimensionality reduction techniques in this scenario. Principal Component Analysis (PCA) is a state-of-the-art tool that simultaneously yields uncorrelated features and reduces data dimensions by projecting data onto the eigenvectors of the population covariance matrix. This paper introduces a novel algorithm called *Consensus-DIstributEd Generalized Oja (C-DIEGO)*, which is based on Oja's method, to estimate the dominant eigenvector of a population covariance matrix in a distributed, streaming setting. The algorithm considers a distributed network of arbitrarily connected nodes without a central coordinator and assumes data samples continuously arrive at the individual nodes in a streaming manner. It is established in the paper that C-DIEGO can achieve an order-optimal convergence rate if nodes in the network are allowed to have enough consensus rounds per algorithmic iteration. Numerical results are also reported in the paper that showcase the efficacy of the proposed algorithm.**

***Index Terms*—Dimension reduction, distributed learning, Oja's method, principal component analysis, streaming data**

I. Introduction

Continuous learning from streaming data having uncorrelated features helps machine learning algorithms improve their performance. The ever-increasing rate of streaming data at geographically distributed nodes coupled with large data dimensions is becoming a core challenge for algorithms that store and process the data at a single computing node. Indeed, collecting and processing a massive number of high-dimensional samples at only one node is resource intensive. This calls for data representation algorithms to be studied under distributed settings where the streaming data can be processed in real time at individual nodes.

Principal Component Analysis (PCA) is one of the pioneer techniques that not only reduces the dimensions of the data but also yields uncorrelated features [1]. Our focus in this paper is on PCA from distributed streaming data, when there is no central coordinator in the system.

PCA is a well-studied problem, and its adaption under the streaming setting was first reported in the works of Krasulina [2] and Oja [3], who proposed stochastic methods that estimate the top eigenvector of a population covariance matrix $\mathbf{\Sigma}$. In most applications, estimating the dominant eigenvector is sufficient for data representation, provided there is an eigengap Λ between the top two eigenvalues of $\mathbf{\Sigma}$. However, these algorithms—being centralized solutions—can, at best, converge to the top eigenvector in a streaming setting at an asymptotic rate of $\mathcal{O}(1/\sqrt{t})$, where t is the time index of algorithmic iteration. Obtaining finite sample convergence rates for these algorithms has been an area of active research [4], [5], where the rate is improved under different choices of step size and the dependency on Λ is minimized.

Recently both Oja and Krasulina algorithms were studied under distributed settings, and an increase in convergence rate under the streaming data was provided. The rate improvement is first seen in the works [6] and [7], where it is shown that the sample complexity can be decreased by a factor equal to the number of nodes N in a network. The proposed algorithms in these works can provide convergence rates on the order of $\mathcal{O}(1/\sqrt{Nt})$ under the assumption that a network is Fully Connected (FC), where all nodes are able to communicate with other nodes directly or through the presence of a coordinator. However, no prior work has been done that provides convergence guarantees on the order of $\mathcal{O}(1/\sqrt{Nt})$ in a Non Fully Connected (NFC) network, where the nodes are assumed to be arbitrarily connected without the need of a central coordinator. This precludes the setup in which processing nodes reach exact averaging using a communication primitive such as `AllReduce` [8].

In this paper, we propose a quasi-two-time scale algorithm, termed Consensus-DIstributEd Generalized Oja (C-DIEGO), that provides an order-optimal convergence rate of $\mathcal{O}(1/\sqrt{Nt})$ for an NFC network. This rate is achieved under the assumption that the per-iteration communication complexity of the algorithm is $\mathcal{O}(T_{mix}\log(Nt))$, where the T_{mix} is the mixing time of a Markov chain associated with the network topology.

A. Relation to Prior Work

The idea of decorrelation and compression of the data samples using principal components dates back to the 1900s when PCA was originally proposed [1]. It was further shown that if data is streaming in nature, then one can estimate the top eigenvector of $\mathbf{\Sigma}$ under certain statistical conditions on data

This research is supported in part by the National Science Foundation under grants CCF-1907658, OAC-1940074, and CNS-2148104, and the Army Research Office under grant W911NF-21-1-0301.

distribution [4]. This was first achieved in 1970 by Krauslina [2], who proposed a stochastic approximation algorithm that converges asymptotically to the top eigenvector. A similar algorithm was later proposed in 1985 by Oja [3], which was based on the Hebbian rule [9]. Extensions of the Oja algorithm for multiple eigenvectors were first provided in [10]. Obtaining convergence rates under finite samples has always been challenging. It was shown in [4] that one could achieve a convergence rate of $\mathcal{O}(1/\sqrt{t})$ for both Krasulina and Oja algorithms. This result was further improved by removing the extra $\mathcal{O}(d^5)$ multiplicative factor in the analysis of the streaming Oja algorithm in [5], where d is the dimension of the data samples. Other papers, including [11], [12], provided convergence rates under finite samples in batch and mini-batch settings, which are not fit for our case of streaming data.

