NONCONVEX REPRESENTATION LEARNING FROM DISTRIBUTED DATASETS

by

HAROON RAJA

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Prof. Waheed U. Bajwa

And approved by

New Brunswick, New Jersey

October, 2019

ABSTRACT OF THE DISSERTATION

Nonconvex Representation Learning from Distributed Datasets

By HAROON RAJA

Dissertation Director:

Prof. Waheed U. Bajwa

Data representation is an important information processing task which finds use in diverse engineering applications like signal processing, machine learning, medical imaging, and geophysical data analysis, to name a few. Once a good representation model is known for a given application then the next question is learning that model under practical constraints imposed by the application. Two such constraints are: *i)* data is available at geographically distributed sites, and *ii)* streaming data. In such scenarios distributed, decentralized, and online algorithms can be deployed for solving data representation problems, which are the focus of this dissertation. Specifically, this thesis focuses on solving following three problems: *i)* Solution for PCA for high-rate streaming data, *ii)* collaborative dictionary learning for big, distributed data, and *iii)* through the wall radar imaging (TWRI) in distributed settings. This thesis proposes new methods to tackle challenges arising due to distributed and steaming nature of data, providing theoretical analysis of the proposed methodologies (except TWRI), and finally using simulations to demonstrate the efficacy of the proposed methods.

Acknowledgements

It was a slow roller coaster ride made interesting and full of learning experiences thanks to all the wonderful people I was lucky enough to meet during the ups and downs of this ride. I would like to start by thanking the person in the control room, i.e., my adviser Prof. Waheed Bajwa. I thank him for: his patience (especially in the start), teaching me everything I know about mathematical rigor, opening my eyes to a new level of detail. But among all the great things he did for me the thing I appreciate the most is the fact that he picked on my biggest weakness (at least in my opinion) in my early days as a graduate student and helped me a lot over the years to overcome that. I would also like to thank my committee members Prof. Anand Sarwate, Prof. Mert Gurbuzbalaban, Prof. Mike Wakin, and Prof. Roy Yates for providing insightful comments on my thesis overall and especially the ones related to Chapter 2. I would like to thank Anand and Roy for teaching me a great deal through the courses I took with them and also as mentor figures throughout my time at Rutgers as a graduate student. I specially thank Roy for being there for my first and the last talk as a graduate student (among other talks). Your support has always given me tremendous amount of confidence. I would like to thank Mike for being a great collaborator during last year of my graduate school. I learned a great deal about collaborations observing Mike managing a project between three research groups. Finally, I would like to thank Mert for suggestions on making some of the bounds tighter in Chapter 2.

I would like to thank all my lab members for their support throughout my time at Rutgers. I especially thank Tong Wu, Talal Ahmed, and Zahra Shakeri for being there for me in the start, which I believe to be the most difficult phase of grad school. I would also like to acknowledge the positive influence of my friends Mudassir Shabbir and Mohsen Ghassemi towards my growth as a researcher and a person during my time at Rutgers. I thank Mudassir for all the discussions we had during my starting days. He helped me stay in touch with my love for Mathematics by bringing up Math puzzles in anecdotal form during our conversations (during the times I was feeling frustrated while proving something). I would like to thank the administrative staff at Rutgers ECE for taking care of logistics. A special thank goes to Prof. Zoran Gajic, our graduate program director, for his kindness and guidance throughout my PhD. I would also like to thank John Scafidi for always being there whenever his help was needed. Thank you for printing all those posters John!

I would also like to thank people who helped me get to the graduate school in the first place. Most important people during that phase of my life were my mentors; Prof. Usman Ilyas and Prof. Ali Khayam who taught me basics of research and graduate school admissions. Secondly, my friends from undergraduate; Hussain Kazmi and Sarmad Munir for providing motivation at the times when I needed it the most.

Finally, I would like to thank my family for being there for me throughout my life. There is no way I can repay my parents, Raja Hafeez Ullah and Tahira Hafeez, for all the sacrifices they have made for me and my siblings. I thank you the most for setting us on the path of self improvement. Through practice, you made us understand that with hard work one can always become better at anything in life. In my opinion, that is the best gift you could have given us. I will end by saying thank you to my siblings, Mamoon, Zainab, and Amna for all the fun we had and for always listening to my problems and helping me stay rational (not a complete list of things I want to thank you guys for).

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Data representation is an important first step towards solving tasks like inference, detection, estimation, etc. In practice since we do not know the data generation model in advance, data driven learning methods have been successfully used to learn a data generation model from sample data, e.g., principal component analysis (PCA), linear discriminant analysis (LDA), dictionary learning, autoencoders, etc [4, 5]. Learning these data representations from sample data can be a computationally intensive task. Due to this, devising computationally efficient methods for data representation problems has been an active area of research over the years [4, 6, 7]. Further computational challenges arise due to practical considerations. Most prevalent practical considerations in modern times have been: *i)* high volumes of data (large sample size), *ii)* high dimensional data (large number of features), *iii)* streaming data (online settings), and *iv)* data arriving at geographically distributed locations (distributed data). Under these settings, classical numerical methods to solve these problems are not very effective. Due to this, there has been an increased interest in developing efficient methods to solve these problems. Among the above mentioned practical considerations this dissertation focuses on streaming and distributed data settings. This type of data is common in online object tracking [8, 9], autonomous vehicles[10, 11], social media websites [12–14], news websites [15–18], etc.

1.2 Challenges due to big, distributed data

This dissertation leverages methods from decentralized methods, distributed and parallel computation, stochastic methods, and online/adaptive algorithms to solve some nonconvex learning problems i.e., principal component analysis (PCA), dictionary learning, and block structured dictionary learning for through the wall radar imaging. To be specific we can categorize our reasons for distributed methods as following:

1. *(Computational challenges.)* Large sample size and high dimensionality of data can render computational resources at a single processor to be insufficient and hence in such scenarios we need to distribute computation across multiple processing nodes. This situation presents itself in many applications to name a few: image classification for social media or autonomous vehicles, anomaly detection in communication networks, scientific applications like high energy particle Physics. Algorithms presented in Chapter 2 and Chapter 3 are devised to tackle some of the computational challenges arising due to these practical situations.

2. *(Privacy and security.)* In some applications due to privacy and security reasons we cannot aggregate all the data at one centralized location. For example, sharing medical data among different providers and between patients and providers needs to be done in a secure and privacy preserving manner [19–22]. Other prominent examples where privacy and security are of paramount importance include financial data [23], military applications, power grid, etc. Methods presented in Chapter 3 and Chapter 4 can be used in such scenarios.

3. *(Communication cost.)* Issues related to communications e.g., communication efficiency and robustness to node failures, provide motivation for developing decentralized methods. To some extent, methods proposed in Chapter 3 and Chapter 4 achieve this by providing better scaling in terms of data communication and resilience against node failures.

4. *(Data storage.)* In large scale machine learning sometimes it is not feasible to store data at a single location due to large storage requirements, distributed methods can be used to circumvent this issue (Chapter 3 and Chapter 2).

1.3 Distributed versus decentralized algorithms

Terms distributed algorithms and decentralized algorithms can mean different things depending on the research community. In order to avoid the confusion we clarify here what we mean by these terms in this thesis. We use term *distributed methods* to describe algorithms where we have a centralized server/fusion center present and each node can send information to the centralized server for decision making. The star topology and a tree structured graph are two examples for such methods. These methods find applications in data center like settings where reliable communication to a central node can be guaranteed. In contrast, *decentralized algorithms* have no central decision making node and all the nodes in network collaborate to reach a decision. Some example graph structures include line graph, expanders, random graphs, etc. These algorithms are applicable in applications like sensor networks, Internet of things (IoT), autonomous vehicles/robots. In these applications reliable communication to a central node can be difficult to guarantee and in addition node failures can be an issue.

1.4 Overview and contributions

Main contributions of this dissertation include a new distributed method for solving PCA for high rate streaming data settings, a decentralized method for collaborative dictionary learning for big, distributed data, and a decentralized method for through the wall radar imaging (TWRI). Further details of these contribution are given as follows:

1. Our first contribution, explained in more detail in Chapter 2, is solving PCA for high rate streaming settings. We consider high rate streaming settings to be when data arrival rate is faster than the processing rate at one node and hence if multiple processing nodes are available then we need to distribute the processing across these multiple processors to use all the data samples. Krasulina's method is one of the classical methods for solving PCA in streaming settings. In this thesis we propose a distributed variant of Krasulina's method which works for such high rate streaming settings. The main challenge here is to show analytically that the proposed distributed method will have same sampling complexity in *high-rate* streaming settings as the classical Krasulina's method in *low-rate* streaming settings (when one processor can process all the samples). For our analysis, we take inspiration from mini-batching based techniques that are developed to speed up convergence of convex optimization problems [24–27]. These techniques are not directly applicable for PCA since it is a nonconvex optimization problem and therefore a new

analysis is provided for distributed Krasulina's method. We further extend our distributed Krasulina's method to propose a distributed mini-batch version of Krasulina's method to solve PCA in high rate streaming settings due to practical benefits of mini-batching, for example, updating iterate in memory less frequently. Further, we provide simulations over synthetic and real world data to prove efficacy of the proposed distributed mini-batch method.

2. In Chapter 3, we consider big data settings where data is available at spatially distributed servers. Due to high volumes of data, it may not be feasible to aggregate data at one centralized node; in addition, privacy reasons may preclude from data sharing as well (e.g., medical data). For such settings, we propose a distributed variant of the K-SVD algorithm [4], which is one of the most well known method for dictionary learning for batch data. The goal in dictionary learning for data representation is to learn an overcomplete basis set which gives us sparse representation of data. Specifically, the goal is to learn both the dictionary and sparse representations of the training data. Most dictionary learning methods use an alternating minimization approach to solve this problem [4, 6]. Our first contribution here is to provide a fully distributed solution using *consensus averaging* to propose a distributed variant of K-SVD and hence term it as *cloud* K-SVD. Our main algorithmic contribution is an insight that in distributed settings if we have a good enough estimate of the dictionary at each node then we can perform sparse representation part locally. In centralized settings, K-SVD updates each column of the dictionary separately, by performing a rank-1 approximation of some matrix. Since this matrix depends on sample data, which is distributed across different nodes, we proposed a distributed variant of power method to compute the rank-1 estimate. Our second main contribution here is to show that if we perform enough iterations of consensus averaging then cloud K-SVD converges to the solution of the K-SVD algorithm. Finally, we provide simulations using synthetic and real world to show the effectiveness of the cloud K-SVD algorithm.