Recently, PCA algorithms were proposed for the distributed setting. These algorithms can be applied by splitting data either $i)$ by features or $ii)$ by samples per node. In the former case, each node has access to some subset of features of one or more data samples, whereas in the latter case, each node has access to one or more samples comprising of all features. A detailed review of such algorithms for both types is done in [13]. One type of feature splitting-based PCA method was proposed in [14], which extends the Oja algorithm in a distributed setting. However, this result only provides convergence guarantees in the asymptotic settings ($t \to \infty$). There is a collection of papers that focus on sample-wise data splitting methods of including, Cloud *K-SVD* [15], [16], the orthogonal iteration-based solution [17], the Decentralized Exact PCA (DEePCA) [18], and most recently FAST-PCA [19], and Distributed Sanger Algorithm (DSA) [20]. While these methods provide different results for estimates of either the principal eigenspace, the top k eigenvectors ($k \geq 1$), or only the dominant eigenvector, all of these methods either use batch setting or mini-batch setting and are not well-suited for our case of streaming data.

The increase in convergence rate by increasing the number of nodes in a network in the streaming setting is first seen in [6], which has extended the Krasulina algorithm to the distributed setting and shown that the optimal convergence rate of $\mathcal{O}(1/\sqrt{Nt})$ can be achieved after t iterations under the streaming setting. However, the analysis in this work assumes an FC network. In this paper, we assume a sample-wise splitting of data and provide convergence guarantees for the Oja algorithm in distributed, streaming settings, and show that the similar order-optimal rate of $\mathcal{O}(1/\sqrt{Nt})$ can be achieved in an NFC network provided that the nodes perform enough consensus rounds per algorithmic iteration t.

B. Our Contributions

The first main contribution of this paper is a novel quasi-two-time scale *Consensus-DIstributEd Generalized Oja (C-DIEGO)* algorithm that estimates the dominant eigenvector of $\mathbf{\Sigma}$ in a network of arbitrarily connected nodes under the assumption that data samples are distributed and streaming in nature. The algorithm achieves this by having multiple consensus rounds per algorithmic iteration t. The second major contribution of this paper is Theorem 1, which provides the sample complexity error bound for the proposed algorithm in an NFC network and shows that the order-optimal convergence rate of $\mathcal{O}(1/\sqrt{Nt})$ can be achieved for distributed, streaming PCA. Finally, we provide experimental results for the proposed algorithm that corroborates our theoretical results.

C. Notation and Organization

We use standard notation for denoting scalars, vectors, and matrices by lowercase, bold lowercase, and bold uppercase letters, respectively. The notation $\mathbf{a}_{i,t}$ denotes a vector at node i and time t, whereas the superscript on a vector, as $\mathbf{a}^{(t_c)}$, denotes updates after t_c rounds of consensus. In addition, unless otherwise stated, all operator norms are denoted as $\|\cdot\|_2$ and all vector l_2 norms by $\|\cdot\|$. Finally, the symbol $\mathcal{O}$ denotes the Big-O notation.

The rest of this paper is organized as follows: In Sec. II, we formulate the problem of distributed streaming PCA. In Sec. III, we introduce our C-DIEGO algorithm and provide its convergence analysis. In Sec. IV, we provide numerical results, and conclude in Sec. V. Finally, proof of the lemma associated with our main result is provided in Appendix A.

II. Problem Formulation

We consider a network of N geographically distributed computational nodes arbitrarily connected in an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{1, 2, \ldots, N\}$ denotes the set of N nodes and $\mathcal{E}$ denotes edges in the graph with $(i, j) \in \mathcal{E}$ whenever there is a connection between node i and node j. We also assume that the network remains fixed for the duration of our algorithm, and consider a sample-wise splitting of data where each data sample $\mathbf{x}_{i,t} \in \mathbb{R}^d$ comprising of the set of d features arrives at node i at every time t. We assume that these data samples are drawn independently from a zero-mean distribution having a covariance matrix $\mathbf{\Sigma}$. This distribution is assumed to remain fixed for the duration of our algorithm and satisfies the following conditions [5]:

- **A1:** $\left\|\mathbf{x}_{i,t}\mathbf{x}_{i,t}^T\right\|_2 \leq r$ almost surely.
- **A2:** $\left\|\mathbb{E}[(\mathbf{x}_{i,t}\mathbf{x}_{i,t}^T - \mathbf{\Sigma})(\mathbf{x}_{i,t}\mathbf{x}_{i,t}^T - \mathbf{\Sigma})^T]\right\|_2 \leq v$.

PCA aims to find a low-dimensional subspace $\mathbf{Y} \in \mathbb{R}^{d\times k}, d \gg k$, for the data samples. Specifically, PCA captures the maximum information in the data while decorrelating the features, which we can write as $\tilde{\mathbf{x}} = \mathbf{Y}^T\mathbf{x}$, where $\tilde{\mathbf{x}} \in \mathbb{R}^k$ and $\mathbf{Y}$ has orthonormal columns. This is done by requiring $\mathbb{E}[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T]_{ij} = \mathbb{E}[\mathbf{Y}^T\mathbf{x}\mathbf{x}^T\mathbf{Y}^T]_{ij} = 0, \forall i \neq j$, which is possible only when $\mathbf{Y}$ contain the top k eigenvectors of $\mathbf{\Sigma}$.

Thus, the goal of PCA is to find the eigenvectors of $\mathbf{\Sigma}$. However, true knowledge of $\mathbf{\Sigma}$ is unknown in reality, and estimating eigenvectors using the batch sample covariance matrix is not practical given the nature of data being distributed and streaming. In the streaming setting, the aim is to have an algorithm with $\mathcal{O}(d)$ computational complexity that updates the eigenvector estimate upon receiving the newest data sample. Therefore, we need stochastic PCA algorithms that can estimate eigenvectors in a distributed streaming setting. We propose the C-DIEGO algorithm for this purpose, which is a stochastic PCA algorithm that estimates the dominant eigenvector of $\mathbf{\Sigma}$, denoted as $\mathbf{q}_1$, when data is streaming and distributed. The algorithm comes with convergence guarantees for the eigenvector estimate at node i after t iterations.

Algorithm 1 Consensus-DIstributEd Generalized Oja (C-DIEGO) Algorithm

Input: Incoming data at N nodes $\{\mathbf{x}_{i,t}\}_{i=1}^{N}$ at time t generated from a fixed distribution of mean $\boldsymbol{\mu} = \mathbf{0}$ and covariance matrix $\mathbf{\Sigma}$, a step size sequence $\{\alpha_t \in \mathbb{R}_+\}$, doubly stochastic weight matrix $\mathbf{W}$, and total number of consensus rounds $T_c > 1$.
Initialize: All nodes are initialized with $\mathbf{v}_0 \in \mathbb{R}^d$ generated randomly over a unit sphere: $\|\mathbf{v}_0\| = 1$

```
1: for $t = 1, 2, \ldots,$ do
2:     (In Parallel) Node $i$ receives the data sample $\mathbf{x}_{i,t}$ and computes the Oja's correction $\boldsymbol{\xi}_{i,t}$:
           $\forall\, i \in \{1, 2, \ldots, N\}, \quad \boldsymbol{\xi}_{i,t} \leftarrow \left(\mathbf{x}_{i,t}\mathbf{x}_{i,t}^T\mathbf{v}_{i,t-1}\right)$
3:     for $t_c = 1, 2, \ldots, T_c$ do
           $\forall\, i \in \{1, 2, \ldots, N\}, \quad \boldsymbol{\xi}_{i,t}^{(t_c)} \leftarrow \sum_{j\in\mathcal{N}_i} w_{ij}\boldsymbol{\xi}_{j,t}^{(t_c-1)}$
4:     end for
5:     $\boldsymbol{\zeta}_{i,t} \leftarrow \frac{\boldsymbol{\xi}_{i,t}^{(T_c)}}{[\mathbf{W}^{T_c}\mathbf{e}_1]_i}$
6:     $\mathbf{v}_{i,t} \leftarrow \mathbf{v}_{i,t-1} + \alpha_t\boldsymbol{\zeta}_{i,t} \quad \forall\, i \in \{1, 2, \ldots, N\}$
7:     $\mathbf{v}_{i,t} \leftarrow \frac{\mathbf{v}_{i,t}}{\|\mathbf{v}_{i,t}\|} \quad \forall\, i \in \{1, 2, \ldots, N\}$
8: end for
```

Return: An estimate $\mathbf{v}_{i,t}$ of $\mathbf{q}_1$ is returned, $i = 1, 2, \ldots, N$.

III. Proposed Algorithm: Consensus-DIstributEd Generalized Oja (C-DIEGO)

We begin with the setup of arbitrarily connected nodes in a network, where it is assumed that no node is isolated, and the network lacks a central coordinator. We next define $\mathcal{N}_i := \{j : (i,j) \in \mathcal{E}\} \cup \{i\}$ to be the neighborhood of node i (including itself). The proposed algorithm C-DIEGO is given in Algorithm 1, in which $\mathbf{v}_{i,t}$ denotes the eigenvector estimate at node i after t algorithmic iterations.

In the algorithm, first all nodes are initialized with the same (unit-norm) vector $\mathbf{v}_0$. Next, data samples $\mathbf{x}_{i,t}$ arrive at each node i (Step 2), and the nodes simultaneously compute their respective local Oja's correction terms $\boldsymbol{\xi}_{i,t}$'s that need to be combined for updates to the prior eigenvector estimates $\mathbf{v}_{i,t}$'s. This is followed by an *inexact* consensus averaging among the $\boldsymbol{\xi}_{i,t}$'s (Steps 3–4), which involves the use of a doubly stochastic weight matrix $\mathbf{W}$ that adheres to the topology of our graph $\mathcal{G}$. The i,j entry of the $\mathbf{W}$ matrix is denoted by w_{ij}, where $w_{ij} = 0$ (whenever) $(i,j) \notin \mathcal{E}$.

Note that Steps 3–4 of Algorithm 1 correspond to an inexact averaging of the $\boldsymbol{\xi}_{i,t}$'s at each node i because of the nodes having a finite number of consensus communication rounds T_c with their neighbors, where $t_c \in \mathbb{N}$ is the index of consensus iteration. Indeed, if $T_c \to \infty$ then $\mathbf{W}^{T_c} \to \frac{1}{N}\mathbf{1}\mathbf{1}^T$, where $\mathbf{1}$ is an all-ones vector. This implies that exact averaging of the $\boldsymbol{\xi}_{i,t}$'s is possible among the nodes provided infinite consensus rounds are performed, but this is not feasible in practice.

Next, all nodes in the network normalize their respective inexact averages as $\boldsymbol{\zeta}_{i,t} = \frac{\boldsymbol{\xi}_{i,t}^{(T_c)}}{[\mathbf{W}^{T_c}\mathbf{e}_1]_i}$, where $\mathbf{e}_1 := [1, 0, \ldots, 0]^T$, and $[\cdot]_i$ denotes the i^{th} entry of a vector. This is followed by the local updates of the estimates (Step 6), and a final normalization in Step 7. The algorithm then repeats the process till convergence.