3. Finally, in Chapter 4 we solve through the wall radar imaging (TWRI) problem in distributed settings. TWRI can be posed as a simultaneous sparse recovery problem (see e.g., [28, 29]) with a block structured dictionary. Additionally, this dictionary can be completely described if we know wall locations of the room/space under interrogation. In practice, we can only know the location of outer surface of the wall and there is always an uncertainty in inner wall location. Hence, we can pose dictionary learning problem

in this case as a parametric dictionary learning problem, where wall locations (inner surface) are the parameters we need to find. Centralized methods for solving this parametric dictionary learning problem have been proposed in [28]. Solutions proposed in [28], use particle swarm optimization (PSO) which is a greedy method and quasi-Newton method (a gradient based approach) to estimate the wall locations. Our main contribution here lies in proposing distributed variants of PSO and quasi-Newton methods to estimate wall locations in a distributed manner. Our solutions rely on consensus averaging to develop distributed variants of these methods. Furthermore, large Lipschitz constant of objective function and presence of spurious local minimums presents additional challenges. These challenges preclude us from using standard distributed quasi-Newton implementation like [30]. Due to these issues we need to develop a solution that is tailored to specific requirements of the TWRI problem. The effectiveness of proposed methods is showed using synthetic data.

Chapter 2

DM-Krasulina: A Distributed Mini-Batch Algorithm for High-Rate Streaming Principal Component Analysis

This chapter considers the problem of computing the principal eigenvector of a covariance matrix from independent and identically distributed data samples in high-rate streaming settings. Krasulina's method is one of the most popular classical stochastic method to accomplish this task. Modern day applications like image segmentation in videos present a computational challenge to Krasulina's method due to the high streaming rate of the data samples. In such situations, streaming rate can be high enough such that a single processor cannot finish an iteration of Krasulina's method in time before a new data sample arrives. Assuming the availability of multiple processing nodes, the first main contribution of this chapter is proposing a distributed variant of Krasulina's method (D-Krasulina) that can keep up with the high streaming rate of data samples by distributing the computational load across multiple processing nodes. Second contribution of this work is showing theoretically that the proposed D-Krasulina's method converges to the principal eigenvector of the population covariance matrix at an optimal rate, i.e., for an N node network, after t iterations of distributed Krasulina's method error in top eigenvector estimate will be on the order of $O(1/Nt)$. Furthermore, due to the practical advantages of mini-batching this chapter proposes a mini-batch version of the distributed Krasulina's method (DM-Krasulina) along with its theoretical guarantees. Next, to account for data losses due to practical constraints in distributed systems (e.g., insufficient communication or processing resources) theoretical results for DM-Krasulina are further extended to show that as long as we choose mini-batch size, B, appropriately DM-Krasulina will asymptotically achieve the near optimal rate. Finally, experiments are performed over synthetic and real-world data to show the improvements achieved by the proposed DM-Krasulina algorithm in high-rate streaming settings.

2.1 Introduction

Dimensionality reduction and feature learning methods like principal component analysis (PCA), sparse PCA, linear discriminant analysis, and dictionary learning, form an important component of any machine learning pipeline. For data lying in a d-dimensional space, dimensionality reduction methods try to find the $k \ll d$ variables/features that are most relevant for solving an application specific task (e.g., classification, inference, estimation, data compression, etc.). The focus of this work is on principal component analysis (PCA), where the objective is to compute k-features that capture most of the variance in data. The proliferation of *big data* (both in terms of dimensionality and number of samples) has resulted in an increased interest in developing new algorithms for solving PCA due to the fact that classical numerical solutions (e.g., power iteration and Lanczos method [31]) for computing eigenvectors of symmetric matrices do not scale well with high dimensionality and large sample sizes. The main goal here is to develop algorithms that are cheap in terms of both memory and computational requirements as a function of dimensionality and number of data samples.

In addition to high dimensionality and large number of samples, data is often *streaming at a high rate*. Example applications include the Internet of things (IoT), video surveillance, autonomous vehicles, social media websites, news websites, and weather data. Stochastic methods have been developed to solve PCA in such streaming settings [32–35]. These methods work under the condition that data arrival rate is slow enough such that each data sample can be processed before the arrival of the next data sample. This may not always hold for modern day applications mentioned previously. To overcome this obstacle, this chapter proposes a distributed and distributed mini-batch variants of the Krasulina's method [33]. Before providing more details of the proposed methods, we first briefly review the existing algorithms for solving the streaming PCA problem.

2.1.1 Principal component analysis (PCA) from streaming data

For data lying in $\mathbb{R}^d$, PCA learns a k-dimensional subspace with maximum data variance. Let $\mathbf{x} \in \mathbb{R}^d$ be a random vector that is drawn from some unknown distribution $\mathcal{P}_x$ with zero mean and $\mathbf{\Sigma}$ covariance matrix. For a constraint set $\boldsymbol{\mathcal{V}} := \{\mathbf{V} \in \mathbb{R}^{d\times k} : \ \mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}\}$, we can pose PCA as the following optimization problem:

$$\mathbf{Q} := \underset{\mathbf{V}\in\boldsymbol{\mathcal{V}}}{\arg\max}\, \mathbb{E}_{\mathcal{P}_x}\Big\{\mathbf{Tr}(\mathbf{V}^{\mathrm{T}}\mathbf{x}\mathbf{x}^{\mathrm{T}}\mathbf{V})\Big\}, \tag{2.1}$$

here $\mathbf{Tr}(.)$ is the trace operator. The solution for (2.1) is the matrix $\mathbf{Q}$ with top k eigenvectors of $\mathbf{\Sigma}$. Note here that (2.1) cannot be solved in its current form since $\mathcal{P}_x$ is unknown. If we have T data samples, $\{\mathbf{x}_t\}_{t=1}^{T}$, drawn independently from $\mathcal{P}_x$ we can accumulate these data samples to calculate the empirical covariance matrix

$$\bar{\mathbf{A}} := \frac{1}{T}\sum_{t=1}^{T}\mathbf{A}_t, \tag{2.2}$$

where $\mathbf{A}_t := \mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}$. Now, instead of solving (2.1) we can solve an empirical risk maximization problem

$$\mathbf{Q} := \underset{\mathbf{V}\in\boldsymbol{\mathcal{V}}}{\arg\max}\, \frac{1}{T}\sum_{t=1}^{T}\mathbf{Tr}(\mathbf{V}^{\mathrm{T}}\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}\mathbf{V}). \tag{2.3}$$

We can solve (2.3) by computing the singular value decomposition (SVD) of the empirical covariance matrix $\bar{\mathbf{A}}$ by solving a linear system of d equations, which is a computationally intensive task that requires $O(d^3)$ multiplications and has a memory overhead of $O(d^2)$. This is not suitable for high dimensional settings where the goal is to have $O(d^2k)$ computational complexity and $O(dk)$ memory complexity.

More efficient and hence popular approaches use methods like the power iteration or Lanczos method [31, Chapter 8]. Although these methods improve overall computational complexity to $O(d^2k)$ but they still have memory requirements on the order of $O(d^2)$. In addition, these batch methods require computing the empirical covariance matrix $\bar{\mathbf{A}}$, which results in $O(d^2T)$ multiplication operations. Furthermore, in streaming settings where the goal is real-time decision making from data, it is infeasible to compute $\bar{\mathbf{A}}$. In such a situation, stochastic methods like Krasulina's method [33] and Oja's rule [32] can be used. These are both simple and computationally efficient algorithms for computing the principal eigenvector (i.e., $k = 1$) of a covariance matrix in streaming settings while achieving $O(d^2)$ computational and $O(dk)$ memory complexity. Recent years have seen an increased popularity of these algorithms and we will highlight these recent advances in Section 3.1.2. For step size γ_t, Krasulina's method for computing the top eigenvector is given by the following equation[1]:

$$\mathbf{v}_t = \mathbf{v}_{t-1} + \gamma_t \left(\mathbf{A}_t \mathbf{v}_{t-1} - \frac{\mathbf{v}_{t-1}^{\mathrm{T}} \mathbf{A}_t \mathbf{v}_{t-1} \mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2^2} \right). \tag{2.4}$$

We can also interpret Krasulina's method as a solution to an optimization problem . Using Courant-Fischer Minimax Theorem [31, Theorem 8.1.2] the top eigenvector computation (1-PCA) can be posed as the following optimization problem ($k = 1$ version of (2.1)):

$$\mathbf{q}_1 := \arg\min_{\mathbf{v} \in \mathbb{R}^d} f(\mathbf{v}) = \arg\min_{\mathbf{v} \in \mathbb{R}^d} \frac{-\mathbf{v}^{\mathrm{T}} \mathbf{A}_t \mathbf{v}}{\|\mathbf{v}\|_2^2} \tag{2.5}$$

where,

$$\nabla f(\mathbf{v}) = \frac{1}{\|\mathbf{v}\|_2^2} \left(\mathbf{A}_t \mathbf{v} - \frac{(\mathbf{v}^{\mathrm{T}} \mathbf{A}_t \mathbf{v})\mathbf{v}}{\|\mathbf{v}\|_2^2} \right). \tag{2.6}$$

Comparing (2.6) with (2.4) we can see that (2.4) is very similar to applying stochastic gradient descent (SGD) to (2.5) with only difference being the scaling factor of $1/\|\mathbf{v}\|_2^2$. Since (2.5) is a nonconvex problem general guarantees of SGD for convex problems do not apply here [24, 36, 37]. Despite its nonconvexity, (2.5) is a well structured problem and hence provably efficient methods have been devised to solve it . Recent years have seen an increased activity towards developing these provably efficient methods by exploiting the structure in nonconvex problems. Some of the examples include PCA, dictionary learning, autoencoders, matrix completion, etc. Solving the nonconvex optimization problem in (2.5) for high-rate streaming settings is the goal of this chapter.

Remark 1. In this chapter we are only computing the top eigenvector of a covariance matrix (solving 1-PCA). As explained in [38, Section 1] one way to extend Krasulina's method for $k > 1$ case is to solve 1-PCA problem k times but it is not efficient in terms of sampling complexity. It is our future goal to provide an algorithm that computes k-PCA directly for high-rate streaming settings.