Note that Steps 3–5 lead to a major source of error in the C-DIEGO algorithm. To see this, consider a hypothetical version of the C-DIEGO algorithm where, instead of Steps 3–5, nodes use a *message passing interface* (MPI) communication primitive such as `AllReduce` [8] to reach exact averaging of the $\boldsymbol{\xi}_{i,t}$'s. In this scenario, we have $w_{ij} = 1/N$ for all edges of the graph, and the eigenvector estimate at each node will be the same, i.e., $\mathbf{v}_{i,t} = \mathbf{v}_{j,t}$. Let us call this estimate as exact averaging estimate. Since all nodes have the same estimate in this case, so we drop the subscript i in $\mathbf{v}_{i,t}$ and denote the exact averaging estimate by $\widehat{\mathbf{v}}_t$ after t iterations. Clearly we have an error between exact and inexact averaging estimates within each iteration t, denoted as $\boldsymbol{\epsilon}_{i,t} = \sum_{i=1}^{N}\boldsymbol{\xi}_{i,t} - \frac{\boldsymbol{\xi}_{i,t}^{(T_c)}}{[\mathbf{W}^{T_c}\mathbf{e}_1]_i}$. Handling of this error is the key in proving the bound between eigenvector estimates of exact averaging and inexact averaging.

Our eventual goal is to establish convergence of the eigenvector estimate in the case of inexact averaging, denoted by $\mathbf{v}_{i,t}$, to the true eigenvector of $\mathbf{\Sigma}$ denoted as $\mathbf{q}_1$.

Remark 1. The normalization by $[\mathbf{W}_c^T\mathbf{e}_1]_i$ in Step 5 is compensated by the normalization in Step 7, and thereby the output result is always a unit-norm vector. This normalization aids us in the convergence analysis of our algorithm.

A. Convergence Analysis

We now investigate the error achieved by the quasi-two-time scale C-DIEGO algorithm between the top eigenvector $\mathbf{q}_1$ of $\mathbf{\Sigma}$ and the estimate available at node $\mathbf{v}_{i,t}$ after t iterations. The ensuing analysis shows that the optimal convergence rate can be achieved without requiring infinite consensus rounds. Specifically, we prove that if $\mathcal{O}(T_{mix}\log(Nt))$ consensus rounds are performed per algorithmic iteration t then convergence is guaranteed at a sub-linear rate of $\mathcal{O}(1/\sqrt{Nt})$, which is optimal. The following theorem summarizes this contribution.

Theorem 1. *Consider the covariance matrix $\mathbf{\Sigma} = \mathbf{E}[\mathbf{x}_{i,t}\mathbf{x}_{i,t}^T]$ having dominant eigenvector $\mathbf{q}_1$ and eigenvalues $\lambda_1 > \lambda_2 \geq \ldots \geq \lambda_d$, and let $\tilde{\gamma}$ denote the Euler constant and C' be an absolute constant. Suppose assumptions **A1** and **A2** are satisfied, then for $\eta > 0.5$, $\beta = 20\max\left(\frac{r\eta}{(\lambda_1-\lambda_2)}, \frac{(v+\lambda_1^2)\eta^2}{(\lambda_1-\lambda_2)^2\log(1+\frac{\rho}{100})}\right)$, $\rho \in (0,1]$, and a step size of $\alpha_t = \frac{\eta}{(\lambda_1-\lambda_2)(\beta+t)}$, the following holds after t iterations with probability $1-\rho$:*

$$\max_{i=1,\ldots,N} \left\|\mathbf{q}_1\mathbf{q}_1^T - \mathbf{v}_{i,t}\mathbf{v}_{i,t}^T\right\|_2 \leq$$

$$\sqrt{\frac{C'\log(1/\rho)}{\rho^3}\left(d\left(\frac{\beta}{t}\right)^{2\eta}+\frac{v(\beta+1)^2\eta^2}{Nt\beta^2(2\eta-1)(\lambda_1-\lambda_2)^2}\right)}$$
$$+\frac{2r\eta}{t^{3/2}\sqrt{N}(\lambda_1-\lambda_2)}\left(\frac{t}{\beta}+\tilde{\gamma}\right), \qquad (1)$$

provided that number of the consensus iterations within each algorithmic iteration t satisfies $T_c = \mathcal{O}(T_{mix}\log(Nt))$.

Theorem 1 provides a bound on the sample complexity of the error between the dominant eigenvector $\mathbf{q}_1$ and the estimate $\mathbf{v}_{i,t}$ available at node i after t iterations in an NFC network. The quantity T_{mix} in the theorem is the mixing time of the Markov chain associated with the doubly stochastic weight matrix $\mathbf{W}$ of the graph $\mathcal{G}$ defined as [21]:

$$T_{mix} = \max_{i=1,\ldots,N}\inf_{t\in\mathbb{N}}\left\{t : \left\|\mathbf{e}_i^T W^t - \frac{1}{N}\mathbf{1}^T\right\| \le \frac{1}{2}\right\}. \qquad (2)$$

The bound is expressed as a subspace difference and is a function of two different errors: i) the error due to the centralized Oja's solution, which decays at a rate of $\mathcal{O}(1/\sqrt{Nt})$, and ii) the error due to the quasi-two-time scale nature of the algorithm. The proof of Theorem 1 relies on two lemmas. The first lemma provides the error bound between $\mathbf{q}_1$ and $\widehat{\mathbf{v}}_t$ and is stated as follows.

Lemma 1. *Fix some $\rho \in (0,1]$ and let $\alpha_t = \frac{\eta}{(\lambda_1-\lambda_2)(\beta+t)}$, where $\eta > 0.5$, $\beta = 20\max\left(\frac{r\eta}{(\lambda_1-\lambda_2)}, \frac{(v+\lambda_1^2)\eta^2}{(\lambda_1-\lambda_2)^2\log(1+\frac{\rho}{100})}\right)$, and λ_i denotes the ith largest eigenvalue of $\mathbf{\Sigma}$. Then assuming **A1** and **A2** hold, and under the case when exact averaging of $\boldsymbol{\xi}_{i,t}$'s is possible in Algorithm 1, the eigenvector estimate $\widehat{\mathbf{v}}_t$ at every node converges to $\mathbf{q}_1$ after t iterations in the sense that*

$$\sin^2(\mathbf{q}_1,\widehat{\mathbf{v}}_t) \le$$
$$\frac{C'\log(1/\rho)}{\rho^3}\left(d\left(\frac{\beta}{t}\right)^{2\eta}+\frac{v(\beta+1)^2\eta^2}{Nt\beta^2(2\eta-1)(\lambda_1-\lambda_2)^2}\right), \qquad (3)$$

with probability at least $1-\rho$, where C' is an absolute constant.

The proof of Lemma 1 is provided in [7, Chapter 5]. This lemma shows that the rate increase of $\sqrt{N}$ is achieved if the Oja algorithm runs in a distributed streaming setting within an FC network where exact averaging is possible. We now provide the second lemma, which bounds the error at node i between the eigenvector estimates corresponding to an FC network (ideal case) and an NFC network (our setting) after t algorithmic iterations provided that in both topologies, the algorithm is initialized with the same eigenvector estimate.

Lemma 2. *Consider all nodes to have been initialized with $\mathbf{v}_0$ and suppose the eigenvalues of $\mathbf{\Sigma}$ satisfies $\lambda_1 > \lambda_2$. Then under assumptions **A1** and **A2** and for any $\delta > 0$, the following holds true:*

$$\|\widehat{\mathbf{v}}_t - \mathbf{v}_{i,t}\| \le \frac{\delta N r\eta}{(\lambda_1-\lambda_2)}\left(\frac{t}{\beta}+\tilde{\gamma}\right), \qquad (4)$$

where $\beta = 20\max\left(\frac{r\eta}{(\lambda_1-\lambda_2)}, \frac{(v+\lambda_1^2)\eta^2}{(\lambda_1-\lambda_2)^2\log(1+\frac{\rho}{100})}\right)$ and $\tilde{\gamma}$ is the Euler constant.

The proof of this lemma is provided in Appendix A. By choosing $\delta = 1/(Nt)^{\frac{3}{2}}$, Lemma 2 allows us to have the same error decay rate as of Lemma 1 and thus to make the convergence rate of the order of $\mathcal{O}(1/\sqrt{Nt})$ for both errors. By using this choice of δ and the modified consensus averaging [22, Theorem 5], we obtain the optimal consensus rounds of $T_c = \mathcal{O}(T_{mix}\log(Nt))$. We now use Lemma 1 and Lemma 2 to prove the main result of this paper.

Proof of Theorem 1. Let $\widehat{\mathbf{v}}_t$ denote the eigenvector estimate within C-DIEGO at time t when exact averaging is utilized in place of Steps 3–5. Then from Lemma 1 and applying the fact that $\left\|\mathbf{q}_1\mathbf{q}_1^T - \widehat{\mathbf{v}}_t\widehat{\mathbf{v}}_t\right\|_2 = |\sin\theta|$, where $\theta = \angle(\mathbf{q}_1,\widehat{\mathbf{v}})$, we have:

$$\left\|\mathbf{q}_1\mathbf{q}_1^T - \widehat{\mathbf{v}}_t\widehat{\mathbf{v}}_t^T\right\|_2 \le$$
$$\sqrt{\frac{C'\log(1/\rho)}{\rho^3}\left(d\left(\frac{\beta}{t}\right)^{2\eta}+\frac{v(\beta+1)^2\eta^2}{Nt\beta^2(2\eta-1)(\lambda_1-\lambda_2)^2}\right)}. \qquad (5)$$

Next, using the fact that $\left\|\mathbf{a}\mathbf{a}^T - \mathbf{b}\mathbf{b}^T\right\|_2 \le 2\left\|\mathbf{a}-\mathbf{b}\right\|$, from Lemma 2, we have:

$$\left\|\widehat{\mathbf{v}}_t\widehat{\mathbf{v}}_t^T - \mathbf{v}_{i,t}\mathbf{v}_{i,t}^T\right\|_2 \le \frac{2\delta N r\eta}{(\lambda_1-\lambda_2)}\left(\frac{t}{\beta}+\tilde{\gamma}\right) \qquad (6)$$

Next for a given node i, we have from triangle inequality:

$$\left\|\mathbf{q}_1\mathbf{q}_1^T - \mathbf{v}_{i,t}\mathbf{v}_{i,t}^T\right\|_2 \le$$
$$\left\|\mathbf{q}_1\mathbf{q}_1^T - \widehat{\mathbf{v}}_t\widehat{\mathbf{v}}_t^T\right\|_2 + \left\|\widehat{\mathbf{v}}_t\widehat{\mathbf{v}}_t^T - \mathbf{v}_{i,t}\mathbf{v}_{i,t}^T\right\|_2. \qquad (7)$$