2.1.2 Our Contributions

This chapter proposes algorithms for computing the top eigenvector of a covariance matrix using Krasulina's method for high rate streaming settings. We then provide theoretical and empirical results to show the efficacy of the proposed methods. Theoretically our goal is to show that

[1]Oja's method is very similar to Krasulina's method and its iterate is given as follows:

$$\mathbf{v}_t = \mathbf{v}_{t-1} + \gamma_t \left(\mathbf{A}_t \mathbf{v}_{t-1} - \mathbf{v}_{t-1}^{\mathrm{T}} \mathbf{A}_t \mathbf{v}_{t-1} \mathbf{v}_{t-1} \right).$$

optimal convergence rate of $O(1/t)$ can be achieved after using t samples (see [39, Theorem 1.1] and [38, Theorem 6]) . Following are the key contributions of this chapter:

1. For a network consisting of N processing nodes we propose a distributed variant of Krasulina's method which is given in Algorithm 1. We then analytically show that the proposed distributed method will have an improved convergence rate on the order of $O(1/Nt)$ (Theorem 1) as compared to the $O(1/t)$ rate for the classical Krasulina's method.

2. When implementing stochastic methods mini-batching has advantages from implementation point of view as compared to a sample-by-sample approach [24]. Following the similar reasoning we then propose a distributed mini-batch variant of Krasulina's method (DM-Krasulina) that is given in Algorithm 2. For a network of N nodes performing mini-batching with a batch of size B/N at node i we have a network-wide batch of size B. We then analytically show that for this mini-batch setup similar to D-Krasulina method we will see an improvement in convergence rate by a factor of B, i.e., $O(1/Bt)$ rate is attained.

3. Further, consider a situation where due to resource constraints (i.e., computational or communication) we receive more data samples than we can process at any iteration t of Algorithm 2. For these resource constrained settings, let μ be the number of samples we lose at each iteration of the Algorithm 2 across the network. In such situations we show that if $\mu = O(B)$, then we can still achieve near optimal convergence rate as shown in Corollary 2. In addition to this, for non-streaming data settings, i.e., fixed number of data samples T we provide Corollary 2 which gives an upper bound of $B = O(T^{1-2/c_0})$ on mini-batch size to achieve the optimal rate, here $c_0 > 2$ is some constant.

4. Finally, we provide simulations using synthetic and real world data to show that mini-batching improves performance of Krasulina's method up to a certain point as we keep increasing batch size B. Furthermore, using simulations on synthetic and real world data we show the impact of having $\mu > 0$ on the convergence of Algorithm 2 as well.

2.1.3 Related Work

Solving PCA efficiently has been an active area of research for decades. Krasulina's method [33] and Oja's rule [32] are among the earliest and popular methods to solve PCA in streaming data settings. Due to effectiveness of these mathods their variants have been proposed over the years [40–42]. Like earlier developments in stochastic approximation methods [43], these algorithms were shown to converge asymptotically. Convergence rate results in stochastic optimization [44, 45] for finite sample settings paved the way to finding the non-asymptotic convergence rate of stochastic PCA, which is a nonconvex optimization problem. In the following we will first provide an overview of advancements in solving PCA in general and in the end we will talk about solving PCA in streaming and distributed data settings which is the main theme of this chapter.

Convex relaxation for solving PCA. One approach to answer this question is to relax PCA as a convex optimization problem and use SGD analysis to give the rates of convexified problems [7, 34, 46, 47]. The penalty we pay for convexification is that we need an iterate matrix of dimensions $\mathbb{R}^{d\times d}$ in contrast to $\mathbb{R}^{d\times k}$ if we solve problem in its original nonconvex form. Due to this reason, solving PCA in nonconvex form is preferable.

Nonconvex optimization based solutions. Examples of solving PCA as a nonconvex optimization problem include works by Zhang and Balzano [35] and De Sa, Olukotun, and Ré [48]. Among these, work by Zhang and Balzano [35] solve an optimization problem over a nonconvex Grassmanian maifold and their analysis requires a good initial guess to work. Contribution by De Sa, Olukotun, and Ré [48] use SGD to solve some nonconvex problems including PCA. Their analysis uses constant step size and it needs to be sufficiently small in order for method to converge which results in longer runtime in practice.

Algebraic approaches. By algebraic approaches we mean algorithms, [38–40, 49–52], that are motivated by classical methods for online PCA like Oja's rule [32] and Krasulina's method [33]. Among these, methods proposed by Shamir [50, 51] use variance reduction techniques to speed-up the convergence, but these algorithms require multiple passes over data and their analyses require initial value of the iterate to be close to the original subspace which, with high probability, does not happen in practice. Allen-Zhu and Li [38] provide eigengap-free results for Oja's rule but these results do not take sampling variance into account and hence the extension to mini-batch settings is not trivial. Among the above mentioned contributions, Jain et al. [39] provide variance based guarantees for Oja's rule but there results only hold with probability of success 3/4. One possible way to get around this limitation is to run algorithm multiple times which is not a feasible option in streaming settings.

Distributed online mini-batch PCA. In recent years, efforts have been made towards solving PCA in distributed settings [53–56]. The goal in some of these works has been towards improving the communication efficiency of distributed methods for PCA [53–55]. Note that Balcan et al. [53] and Boutsidis, Woodruff, and Zhong [55] achieve this by applying compression on iterates before communicating with the server node. On the other hand, Garber, Shamir, and

Srebro [54] computes the top eigenvector of local covariances at each node and then add all these local eigenvectors in the last iteration to compute the final estimate of eigenvector. Hence, this way they only need a single round of communication at the last iteration of algorithm. Since, we add up all the local estimates in the last iteration of algorithm the approach taken by Garber, Shamir, and Srebro [54] is well suited for batch data but the extension to streaming settings is not obvious. In contrast, goal in this chapter is to show that distributed mini-batch extensions of traditional methods, like Oja's rule and Krasulina's method, for solving streaming PCA problem results in improved convergence rates. In this regard our work is more closely related to the work in [56]; which uses momentum method to accelerate convergence of power method and further extend this method to stochastic settings. This method relies on a variance reduction technique that requires a pass over the complete dataset once in a while, which is not possible in streaming settings. Furthermore, theoretical guarantees provided by Xu et al. [56] require a good initialization to work and they also do not consider issues arising as a result of communication delay while averaging over a mini-batch. This chapter addresses some of these issues while using mini-batching to speed-up the convergence of Krasulina's method.

Nonconvex stochastic optimization. Another relevant research thrust to streaming PCA is the use of stochastic methods to solve general noncovex problems. In this regard, stochastic gradient descent (SGD) has been shown to work well for solving nonconvex optimization problems [57–59]. In addition to simple SGD, variance reduction variants of SGD also exist for noncovex problems [60, 61]. Furthermore, accelerated methods for stochastic optimization have also been proposed recently [62, 63]. Among these works, contributions by Reddi et al. [60] and Allen-Zhu and Hazan [64] are most relevant to the current work since they use mini-batching framework to reduce variance in estimates which in return improves the performance of SGD, which is the main algorithmic ingredient of this chapter as well. Our work in this chapter differs from these contributions as it provides a mini-batch approach to compute the global minimum of a specific problem i.e., PCA, in contrast to convergence to the first order stationary points of a more general nonconvex optimization problems in above mentioned works.

2.1.4 Chapter Organization

In the following we first provide problem setup in Section 2.2, then in Section 2.3 we propose distributed variant of Krasulina's method (Algorithm 1) and a distributed mini-batch variant as well (Algorithm 2). Then in Section 2.4 we provide theoretical guarantees for the proposed methods and in Section 2.5 we provide proofs of our main theoretical results while the proof of supporting lemmas are given in Sections 2.6, 2.7, and 2.8. Finally, we provide simulation results in Section 2.10.

Notation. We use bold face small case letters, $\mathbf{a}$, to represent vectors, boldface capital letters, $\mathbf{A}$, to represent matrices, and plain small case letters, a, to represent scalars. We also use plain small case letters to represent elements of a set of all possible realizations of a distribution. $\lceil . \rceil$ is used to define a ceiling operation over a scalar value. The spectral norm of a matrix is denoted by $\|.\|_2$ and Frobenius norm is denoted by $\|.\|_F$. For a positive definite matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ we have $\|\mathbf{A}\|_2 = \lambda_1(\mathbf{A}) \leq \sqrt{\sum_{i=1}^{d} \lambda_i^2(\mathbf{A})} = \|\mathbf{A}\|_F$, we will be using this inequality in our proofs.

2.2 Problem Setup

2.2.1 Data model

In this chapter we consider a streaming setup where at any given time t we observe an independent and identically distributed sample $\mathbf{x}_t \in \mathbb{R}^d$ from an unknown distribution $\mathcal{P}_x$. We require following assumptions on distribution $\mathcal{P}_x$ to hold for our convergence analysis.

[**A1**] *(Zero mean bounded input)* At any time t we observe a zero-mean bounded random vector $\mathbf{x}_t \in \mathbb{R}^d$, such that, $\mathbb{E}\{\mathbf{x}_t\} = 0$ and $\|\mathbf{x}_t\|_2 \leq r$ (let $r \geq 1$ without loss of generality).

[**A2**] *(Spectral gap of the covariance matrix)* We assume that the largest eigenvalue of $\boldsymbol{\Sigma} := \mathbb{E}\{\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}\}$ is strictly greater than the second largest eigenvalue, i.e., $\lambda_1 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_d$.

Assumptions [**A1**] and [**A2**] are the standard assumptions for proving convergence of Krasulina's method and Oja's rule (cf. [32, 38, 39, 65]). From these samples, at any time t we can compute an estimate, $\mathbf{A}_t := \mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}$, of the population covariance matrix $\boldsymbol{\Sigma}$. We define the variance in the estimate, $\mathbf{A}_t$, of the covariance matrix $\boldsymbol{\Sigma}$ as follows:

Definition 1. *(Variance of sample covariance matrix)* We define the variance in sample covariance matrix $\mathbf{A}_t := \mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}$ as follows:

$$\sigma^2 := \mathbb{E}\big\|\mathbf{A}_t - \boldsymbol{\Sigma}\big\|_F^2.$$

Since we are assuming bounded random vectors in Assumption [**A1**] all the moments of the probability distribution $\mathcal{P}_x$ exist and hence variance as defined above exists as well. In this chapter we initialize Krasulina's method with $\mathbf{v}_0 \in \mathbb{R}^d$ in a uniform random fashion over a unit sphere in $\mathbb{R}^d$. Based on random initialization $\mathbf{v}_0$ and a stream of samples $\mathbf{x}_1, \mathbf{x}_2, \ldots$ we define Ω

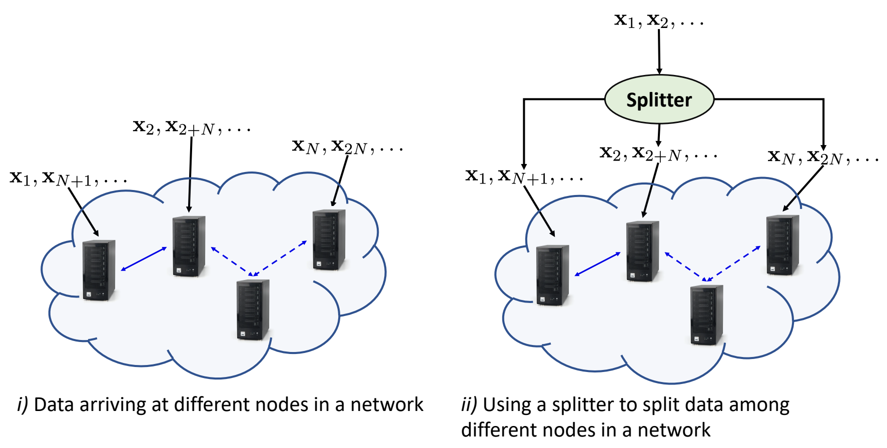

$\mathbf{x}_1, \mathbf{x}_{N+1}, \ldots$
$\mathbf{x}_2, \mathbf{x}_{2+N}, \ldots$
$\mathbf{x}_N, \mathbf{x}_{2N}, \ldots$
i) Data arriving at different nodes in a network
$\mathbf{x}_1, \mathbf{x}_2, \ldots$
Splitter
$\mathbf{x}_1, \mathbf{x}_{N+1}, \ldots$
$\mathbf{x}_2, \mathbf{x}_{2+N}, \ldots$
$\mathbf{x}_N, \mathbf{x}_{2N}, \ldots$
ii) Using a splitter to split data among different nodes in a network

Figure 2.1: In *i)* data is arriving at N different nodes in a network at the rate of $R_S = R_P$ samples per second while in *ii)* a single stream of data is arriving at the rate of $R_S = N \times R_P$ samples per second and we are splitting a single stream into N streams at the splitter. In case *ii)* here we are assuming a buffer of N samples at the splitter which distribute samples across N nodes as soon as N samples become available.

to be the *sample space* of all the possible outcomes $(\mathbf{v}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ and $\mathbf{Pr}$ to be the probability distribution over all the events in the sample space Ω.

2.2.2 High-rate streaming settings

Let R_S be the number of samples, $\mathbf{x}_t$, arriving every second and R_P be the number of samples we can process in a second. If $R_S > R_P$ we will not be able to make an update in iterate (2.4) for every sample in the stream, we term this regime as the *high-rate* streaming settings. In these settings if $R_S = \alpha R_P$ for $\alpha > 1$ then we can only use $1/\alpha$ of total samples to update iterate (2.4). Now assume that we have $N \geq \lceil \alpha \rceil$ processors at our disposal with each having processing capacity of R_P samples per second. Under this assumption one possible solution to this problem is to deploy all these processors and hence using all the data samples. In the following we describe two possible ways for learning the top eigenvector from the sample data in *high rate streaming* settings.

Distributed processing over a network

We consider a connected network of N nodes. For a connected network, if the nodes in the network have locally available vector values $\{\mathbf{a}_i\}_{i=1}^N$, then we can devise a method to compute the summation $\sum_{i=1}^N \mathbf{a}_i$. We now define $R_C = 1/T_C$, where T_C is the total time spent in finishing the summation operation across the network. A connected network on which we can perform summation operation is helpful since Krasulina's method can be written down as a summation of local iterates when each node, i, is observing a separate and independent stream of data samples, $\{\mathbf{A}_{i,t} := \mathbf{x}_{i,t}\mathbf{x}_{i,t}^{\mathrm{T}}\}_{i=1}^{N}$, from the same distribution $\mathcal{P}_x$. As an example, such a summation operation across the network can be accomplished by having a fully connected network topology, a star topology (where one central node accumulates data from all the other nodes), or generating a tree structure where starting from leaf nodes each node passes its average value to nodes above it in the hierarchy until we reach the root node and then we can propagate this average value from root to rest of the network.

In case of high rate streaming data (i.e., $R_S > R_P$), under the assumption that $N \geq \lceil \alpha \rceil$ we can distribute processing across these nodes and hence process all the samples to make an update. This distributed setting can be an outcome of one of the following two possible scenarios for data arrival: *i)* multiple streams with sample arrival rate of R_S/N samples per second arriving at N nodes in the network, or *ii)* a single stream with the data arrival rate of R_S is arriving at the load balancer/splitter which buffers N samples and then divide them across the N processors as shown in Figure 2.1. For such a distributed network of N nodes we propose a distributed variant of Krasulina's method in Section 2.3 and then we provide convergence guarantees of the proposed method in Section 2.4.

Mini-batching in a distributed network

Mini-batching reduces the number of I/O operations per iteration as compared to sample-by-sample computation in stochastic algorithms and hence results in an improved performance in practice (e.g., [24, 25]). Mini-batching can be seen as a bridge between a batch method (e.g., gradient descent) and its incremental variants (e.g., stochastic gradient descent (SGD)). In a distributed network of N nodes let $\{\mathbf{A}_{i,j,t} := \mathbf{x}_{i,j,t}\mathbf{x}_{i,j,t}^{\mathrm{T}}\}_{j=1}^{B/N}$ be the mini-batch of size B/N arriving at a node i at time t (note that total mini batch size across the network is B). For the distributed mini-batching setup described here we propose a variant of Krasulina's method which is given in Algortihm 2.

2.3 Solutions for PCA from high-rate streaming data

In the following we provide solutions for two problem setups described in Section 2.2.

2.3.1 Distributed Krasulina's method

Recall that $\{\mathbf{A}_{i,t}\}_{i=1}^{N}$ is the sequence of sample covariance matrices received across a network at time t. Starting from the same initial estimate of eigenvector $\mathbf{v}_0$ at each node, the distributed

Algorithm 1: Distributed Krasulina's method for PCA.

```
Input: Streaming data $\mathbf{x}_1, \mathbf{x}_2, \ldots$.
Initialize: Randomly generate $\mathbf{v}_0$.
1: for $t = 1, 2, \ldots$ do
2:     Receive sample covariance $\mathbf{A}_{i,t} = \mathbf{x}_{i,t}\mathbf{x}_{i,t}^{\mathrm{T}}$ and update $\boldsymbol{\xi}_{i,t}$ locally at each node as follows
           $\boldsymbol{\xi}_{i,t} = \mathbf{A}_{i,t}\mathbf{v}_{t-1} - \frac{\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{A}_{i,t}\mathbf{v}_{t-1}\mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2^2}$
3:     Distributed vector sum computation routine to compute the average $\boldsymbol{\xi}_t \leftarrow \sum_{i=1}^{N} \boldsymbol{\xi}_{i,t}$
4:     Update eigenvector estimate: $\mathbf{v}_t \leftarrow \mathbf{v}_{t-1} + \frac{\gamma_t}{N}\boldsymbol{\xi}_t$
5: end for
Return: Top eigenvector $\mathbf{v}_T$.
```

variant of Krasulina's method can be written as follows:

$$\begin{aligned}\mathbf{v}_t &= \mathbf{v}_{t-1} + \gamma_t\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{A}_{i,t}\mathbf{v}_{t-1} - \frac{1}{\|\mathbf{v}_{t-1}\|_2^2}\left(\mathbf{v}_{t-1}^{\mathrm{T}}\frac{1}{N}\sum_{i=1}^{N}\mathbf{A}_{i,t}\mathbf{v}_{t-1}\mathbf{v}_{t-1}\right)\right)\\ &= \mathbf{v}_{t-1} + \gamma_t\boldsymbol{\xi}_t. \qquad (2.7)\end{aligned}$$

Note here that in contrast to the classical Krasulina' s method where we use t samples after t iterations of algorithm here we have used Nt samples to build our estimate $\mathbf{v}_t$. This difference poses a question that whether both the algorithms will have similar performance in terms of sample complexity. Specifically, after t iterations the distributed Krasulina's method (Algorithm 1) has used Nt samples and hence one should expect an improved convergence rate i.e., on the order of $O(1/Nt)$. One of the main contribution of this chapter is to show that we indeed achieve linear improvement in convergence as a function of number of nodes N in the network. This is result is given in more detail in Theorem 1. We can view Krasulina's method in distributed settings as performing classical Krasulina's method over an average of N sample covariance matrices $\{\mathbf{A}_{i,t} := \mathbf{x}_{i,t}\mathbf{x}_{i,t}^{\mathrm{T}}\}_{i=1}^{N}$. For this setup we can define the variance in covariance matrix estimates as follows:

Definition 2. *(Variance of sample covariance in distributed settings)* We define the variance in sample covariance matrix $\mathbf{A}_t := \mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}$ as follows:

$$\sigma_N^2 := \mathbb{E}\Big\|\frac{1}{N}\sum_{i=1}^{N}\mathbf{A}_{i,t} - \boldsymbol{\Sigma}\Big\|_F^2.$$

2.3.2 Distributed Krasulina's method using mini batches

As described in Section 2.4.2, in distributed mini-batch settings each node i receives a mini-batch, $\{\mathbf{A}_{i,j,t} := \mathbf{x}_{i,j,t}\mathbf{x}_{i,j,t}^{\mathrm{T}}\}_{j=1}^{B/N}$, of size B/N and computes $\boldsymbol{\xi}_{i,t}$ locally

$$\boldsymbol{\xi}_{i,t} = \sum_{j=1}^{B/N}\left(\mathbf{A}_{i,j,t}\mathbf{v}_{t-1} - \frac{\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{A}_{i,j,t}\mathbf{v}_{t-1}\mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2^2}\right).$$

As shown in Step 7 of Algorithm 2, we sum up all these local estimates $\boldsymbol{\xi}_{i,t}$ and get the update of distributed mini-batch Krasulina's method as

$$\mathbf{v}_t = \mathbf{v}_{t-1} + \frac{\gamma_t}{B}\sum_{i=1}^{N}\boldsymbol{\xi}_{i,t}. \tag{2.8}$$

In addition to the practical gains from mini batching as described in Section 2.2.2 it also provides a flexibility to analyze the impacts due to practical situations like delays in the network. Next we describe two possible situations that can occur when implementing Algorithm 2 in practice.