Using (5) and (6) in (7) completes the proof. ■

IV. Numerical Results

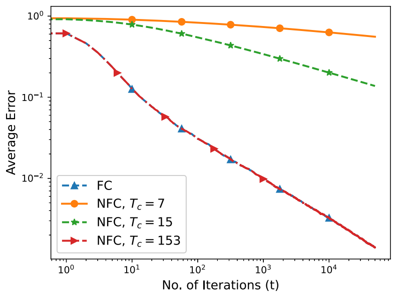
In this section, we demonstrate the convergence behavior of C-DIEGO through numerical experiments. We generate an undirected connected network of N nodes for each experiment using the Erdős-Rényi model with parameter p. The weight matrix $\mathbf{W}$ is generated using the Metropolis-Hastings algorithm [23], in which the weights computed for each edge use the following local degree rule:

$$w_{i,j} = \frac{1}{\max\{d_i, d_j\}}, \qquad (8)$$

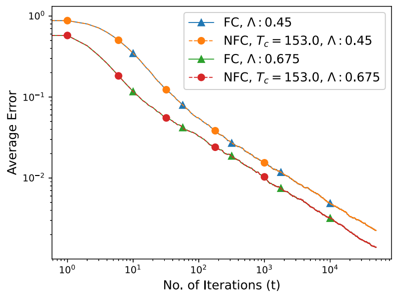
where d_i and d_j denote the degree of node i and node j, respectively. We further use the following error metric to compute the error between the eigenvector estimate at node i at time t and the true eigenvector $\mathbf{q}_1$ of $\mathbf{\Sigma}$ to comply with our analysis:

$$e_{i,t} = \max_{i=1,2,\ldots,N}\left|\sqrt{1-\frac{(\mathbf{v}_{i,t}^T\mathbf{q}_1)^2}{\mathbf{v}_{i,t}^T\mathbf{v}_{i,t}}}\right|. \qquad (9)$$

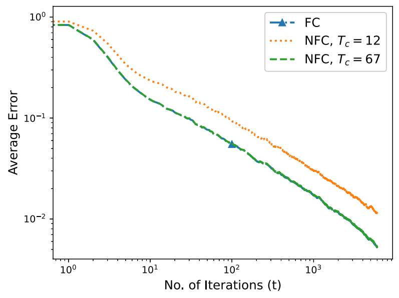
This error metric computes the sine angle between $\mathbf{q}_1$ and $\mathbf{v}_{i,t}$ and is a standard used in convergence analysis for PCA



(a) Effect of different values of T_c with $\Lambda = 0.68$ in an NFC versus FC network.



(b) Effect of convergence under different values of Λ in an FC vs NFC network.



(c) Effect of different values of T_c on MNIST dataset.

Fig. 1: Average error for different values of T_c and Λ for both synthetic and real-world data.

algorithms. The maximum number of consensus rounds is taken to be $T_c = T_{mix}\tilde{c}\log(Nt)$, which we define as our optimal T_c. We set the tuning parameter $\tilde{c} = 3/2$ for all our experiments, and T_{mix} is computed using the rule defined in (2). We provide experimental results with both synthetic data and real-world MNIST dataset.

A. Experiments using Synthetic Data

In synthetic data experiments, we generate an undirected Erdős-Rényi graph of $N = 40$ nodes with parameter $p = 0.1$. The data samples at each node $\mathbf{x}_{i,t}$ having $d = 20$ dimensions are generated using a zero-mean multivariate Gaussian distribution with covariance matrix $\mathbf{\Sigma}$ with a pre-defined eigengap $\Lambda = \lambda_1 - \lambda_2$. At every iteration t, the error is computed using the metric defined in (9), the results are further averaged over 50 Monte-Carlo trials, and the step size is set as $\alpha_t = \frac{\alpha}{t}$, where $\alpha = 0.05$ is chosen after cross-validation. Figure 1a shows the experimental results for different values of T_c under fixed $\Lambda = 0.68$. We can see that for optimal T_c, the gap between the error of the NFC and FC graphs vanishes. Moreover, the slope at the tail of the FC and NFC curve is -0.51, which coincides with our analysis. Next, we provide an additional experiment illustrating the impact of Λ on convergence behavior in Figure 1b. We performed this experiment by generating an Erdős-Rényi graph of $N = 40$ nodes with parameter $p = 0.1$. The optimal $T_c = 153$, and we can observe that the convergence is better when the Λ is larger, which coincides with our theoretical results.

B. Experiments using Real-World Data

In this section, we demonstrate the performance of our algorithm using real-world data. For this purpose, we choose the MNIST dataset of handwritten images [24]. Each sample of this dataset has $d = 784$ dimensions, and the total number of samples is $n = 60,000$. We generate an undirected Erdős-Rényi graph of $N = 10$ nodes with parameter $p = 0.1$. As the distribution of the MNIST dataset is unknown, we use the batch method estimate of $\mathbf{q}_1$ using the sample covariance matrix for our algorithm performance. The results are averaged over 50 Monte-Carlo trials, where random shuffling of data is performed in each trial. After cross-validation, the initial step size is set as $\alpha = 0.01$. The results are shown in Figure 1c, and we can observe the best performance is achieved when the optimal T_c is used in an NFC graph.