- *Case 1 ($R_S \leq R_P + R_C$):* As long as this stays true we can process all the samples arriving across the network. Convergence analysis of this case shows that we will get an optimal rate of $O(1/Bt)$ after t iterations of Algorithm 2.

- *Case 2 ($R_S > R_P + R_C$):* In this scenario since data arrival rate is faster than what the computation and communication resources can handle we will have to discard some samples at each update. We define μ be the total number of data samples that are being discarded across the network during each iteration of Algorithm 1. Our analysis shows that as long as $\mu = O(B)$ we can still achieve an optimal rate asymptotically.

Our theoretical results for both the cases rely on variance reduction in the sample covariance matrix due the averaging operation in Steps 7–8 of Algorithm 2. We formally define the variance in sample covariance matrix of mini-batch size B in the following.

Definition 3. *(Variance of sample covariance matrix for distributed minibatch settings)* We define the variance in sample covariance matrix $\mathbf{A}_t := \frac{1}{B}\sum_{i=1}^{N}\sum_{j=1}^{B_i}\mathbf{A}_{i,j,t}$ as follows:

$$\sigma_B^2 := \mathbb{E}\left\|\frac{1}{B}\sum_{i=1}^{N}\sum_{j=1}^{B_i}\mathbf{A}_{i,j,t} - \boldsymbol{\Sigma}\right\|_F^2.$$

2.4 Convergence Analysis

Our convergence analysis is based on error measure Ψ_t. The goal here is to show that $\Psi_t \to 0$ as $t \to \infty$. We formally define Ψ_t in the following.

Algorithm 2: (DM-Krasulina) Distributed mini-batch Krasulina's method for PCA.

```
Input: Streaming data $\mathbf{x}_1, \mathbf{x}_2, \ldots$.
Initialize: Randomly generate $\mathbf{v}_0$.
1: for $t = 1, 2, \ldots$ do
2:   Initialize $\boldsymbol{\xi}_{i,t} \leftarrow 0$ at node $i$
3:   (At every node $i$ perform following computation over mini-batch of size $B_i$):
4:   for $j = 1, \ldots, B_i$ do
5:     Receive sample covariance $\mathbf{A}_{i,j,t} = \mathbf{x}_{i,j,t}\mathbf{x}_{i,j,t}^{\mathrm{T}}$ and update $\boldsymbol{\xi}_{i,t}$ locally at each node as
       follows
       $$\boldsymbol{\xi}_{i,t} = \boldsymbol{\xi}_{i,t} + \mathbf{A}_{j,t}\mathbf{v}_{t-1} - \frac{\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{A}_{j,t}\mathbf{v}_{t-1}\mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2^2}$$
6:   end for
7:   Call distributed vector sum computation routine to compute the sum $\boldsymbol{\xi}_t \leftarrow \sum_{i=1}^{N} \boldsymbol{\xi}_{i,t}$, in
     the meantime receive $\mu \geq 0$ new samples across the network which will not contribute
     towards the estimate
8:   Update eigenvector estimate: $\mathbf{v}_t \leftarrow \mathbf{v}_{t-1} + \frac{\gamma_t}{B}\boldsymbol{\xi}_t$
9: end for
Return: Top eigenvector $\mathbf{v}_T$.
```

Definition 4. (Estimation error) Let $\mathbf{q}_1$ be the principal eigenvector of $\mathbf{\Sigma}$ and $\mathbf{v}_t$ be the output of Algorithm 1 (or Algorithm 2) at iteration t, then we can define Ψ_t, the error in eigenvector estimate, in the form of a potential function as

$$\Psi_t := 1 - \frac{(\mathbf{v}_t^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_t\|^2}. \tag{2.9}$$

Using this error measure is a common practice when arguing about PCA. It essentially computes sine squared of the angle between the true eignevector and its estimate. If we initialize Algorithm 1 (or Algorithm 2) with a vector $\mathbf{v}_0$ randomly over a unit sphere in $\mathbb{R}^d$ then it can be shown that $\mathbb{E}\Psi_0 \leq 1 - 1/d$ [65]. Since the preceding statement only tells us about the expected behavior, we first need to establish that using Algorithm 1 (or Algorithm 2) we can make a much stronger statement about $\mathbf{v}_t$, i.e., with high probability. Our first contribution here will be in showing that for $t > 0$ if we choose step size of the form $\gamma_t = c/(L + t)$ with large enough value of L then with high probability we have $\Psi_t \leq 1 - O(1/d)$ (Theorem 3). In order to prove this result we first define a filtration $\mathcal{F}_{t-1}$ over samples observed from the sample space Ω that captures the progress of the iterate of Algorithm 1 (or Algorithm 2) upto iteration $t - 1$. In the following we define sigma algebra in terms of $\mathbf{A}_t$ for both Algorithm 1 and Algorithm 2 at iteration t of the respective algorithm. Recall that $\mathbf{A}_t$ for Algorithm 1 is defined as $\mathbf{A}_t = \frac{1}{N}\sum_{i=1}^{N} \mathbf{A}_{i,t}$ and for Algorithm 2 $\mathbf{A}_t = \frac{1}{B}\sum_{i=1}^{N}\sum_{j=1}^{B/N} \mathbf{A}_{i,j,t}$. For the ease of exposition, from this point onwards we are will be providing analysis in terms of random variable $\mathbf{A}_t$ instead of $\mathbf{x}_t$.

Definition 5. (σ-algebra $\mathcal{F}_t$) Starting from initial guess $\mathbf{v}_0$ of eigenvector we update eigenvector estimate based on covariance matrix estimates $\mathbf{A}_1, \ldots, \mathbf{A}_t$ until iteration t of Algorithm 1 (or Algorithm 2), we define $\mathcal{F}_t$ to be the σ-field of all these outcomes, here $(\mathbf{v}_0, \mathbf{A}_1, \ldots, \mathbf{A}_t)$ is one particular realization of the sample space Ω_t:

$$\mathcal{F}_t := \sigma(\mathbf{v}_0, \mathbf{A}_1, \ldots, \mathbf{A}_t).$$

In addition to the filtration over all the observed samples we also need to define a sequence of nested sample spaces which are subsets of the complete sample space Ω.

Definition 6. (Nested sample spaces) We define a nested sequence of sample spaces $\Omega \supset \Omega_0^{'} \supset \Omega_1^{'} \supset \ldots$ such that each $\Omega_t^{'}$ is $\mathcal{F}_{t-1}$-measurable and for $\epsilon_j > 0$:

$$\Omega_t^{'} := \{\omega \in \Omega : \sup_{t_j \leq l < t} \Psi_l(\omega) \leq 1 - \epsilon_j, \quad \forall \quad 0 \leq j \leq J\}. \tag{2.10}$$

Here, $\Psi_l(\omega)$ is estimation error for realization ω at iteration l of Algorithm 1 (or Algorithm 2). Furthermore, we define $\mathbf{Pr}_t$ to be the probability distribution and $\mathbb{E}_t$ to be the expectation over sample space $\Omega_t^{'}$. Formally, $\mathbb{E}_t$ can be defined as follows:

$$\mathbb{E}_t f = \frac{1}{\mathbf{Pr}(\Omega_t^{'})} \int_{\Omega_t^{'}} f(\omega) \mathbf{Pr}(d\omega). \tag{2.11}$$

In Definition 6, $\Omega_{t_0}^{'}$ is the subset of sample space at the initial iteration (i.e., $t_0 = 0$) and subsequent sample spaces $\Omega_l^{'}$ with $t_j \leq l \leq t$ correspond to the event when roughly speaking, the error is below $1 - \epsilon_j$ as defined in (2.10). An important implication of this definition is that we are restricting ourselves to such subsets of sample space Ω which ensure the convergence of the iterate of Algorithm 1 (or Algorithm 2) to the top eigenvector of the population covariance matrix, $\boldsymbol{\Sigma}$, at a linear rate. One main challenge here is to show that such subsets, $\Omega_t^{'}$, of the sample space occur with high probability. Specifically, we need to show that for any $\delta > 0$, we have $\mathbf{Pr}(\Omega_t^{'}) \geq 1 - \delta$ (Theorem 4).

2.4.1 Convergence of distributed Krasulina's method (Algorithm 1)

Our first convergence result of this chapter shows that for a distributed network of N nodes $\Psi_t \to 0$ as $t \to \infty$ on the order of $O(1/Nt)$ which means that Algorithm 1 results in an optimal sampling complexity for streaming PCA. The key to proving this improved convergence rate in distributed settings is the reduction in variance of covariance matrix estimate by a factor of N due to averaging of the iterate (Step 4 in Algorithm 1). For σ_N^2 as defined in Definition 2 we can upper bound variance of the average estimate of the covariance matrix across network in terms of variance of a single sample as, $\sigma_N^2 \leq \sigma^2/N$. The main theoretical result regarding distributed Krasulina's method is stated in the following theorem.

Theorem 1. *For some $c_0 > 2$, pick $0 < \delta < 1$ and $c = c_0/2(\lambda_1 - \lambda_2)$. Define*

$$L_1 := \frac{64edr^4 \max(1, c^2)}{\delta^2} \ln \frac{4}{\delta}, \quad L_2 := \frac{512e^2d^2\sigma_N^2 \max(1, c^2)}{\delta^4} \ln \frac{4}{\delta} \tag{2.12}$$

and set: $L \geq L_1 + L_2$. For $\phi := \lambda_1^2 + \lambda_2^2$ define constants

$$C_1 := \frac{1}{2}\left(\frac{4ed}{\delta^2}\right)^{\frac{5}{2\ln 2}} e^{c^2\phi/L} \quad \text{and} \quad C_2 := \frac{2c^2 e^{(c_0 + c^2\phi)/L}}{(c_0 - 2)}$$

*then, if Assumptions [**A1**] and [**A2**] hold and we choose step size as $\gamma_t = c/(L+t)$, then there is a nested sequence of subsets of the sample space $\Omega \supset \Omega_0^{'} \supset \Omega_1^{'} \supset \dots$ such that*

$$\mathbf{Pr}(\Omega_t^{'}) \geq 1 - \delta, \ \textit{and}$$

$$\mathbb{E}_t\{\Psi_t\} \leq C_1\Big(\frac{L+1}{L+t+1}\Big)^{\frac{c_0}{2}} + C_2\Big(\frac{\sigma_N^2}{t+L+1}\Big), \tag{2.13}$$

where $\mathbb{E}_t$ is expectation restricted to the sample space $\Omega_t^{'}$ (Definition 6).