V. Conclusion

In this paper, we have proposed a new quasi two-time scale distributed algorithm based on the Oja algorithm termed C-DIEGO that guarantees the order-optimal convergence rate for the dominant eigenvector of the population covariance matrix in a non-fully connected network in which samples arrive in a streaming manner. Theoretical analysis of our algorithm shows that we can recover the same convergence rate of a fully connected network provided $\mathcal{O}(T_{mix}\log(Nt))$ consensus rounds are performed in each algorithmic iteration t, and our experimental results confirm the efficacy of our analysis. The generalization of this analysis to the top k eigenvectors is a promising future direction for this work.

Appendix A
Proof of Lemma 2

In order to prove Lemma 2, we first prove a supporting Lemma 3, which bounds the consensus error at each node i. Lemma 3 uses the following proposition from the literature that characterizes the convergence behavior of vector consensus averaging as a function of the number of consensus rounds.

Proposition 1. *[22, Theorem 5] Define $\boldsymbol{\xi}_{i,t}^{(T_c)} \in \mathbb{R}^{d\times 1}$ to be a vector at node i after T_c consensus rounds for $i \in \{1, 2, \dots, N\}$, where the initial value at each node is given by $\boldsymbol{\xi}_{i,t}^{(0)}$. Let $\bar{\boldsymbol{\zeta}}_t = \sum_{i=1}^{N} \boldsymbol{\xi}_{i,t}^{(0)}$, and define $\boldsymbol{\zeta}_t^{'} = \sum_{i=1}^{N} |\boldsymbol{\xi}_{i,t}^{(0)}|$, where $|\mathbf{a}|$ denotes the element-wise absolute value of $\mathbf{a}$. For any $\delta > 0$ and $T_c = \mathcal{O}(T_{mix}\log\delta^{-1})$, the approximation error of averaging consensus satisfies $\left\| \frac{\boldsymbol{\xi}_{i,t}^{(T_c)}}{[\mathbf{W}^{T_c}\mathbf{e}_1]_i} - \bar{\boldsymbol{\zeta}}_t \right\| \leq \delta \left\| \boldsymbol{\zeta}_t^{'} \right\|$, $\forall i$.*

We now state and prove Lemma 3, which uses Proposition 1:

Lemma 3. *Let the output of Step 5 of Algorithm 1 in the case of exact averaging to be $\bar{\boldsymbol{\zeta}}_t$, and in the case of inexact averaging, denote the output to be $\boldsymbol{\zeta}_{i,t}$. Then from*

*Proposition 1 and under the assumption **A1**, the following is true at any time t:*

$$\|\bar{\boldsymbol{\zeta}}_t - \boldsymbol{\zeta}_{i,t}\| \leq \delta N r. \tag{10}$$

Proof. The error due to finite consensus at node i after t iterations is $\boldsymbol{\epsilon}_{i,t} = \bar{\boldsymbol{\zeta}}_t - \boldsymbol{\zeta}_{i,t}$. Then,

$$\begin{aligned}
\|\bar{\boldsymbol{\zeta}}_t - \boldsymbol{\zeta}_{i,t}\| &= \left\| \sum_{i=1}^{N} \boldsymbol{\xi}_{i,t} - \frac{\boldsymbol{\xi}_{i,t}^{(T_c)}}{[\mathbf{W}^{(T_c)}\mathbf{e}_1]_i} \right\| && (11)\\
&\leq \delta \left\| \sum_{i=1}^{N} |\boldsymbol{\xi}_{i,t}| \right\| && (12)\\
&\leq \delta\sqrt{N} \left(\sum_{k=1}^{d}\sum_{i=1}^{N} |\boldsymbol{\xi}_{i,t}[k]|^2 \right)^{1/2} && (13)\\
&= \delta\sqrt{N} \left(\sum_{i=1}^{N} \|\boldsymbol{\xi}_{i,t}\|^2 \right)^{1/2} && (14)\\
&= \delta\sqrt{N} \left(\sum_{i=1}^{N} \|\mathbf{x}_{i,t}\mathbf{x}_{i,t}^T\mathbf{v}_{i,t-1}\|^2 \right)^{1/2} && (15)\\
&\leq \delta\sqrt{N} \left(\sum_{i=1}^{N} \|\mathbf{x}_{i,t}\mathbf{x}_{i,t}^T\|_2^2 \|\mathbf{v}_{i,t-1}\|^2 \right)^{1/2} && (16)\\
&\leq \delta N r && (17)
\end{aligned}$$

where (12) follows from Proposition 1, we apply Cauchy-Schwarz inequality in (13), and Oja's correction term in (15), and finally (16) follows from the fact that $\|\mathbf{A}\mathbf{x}\| \leq \|\mathbf{A}\|_2 \|\mathbf{x}\|$. ■

We are now ready to prove our main Lemma 2.

Proof. Consider all nodes to have been initialized with $\mathbf{v}_0$. We first prove that the following holds after t iterations:

$$\|\widehat{\mathbf{v}}_t - \mathbf{v}_{i,t}\| \leq \delta N r \sum_{i=1}^{t} \alpha_i. \tag{18}$$