In the following we explain the result in Theorem 1 and the proof is moved to Section 2.5. Our aim here is to explain the dependence of convergence properties of Algorithm 1 on the parameters of interest in solving (2.3). The important problem parameters while solving the PCA are: *i)* dimensionality of the ambient space, d, *ii)* the eigengap of the population covariance matrix, $(\lambda_1 - \lambda_2)$, *iii)* variance in sample covariance matrix, σ_N^2, and *iv)* the upper bound on received data samples, r. Since we followed the proof technique devised by Balsubramani, Dasgupta and Freund [65], therefore our results have a similar form. But, despite the apparent similarities our result differs in terms of dependence on the problem parameters described above. Specifically, the result by Balsubramani, Dasgupta and Freund [65] is independent of the variance, σ_N^2, in the sample covariance matrix which is an important problem parameter for stochastic and online methods. The result provided in [65] is based on an assumption that $\|\mathbf{x}_t\|_2 \leq r$ (Assumption [**A1**] here) while our analysis also takes into account the sample variance, σ_N^2, of the empirical covariance matrix $\mathbf{A}_t$. Due to this difference, result in Theorem 1 is on the order of $O(\sigma_N^2/t)$ in contrast to $O(r^4/t)$ result in [65]. This improvement is important since we can now decrease the error by using larger values of N. In addition, this also results in an improved lower bound on choice of L by splitting it into two quantities where the first term (L_1 in (2.12)) is on the order of $\Omega(r^4d/\delta^2)$ which is an improvement over $\Omega(r^4d^2/\delta^4)$ bound of [65]. Here, the second term in the lower bound of L (L_2 in (2.12)) has same dependence on d and $1/\delta$ as [65] but instead of r^4 it is being multiplied with σ_N^2 which we can always decrease by using a bigger batch size. The improved lower bound on L implies that we can use a larger value for the step size which also results in faster convergence. In terms of the eigengap, our result has optimal dependence of $1/(\lambda_1 - \lambda_2)^2$ which is the same as in [65]. While for dimensionality the dominant error term in (2.13) is independent of dimension but the lower bound on L has a quadratic dependence on d which results in a polynomial dependence (order 4 polynomial for $c_0 = 2$) on d for the non-dominant term in (2.13). As already explained, for large values of N the lower bound on L will depend linearly on d and hence the polynomial dependence on d can be improved e.g., for $c_0 = 2$ the non-dominant term in (2.13) will depend quadratically over d. Hence, our result improves upon the result of [65] in terms of the dependence on d as well. Despite this improvement, the dependence on d is still rather lose, as an example, the result in [39] has $\log^2(d)$ dependence in higher order error terms. We highlight these dependencies on problem parameters further with the help of simulations in Section 2.10.

Remark 2. Note that, same as in [65] for the case of $c_0 < 2$, we will get a rate of $O(t^{-c_0/2})$.

Remark 3. In this chapter we are only providing analysis for Krasulina's method, but as described by mBalsubramani, Dasgupta and Freund [65] we can analyze Oja's rule using similar techniques.

2.4.2 Mini-batching in Distributed Settings (Algorithm 2)

As explained in Section 2.3.2 we consider two situations for mini-batching in distributed settings. In first case we consider that $R_S \leq R_P + R_C$ hence no data loss occurs i.e., $\mu = 0$. While in second case $R_S > R_P + R_C$ and hence we will experience loss of $\mu > 0$ samples in each iteration of Algorithm 2. In the following we provide convergence rate analysis for both these cases.

Case 1: Mini-batching with no data loss ($R_S \leq R_P + R_C$ and $\mu = 0$)

This scenario is very similar to distributed Krasulina's method (Algorithm 1). The difference here is that instead of averaging over N samples with one sample at each node we are performing averaging over B samples with B_i samples at node i. We can bound the sample variance (Definition 3) of mini-batch method as $\sigma_B^2 \leq \sigma^2/B$ which gives us improvements in convergence rate. Using the same procedure as in Section 2.4.1 we get the following result for distributed mini-batch settings.

Theorem 2. *Under the same conditions as in Theorem 1, if there is a nested sequence of subsets of the sample space $\Omega \supset \Omega_0^{'} \supset \Omega_1^{'} \supset \dots$ then we have*

$$\mathbf{Pr}(\Omega_t^{'}) \geq 1 - \delta, \ \textit{and}$$

$$\mathbb{E}_t\{\Psi_t\} \leq C_1\Big(\frac{L+1}{L+t+1}\Big)^{\frac{c_0}{2}} + C_2\Big(\frac{\sigma_B^2}{t+1}\Big),$$

where $\mathbb{E}_t$ is expectation restricted to the sample space $\Omega_t^{'}$ (see (2.11)*) and $L \geq L_1 + L_2$ where L_1 and L_2 are given by* (2.12)*.*

Proof. The proof is the same as that of Theorem 1 by replacing number of nodes N by mini-batch size B. □

This shows that, same as in the case of distributed Krasulina's method with N nodes, in this case we will observe a linear improvement in convergence rate as a function of batch-size B. So far we have talked about applying Algorithm 1 and Algorithm 2 to the streaming setting. In practice these algorithms can also be applied when we have fixed number of samples T and T is large enough that applying a batch algorithm is not feasible. Next, we extend the result in Theorem 2 to show that Algorithm 2 can be used to solve problem (2.3) for some very large but fixed number of samples T. For such settings following result shows that if we choose mini-batch size B appropriately we can still achieve the optimal rate:

Corollary 1. *Under the conditions defined in Theorem 1 and if Assumptions* [**A1**]–[**A2**] *hold, then there is a nested sequence of subsets of the sample space $\Omega \supset \Omega_0^{'} \supset \Omega_1^{'} \supset \dots$ and if we choose $B = O(T^{1-2/c_0})$, then with:*

$$\mathbf{Pr}(\Omega_{T_B}^{'}) \geq 1 - \delta, \ \textit{and}$$

$$\mathbb{E}_{T_B}\{\Psi_{T_B}\} \leq \frac{c_0 C_1 L_1^{c_0/2}}{T} + c_0 C_1 \left(\frac{L_2}{T}\right)^{c_0/2} + \frac{2C_2\sigma^2}{T},$$

where $\mathbb{E}_{T_B}$ is as defined in (2.11) *and $L \geq L_1 + L_2$ where L_1 and L_2 are given by* (2.12)*.*

Proof. Substituting $t = T_B$ in result from Theorem 1 we get:

$$\begin{aligned}\mathbb{E}_{T_B}\{\Psi_{T_B}\} &\leq C_1\Big(\frac{L+1}{L+T_B}\Big)^{\frac{c_0}{2}} + C_2\Big(\frac{\sigma_B^2}{T_B}\Big)\\ &\leq 2C_1\Big(\frac{L}{T_B}\Big)^{\frac{c_0}{2}} + C_2\Big(\frac{\sigma_B^2}{T_B}\Big).\end{aligned}$$

Substituting $L = L_1 + L_2$ we get

$$\mathbb{E}_{T_B}\{\Psi_{T_B}\} \leq c_0 C_1\Big(\frac{L_1}{T_B}\Big)^{\frac{c_0}{2}} + c_0 C_1\Big(\frac{L_2}{T_B}\Big)^{\frac{c_0}{2}} + C_2\Big(\frac{\sigma_B^2}{T_B}\Big). \tag{2.14}$$

Let L_2 be the value selected for classical Krasulina's method then from lower bound on L_2 we can pick L_2/B for distributed mini-batch Krasulina's method (Algorithm 2). Now substituting L_2/B and $T_B = T/B$ gives us:

$$\mathbb{E}_{T_B}\{\Psi_{T_B}\} \leq c_0 C_1\left(\frac{BL_1}{T}\right)^{c_0/2} + c_0 C_1\left(\frac{L_2}{T}\right)^{c_0/2} + \frac{2C_2\sigma^2 B}{BT},$$

substituting $B = O(T^{1-2/c_0})$ in this inequality completes the proof. □

This result shows that for a fixed batch size optimal rate can be achieved as long as we choose mini-batch size appropriately. In Section 2.10 we corroborate this with numerical simulations over synthetic and real-world data as well.

Case 2: loss of samples due to resource constraints $\mu > 0$

In the following we consider streaming settings where we assume that we know in advance that we will be provided T data samples in a streaming fashion. The constraint here is that due to large volume of data we cannot store the data samples and we need to provide real-time estimate as soon as we received the T-th data sample (e.g., video surveillance over a short interval of time). In distributed settings, as described in Section 2.3.2 when due to resource constraints we need to discard μ samples per iteration of Algorithm 2 then after observing T samples this translates into performing $T_B := T/(B+\mu)$ iterations of the algorithm as compared to T/B iterations for $\mu = 0$ case. The convergence rate of Krasulina's method for such distributed settings is given by the following result.

Corollary 2. *Under the same conditions as in Theorem 1 and if Assumptions* [**A1**] *and* [**A2**] *hold, then there is a nested sequence of subsets of the sample space* $\Omega \supset \Omega_0^{'} \supset \Omega_1^{'} \supset \dots$ *such that*

$$\mathbf{Pr}(\Omega_{T_B}^{'}) \geq 1-\delta, \text{ and}$$

$$\mathbb{E}_{T_B}\{\Psi_{T_B}\} \leq c_0 C_1 \left(\frac{(B+\mu)L_1}{T}\right)^{c_0/2} + c_0 C_1 \left(\frac{(B+\mu)L_2}{BT}\right)^{c_0/2} + \frac{2C_2\sigma^2(B+\mu)}{BT}, \tag{2.15}$$

where $\mathbb{E}_{T_B}$ *is expectation restricted to the sample space* $\Omega_{T_B}^{'}$ *and* $L \geq L_1 + L_2$ *where* L_1 *and* L_2 *are given by* (2.12)..