To this end, notice that at $t = 1$, we have:

$$\begin{aligned}
\|\widehat{\mathbf{v}}_1 - \mathbf{v}_{i,1}\| &= \|\mathbf{v}_0 + \alpha_1\bar{\boldsymbol{\zeta}}_1 - \mathbf{v}_0 + \alpha_1\boldsymbol{\zeta}_{i,1}\| && (19)\\
&= \|\alpha_1(\bar{\boldsymbol{\zeta}}_1 - \boldsymbol{\zeta}_{i,1})\| \leq \alpha_1 N\delta r, && (20)
\end{aligned}$$

where (20) follows from Lemma 3. Now assuming (18) holds at $t = k-1$, then at $t = k$ we have:

$$\begin{aligned}
\|\widehat{\mathbf{v}}_k - \mathbf{v}_{i,k}\| &= \|\widehat{\mathbf{v}}_{k-1} + \alpha_k\bar{\boldsymbol{\zeta}}_k - \mathbf{v}_{i,k-1} - \alpha_k\boldsymbol{\zeta}_{i,k}\| && (21)\\
&\leq \|\widehat{\mathbf{v}}_{k-1} - \mathbf{v}_{i,k-1}\| + \alpha_k \|\bar{\boldsymbol{\zeta}}_k - \boldsymbol{\zeta}_{i,k}\| && (22)\\
&\leq \delta N r \sum_{i=1}^{k-1} \alpha_i + \alpha_k \delta N r = \delta N r \sum_{i=1}^{k} \alpha_i, && (23)
\end{aligned}$$

which proves (18). Now plugging $\alpha_i = \frac{\eta}{(\lambda_1-\lambda_2)(\beta+i)}$ in (18) and the fact that $\sum_{i=1}^{t} \alpha_i$ is a partial harmonic series we have:

$$\begin{aligned}
\|\widehat{\mathbf{v}}_t - \mathbf{v}_{i,t}\| &= \frac{\delta N r \eta}{(\lambda_1 - \lambda_2)} \left(\log(1 + \frac{t}{\beta}) + \tilde{\gamma} \right) && (24)\\
&\leq \frac{\delta N r \eta}{(\lambda_1 - \lambda_2)} \left(\frac{t}{\beta} + \tilde{\gamma} \right). && (25)
\end{aligned}$$

Note that in (25), we used the fact that $\log(1 + x) \leq x$. ■

References

[1] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of Educational Psychology*, p. 417, 1933.

[2] T. Krasulina, "Method of stochastic approximation in the determination of the largest eigenvalue of the mathematical expectation of random matrices," *Automatation and Remote Control*, pp. 50–56, 1970.

[3] E. Oja, "Simplified neuron model as a principal component analyzer," *Journal of Mathematical Biology*, pp. 267–273, 1982.

[4] A. Balsubramani, S. Dasgupta, and Y. Freund, "The fast convergence of incremental PCA," *Proc. Advances in Neural Information Processing Systems*, 2013.

[5] P. Jain, C. Jin, S. M. Kakade, P. Netrapalli, and A. Sidford, "Streaming PCA: Matching matrix Bernstein and near-optimal finite sample guarantees for Oja's algorithm," in *Proc. Conference on Learning Theory*. PMLR, 2016, pp. 1147–1164.

[6] H. Raja and W. Bajwa, "Distributed stochastic algorithms for high-rate streaming principal component analysis," *Transactions on Machine Learning Research*, 2022. [Online]. Available: https://openreview.net/forum?id=CExeD0jpB6

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

[8] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine *et al.*, "Open MPI: Goals, concept, and design of a next generation MPI implementation," in *Proc. European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting*. Springer, 2004, pp. 97–104.

[9] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

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

[11] O. Shamir, "A stochastic PCA and SVD algorithm with an exponential convergence rate," in *Proc. International Conference on Machine Learning*, 2015, pp. 144–152.

[12] P. Xu, B. He, C. De Sa, I. Mitliagkas, and C. Re, "Accelerated stochastic power iteration," in *Proc. International Conference on Artificial Intelligence and Statistics*. PMLR, 2018, pp. 58–67.

[13] S. X. Wu, H.-T. Wai, L. Li, and A. Scaglione, "A review of distributed algorithms for principal component analysis," *Proceedings of the IEEE*, pp. 1321–1340, 2018.

[14] L. Li, A. Scaglione, and J. H. Manton, "Distributed principal subspace estimation in wireless sensor networks," *IEEE Journal of Selected Topics in Signal Processing*, pp. 725–738, 2011.

[15] H. Raja and W. U. Bajwa, "Cloud K-SVD: Computing data-adaptive representations in the cloud," in *Proc. 51st Annual Allerton Conference on Communication, Control, and Computing*, 2013, pp. 1474–1481.

[16] ——, "Cloud K-SVD: A collaborative dictionary learning algorithm for big, distributed data," *IEEE Transactions on Signal Processing*, pp. 173–188, 2015.

[17] A. Gang, B. Xiang, and W. U. Bajwa, "Distributed principal subspace analysis for partitioned big data: Algorithms, analysis, and implementation," *IEEE Transactions on Signal and Information Processing over Networks*, 2021.

[18] H. Ye and T. Zhang, "DeEPCA: Decentralized Exact PCA with linear convergence rate." *J. Mach. Learn. Res.*, pp. 1–27, 2021.

[19] A. Gang and W. U. Bajwa, "FAST-PCA: A fast and exact algorithm for distributed principal component analysis," *IEEE Transactions on Signal Processing*, pp. 6080–6095, 2022.

[20] ——, "A linearly convergent algorithm for distributed principal component analysis," *Signal Processing*, p. 108408, 2022.

[21] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip and mixing times of random walks on random graphs," 2004.

[22] D. Kempe and F. McSherry, "A decentralized algorithm for spectral analysis," *Journal of Computer and System Sciences*, pp. 70–83, 2008.

[23] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing Markov chain on a graph," *SIAM review*, pp. 667–689, 2004.

[24] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, pp. 141–142, 2012.