Proof. Let L_2 be the value selected for classical Krasulina's method then from lower bound on L_2 we can pick L_2/B for distributed mini-batch Krasulina's method (Algorithm 2). Now substituting L_2/B and $T_B = T/(B+\mu)$ in (2.14) gives us:

$$\mathbb{E}_{T_B}\{\Psi_{T_B}\} \leq c_0 C_1 \left(\frac{(B+\mu)L_1}{T}\right)^{c_0/2} + c_0 C_1 \left(\frac{(B+\mu)L_2}{BT}\right)^{c_0/2} + \frac{2C_2\sigma^2(B+\mu)}{BT},$$

□

In this case if we choose $B = O(T^{1-c_0/2})$ then the error after T_B iterations will be $O(1/T + \mu/BT)$ which implies that as long as $\mu = O(B)$, we will achieve the optimal rate of $O(1/T)$. For a special case of tree structured network of N nodes, we have $\mu = \log_2 N$ which results in error on the order of $O(1/T + (\log_2 N)/BT)$. In this special case we can notice that since

$B = \sum_{i=1}^{N} B_i \leq N$ we have $\frac{\log_2 N}{BT} \ll 1$ which shows that the penalty we are paying due to loss of samples is not very large.

2.5 Proofs of Main Results

In this section we provide proofs of the main results which are Theorem 3, Theorem 4, and Theorem 1. Proofs of the supporting lemmas are moved to Sections 2.6, 2.7, and 2.8. The main result of this chapter is given in Theorem 1 whose proof consists of three phases. The initial and intermediate phases here correspond to proving Theorem 3 and Theorem 4, while the final phase proves the main result i.e., Theorem 1.

In initial phase we establish that if we choose step size, γ_t, as in Theorem 1 then for $t \geq 1$ we will have error less than some constant $(1-\epsilon_0)$ with high probability. In second phase, starting from error $1-\epsilon_0$, we sequentially decrease error until it gets below 1/2. For $j = \{0, 1, \ldots, J\}$, defining $\epsilon_j = 2^j \epsilon_0$ and $\epsilon_J \geq 1/2$ we get following sequence of errors:

$$\big\{1-\epsilon_0, 1-\epsilon_1, 1-\epsilon_2, \ldots, 1-\epsilon_J\big\}.$$

Let t_j be the number of Krasulina's method iterations required to decrease the error from $1-\epsilon_{j-1}$ to $1-\epsilon_j$. The main challenge here is to provide a lower bound on t_j such that if we perform t_j iterations of Krasulina's method then, the error will be less than $1-\epsilon_j$ with high probability.

2.5.1 Initial Phase

Our goal in the initial phase is to show that if we pick step size appropriately, i.e, we set L to be large enough (cf. (2.12)) and Assumptions [**A1**] and [**A2**] hold, then the error, Ψ_t, will not exceed a certain value with high probability. This is formally stated in the following result.

Theorem 3. *Let $0 < \epsilon < 1$ (where $\epsilon = \delta^2/8e$ and $0 < \delta < 1$) and*

$$L \geq \frac{8dr^4 \max(1, c^2)}{\epsilon} \ln \frac{4}{\delta} + \frac{8d^2\sigma_B^2 \max(1, c^2)}{\epsilon^2} \ln \frac{4}{\delta} \tag{2.16}$$

*if Assumptions [**A1**] and [**A2**] hold and we choose step size to be $\gamma_t = c/(L+t)$, then we have*

$$\mathbf{Pr}\Big(\sup_{t\geq 0} \Psi_t \geq 1 - \frac{\epsilon}{d}\Big) \leq \sqrt{2e\epsilon} = \frac{\delta}{2}. \tag{2.17}$$

In order to prove Theorem 3 we need lemmas that are stated in the following. We only provide lemma statements in this section and move the proofs to the Appendix 2.6. We start by writing the recursion of error metric Ψ_t in the following lemma.

Lemma 1. *Defining a scalar random variable*

$$z_t := 2\gamma_t \frac{(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)(\boldsymbol{\xi}_t^{\mathrm{T}}\mathbf{q}_1)}{\|\mathbf{v}_{t-1}\|_2^2} \tag{2.18}$$

and $\phi := \lambda_1^2 + \lambda_2^2$, we get the following recursion

(i) $\Psi_t \leq \Psi_{t-1} + 4\gamma_t^2\Big(\Big\|\frac{1}{B}\sum_{i=1}^{B}\mathbf{A}_{i,t} - \mathbf{\Sigma}\Big\|_F^2 + \Psi_{t-1}\phi\Big) - z_t.$

(ii) $\Psi_t \leq \Psi_{t-1} + \gamma_t^2 r^4 - z_t.$

Part (i) of this Lemma will be used to prove the final phase of Algorithm in Theorem 1, while Part (ii) will be used to prove the initial phase (Theorem 3) and the intermediate phase (Theorem 4). Next we will bound moment generating function of Ψ_t conditional on iterate values up to iteration $t-1$. We define this by a σ-field $\mathcal{F}_{t-1}$ (Definition 5) which encodes the previous iterate information. To bound moment generating function of Ψ_t, we need upper bound on variance of z_t which is given in the following lemma.

Lemma 2. *The variance of the random variable z_t is given by*

$$\mathbb{E}\{(z_t - \mathbb{E}\{z_t\})^2|\mathcal{F}_{t-1}\} \leq 16\gamma_t^2\sigma_N^2. \tag{2.19}$$

Using this upper bound on variance of z_t we can now upper bound the moment generating function of Ψ_t as follows.

Lemma 3. *The conditional moment generating function of Ψ_t is upper bounded as*

$$\mathbb{E}\{\exp(s\Psi_t)|\mathcal{F}_{t-1}\} \leq \exp\Bigg(s\Psi_{t-1} - s\mathbb{E}\{z_t|\mathcal{F}_{t-1}\} + s\gamma_t^2 r^4 + s^2\gamma_t^2\sigma_N^2\Bigg), \tag{2.20}$$

This result is similar to [65, Lemma 2.3] with the difference that last term here is sample variance, σ_N^2, in contrast to upper bound on input $\|\mathbf{x}_t\|_2 \leq r$ in [65, Lemma 2.3]. This difference prompts changes in next steps of the analysis of Krasulina's method. Furthermore, it enables us to characterize improvements in convergence rate of Krasulina's method using mini-batches. Having proved Lemma 1 and Lemma 3, we can now use these to prove Theorem 3 as follows.

Proof of Theorem 3. We start by constructing a super-martingale sequence of error Ψ_t. First we define constants

$$\beta_t := \gamma_t^2 r^4, \quad \zeta_t := s\gamma_t^2\sigma_N^2, \quad \tau_t := \sum_{l>t}(\beta_l + \zeta_l), \quad \text{and} \quad M_t := \exp\left(s\Psi_t + s\tau_t\right).$$

Now, taking expectation of M_t conditioned on the filtration $\mathcal{F}_{t-1}$ we get

$$\begin{aligned}\mathbb{E}\{M_t|\mathcal{F}_{t-1}\} &= \mathbb{E}\{\exp(s\Psi_t)|\mathcal{F}_{t-1}\}\exp(s\tau_t)\\ &\overset{(a)}{\leq} \exp(s\Psi_{t-1}+s\beta_t+s\zeta_t+s\tau_t)\\ &= \exp(s\Psi_{t-1}+s\tau_{t-1})\\ &= M_{t-1}.\end{aligned}$$

Here, (a) is due to Lemma 3 and using the fact that $\mathbb{E}\{z_t|\mathcal{F}_{t-1}\} \geq 0$ [65, Theorem 2.1]. These calculations show that sequence $\{M_t\}$ forms a super-martingale. Using sequence M_t, we can now use Doob's martingale inequality [66, pg. 231] to show that Ψ_t will be bounded away from 1 with high probability.

$$\begin{aligned}\mathbf{Pr}\Big(\sup_{t\geq 0}\Psi_t \geq \Delta\Big) &\leq \mathbf{Pr}\Big(\sup_{t\geq 0}\Psi_t+\tau_t \geq \Delta\Big)\\ &= \mathbf{Pr}\Big(\sup_{t\geq 0}\exp(s\Psi_t+s\tau_t) \geq e^{s\Delta}\Big)\\ &= \mathbf{Pr}\Big(\sup_{t\geq 0}M_t \geq e^{s\Delta}\Big)\\ &\leq \frac{\mathbb{E}\{M_{t_0}\}}{e^{s\Delta}} = \exp(-s(\Delta-\tau_0))\mathbb{E}\{e^{s\Psi_0}\}.\end{aligned}$$

Substituting $\Delta = 1-\epsilon/d$ and using [65, Lemma 2.5] to bound $\mathbb{E}e^{s\Psi_0}$ we get

$$\mathbf{Pr}\Big(\sup_{t\geq 0}\Psi_t \geq 1-\frac{\epsilon}{d}\Big) \leq \exp(-s(1-(\epsilon/d)-\tau_0))e^s\sqrt{\frac{d}{2s}}. \tag{2.21}$$

Next we need to compute $\sum_{l>0}\beta_l$ and $\sum_{l>0}\zeta_l$. First we compute

$$\sum_{l>0}\beta_l = \sum_{l>0}\gamma_l^2 r^4 = r^4\sum_{l>0}\gamma_l^2 = r^4\sum_{l>0}\frac{c^2}{(l+L)^2} \leq \frac{r^4c^2}{L}. \tag{2.22}$$

Again using a similar procedure we get

$$\sum_{l>0}\zeta_l \leq \frac{s\sigma_B^2c^2}{L}. \tag{2.23}$$

Combining (2.22) and (2.23), we get

$$\tau_0 \leq \frac{c^2}{L}\Bigg(r^4+s\sigma_B^2\Bigg). \tag{2.24}$$

Now using lower bound on L from (2.12) we get $\tau_{t_0} \leq \epsilon/d$ as shown in Proposition 4. Substituting this in (2.21) we get

$$\mathbf{Pr}\Big(\sup_{t\geq 0}\Psi_t \geq 1-\frac{\epsilon}{d}\Big) \leq \exp(-s(1-\epsilon/d-\epsilon/d))e^s\sqrt{\frac{d}{2s}} = \exp(2s\epsilon/d)\sqrt{\frac{d}{2s}}.$$

Finally, substituting $s = d/4\epsilon$, we have

$$\mathbf{Pr}\Big(\sup_{t\geq 0}\Psi_t \geq 1-\frac{\epsilon}{d}\Big) \leq \sqrt{2e\epsilon}.$$

□

2.5.2 Intermediate Epochs of Improvement

In Theorem 3 we have shown that if we choose L such that it satisfies the lower bound given in Theorem 3 then we have error Ψ_t greater than $1-\epsilon_0$ (here, $\epsilon_0 = \delta^2/8ed$) with probability δ. Next, our aim is to show that if we perform enough iterations t_J of Krasulina's method then for any $t \geq t_J$ the error in the iterate will be bounded by $\Psi_t \leq 1/2$ with high probability . In order to prove this we divide our analysis into different epochs $j = \{1, \dots, J\}$. Starting from $1-\epsilon_0$ we provide a lower bound on number of Krasulina's method iterations, t_j, such that we progressively increase ϵ_j in each epoch until we reach ϵ_J.

Theorem 4. *For $\rho \in (0, 1/2)$, $c = c_0/(2(\lambda_1 - \lambda_2))$, $c_0 > 0$, number of nodes $N > 1$, and $L \geq \frac{8r^4 \max(1,c^2)}{\epsilon_0} \ln \frac{4}{\delta} + \frac{8\sigma_N^2 \max(1,c^2)}{\epsilon_0^2} \ln \frac{4}{\delta}$, take step size to be $\gamma_t = c/(L+t)$. Now pick any $0 < \delta < 1$ and select a schedule $(0, \epsilon_0), (t_1, \epsilon_1), \dots, (t_J, \epsilon_J)$ such that the following conditions are satisfied*

[**C1**] $\epsilon_0 = \frac{\delta^2}{8ed}$ *and* $\frac{3}{2}\epsilon_j \leq \epsilon_{j+1} \leq 2\epsilon_j$ *for* $0 \leq j < J$, *and* $\epsilon_{J-1} \leq \frac{1}{4}$

[**C2**] $\Big(t_{j+1} + L + 1\Big) \geq e^{5/c_0}\Big(t_j + L + 1\Big)$ *for* $0 \leq j < J$

then, $\mathbf{Pr}(\Omega^{'}) \geq 1 - \delta$.

In order to prove this result we need Lemmas 4–7 which are stated as follows:

Lemma 4. *For $t > t_j$ the moment generating function of Ψ_t, restricted to sample space $\Omega_t^{'}$ is given by:*

$$\mathbb{E}_t\Big\{e^{s\Psi_t}\Big\} \leq \exp\left(s\left(\Psi_{t-1}\Big(1 - \frac{c_0\epsilon_j}{t+L}\Big) + \frac{c^2r^4}{(t+L)^2} + \frac{sc^2\sigma_N^2}{(t+L)^2}\right)\right)$$

Lemma 5. *For $t > t_j$ and any $s > 0$*

$$\mathbb{E}_t\{e^{s\Psi_t}\} \leq \exp\left(s(1-\epsilon_j)\left(\frac{t_j+L+1}{t+L+1}\right)^{c_0\epsilon_j} + \left(sc^2r^4 + s^2c^2\sigma_N^2\right)\left(\frac{1}{t_j+L} - \frac{1}{t+L}\right)\right). \tag{2.25}$$

Using result from Lemma 5 our next result deals with specific value of t, i.e., $t = t_{j+1}$.

Lemma 6. *If Conditions* [**C1**]–[**C2**] *are satisfied, then for $0 \leq j < J$ and any $s > 0$*

$$\mathbb{E}_{t_{j+1}}\{e^{s\Psi_{t_{j+1}}}\} \leq \exp\left(s(1-\epsilon_{j+1}) - s\epsilon_j + \left(sc^2r^4 + s^2c^2\sigma_N^2\right)\Big(\frac{1}{t_j+L} - \frac{1}{t_{j+1}+L}\Big)\right)$$

Lemma 7. *If Conditions* [**C1**]–[**C2**] *are satisfied, then picking any $0 < \delta < 1$ we have*

$$\sum_{j=1}^{J} \mathbf{Pr}_{t_j}\Big(\sup_{t \geq t_j} \Psi_t > 1 - \epsilon_j\Big) \leq \frac{\delta}{2}.$$

Proof. (Proof of Theorem 4) Now using results from Lemma 7 and Theorem 3 and applying union bound we get the proof for Theorem 4. □

2.5.3 Final Epoch

Now that we have shown that $\Psi_t \leq 1/2$ with probability $1-\delta$, in the final epoch we show that how Ψ_t decreases further as a function of algorithm iterations. The following result captures the rate at which Ψ_t decreases during this phase.

Lemma 8. *For any $t > t_J$, expected error in Ψ_t is given by*

$$\mathbb{E}_t\{\Psi_t\} \leq \left(1 + \frac{c_0^2\phi^2}{4(t+L)^2(\lambda_1-\lambda_2)^2} - \frac{c_0}{2(t+L)}\right)\mathbb{E}_{t-1}\{\Psi_{t-1}\} + \frac{c^2\sigma_N^2}{(t+L)^2}.$$

Next we will prove our main result in Theorem 1.

Proof. (*Proof of Theorem 1*) Following the same procedure as in the proof of [65, Theorem 1.1] we define epochs (t_j, ϵ_j) that satisfy conditions in Theorem 4. We define epochs such that $\epsilon_J = 1/2$ and $\epsilon_{j+1} = 2\epsilon_j$ whenever possible . Then we have $J = \log_2\big(1/(2\epsilon_0)\big)$ (since $\epsilon_J = 2\epsilon_{J-1} = \cdots = 2^J\epsilon_0 \Rightarrow 2^J = \epsilon_J/\epsilon_0 = 1/2\epsilon_0$). This implies

$$t_J + L + 1 = \big(L+1\big)\exp\Big(\frac{5J}{c_0}\Big) = \big(L+1\big)\Big(\frac{1}{2\epsilon_0}\Big)^{5/(c_0\ln 2)} = \big(L+1\big)\Big(\frac{4ed}{\delta^2}\Big)^{5/(c_0\ln 2)}. \quad (2.26)$$

Defining $a_1 = c_0^2\phi^2/4(\lambda_1-\lambda_2)^2$, $a_2 = c_0/2$ and $b = c^2\sigma_N^2$, and for $t > t_J$ using Lemma 8 we have

$$\mathbb{E}_t\{\Psi_t\} \leq \Big(1 + \frac{a_1}{(t+L)^2} - \frac{a_2}{t+L}\Big)\mathbb{E}_{t-1}\{\Psi_{t-1}\} + \frac{b}{(t+L)^2}.$$

Now using Proposition 1 for $c_0 > 2$ we get

$$\begin{aligned}
\mathbb{E}_t\{\Psi_t\} &\leq \Big(\frac{t_J+L+1}{t+L+1}\Big)^{\frac{c_0}{2}} \exp\Big(\frac{a_1}{t_J+L+1}\Big)\mathbb{E}_{t_J}\{\Psi_{t_J}\} \\
&\quad + \frac{b}{a_2-1}\Big(1+\frac{1}{t_J+L+1}\Big)^2 \exp\Big(\frac{a_1}{t_J+L+1}\Big)\frac{1}{t+L+1} \\
&\overset{(a)}{\leq} \frac{1}{2}\Big(\frac{L+1}{t+L+1}\Big)^{\frac{c_0}{2}}\Big(\frac{4ed}{\delta^2}\Big)^{\frac{5a_2}{(c_0\ln 2)}} \exp\Big(\frac{a_1}{t_J+L+1}\Big) \\
&\quad + \frac{b}{a_2-1}\exp\Big(\frac{2}{t_J+L+1}\Big)\exp\Big(\frac{a_1}{t_J+L+1}\Big)\frac{1}{t+L+1} \\
&= \frac{1}{2}\Big(\frac{L+1}{t+L+1}\Big)^{\frac{c_0}{2}}\Big(\frac{4ed}{\delta^2}\Big)^{\frac{5}{(2\ln 2)}} \exp\Big(\frac{a_1}{(L+1)(4ed/\delta^2)^{(5/2\ln 2)}}\Big) \\
&\quad + \frac{2c^2\sigma_N^2}{c_0-2}\exp\Big(\frac{2+a_1}{(L+1)(4ed/\delta^2)^{(5/2\ln 2)}}\Big)\frac{1}{(t+L+1)}.
\end{aligned}$$

Here, the inequality in (a) is due to (2.26) and we use the fact that $(1+x)^a \leq \exp(ax)$, for $x < 1$. In addition, since $(4ed/\delta^2)^{(5/2\ln 2)} \geq 1$, therefore we get

$$\begin{aligned}
\mathbb{E}_t\{\Psi_t\} &\leq \frac{1}{2}\Big(\frac{L+1}{t+L+1}\Big)^{\frac{c_0}{2}}\Big(\frac{4ed}{\delta^2}\Big)^{\frac{5}{(2\ln 2)}} \exp\Big(\frac{a_1}{L+1}\Big) + \frac{2c^2\sigma_N^2}{c_0-2}\exp\Big(\frac{a_1+2}{L+1}\Big)\frac{1}{(t+1)} \\
&\leq \frac{1}{2}\Big(\frac{L+1}{t+L+1}\Big)^{\frac{c_0}{2}}\Big(\frac{4ed}{\delta^2}\Big)^{\frac{5}{(2\ln 2)}} e^{a_1/L} + \frac{2c^2\sigma_N^2 e^{(a_1+2)/L}}{c_0-2}\frac{1}{(t+L+1)} \\
&= C_1\Big(\frac{L+1}{t+L+1}\Big)^{\frac{c_0}{2}} + C_2\Big(\frac{\sigma_N^2}{t+L+1}\Big).
\end{aligned} \quad (2.27)$$

□

2.6 Proof of Supporting Lemmas for Initial Phase

To prove Theorem 3 we start by writing the recursion of error metric Ψ_t in the Lemma 1. In order to prove Lemma 1 we first need the following result:

Lemma 9. *The second moment of the stochastic update vector ξ_t in Krasulina's method is upper bounded as*

$$\mathbb{E}\left(\frac{\|\boldsymbol{\xi}_t\|_2^2}{\|\mathbf{v}_{t-1}\|_2^2}\right) \leq \frac{\mathbb{E}\|\boldsymbol{\xi}_t - \mathbb{E}\boldsymbol{\xi}_t\|_2^2}{\|\mathbf{v}_{t-1}\|_2^2} + \Psi_{t-1}\phi,$$

where $\phi := \lambda_1^2 + \lambda_2^2$.

Proof. We start by writing $\mathbb{E}\|\boldsymbol{\xi}_t - \mathbb{E}\boldsymbol{\xi}_t\|_2^2$ in terms of $\mathbb{E}\|\boldsymbol{\xi}_t\|_2^2$ as follows

$$\begin{aligned}\mathbb{E}\|\boldsymbol{\xi}_t - \mathbb{E}\boldsymbol{\xi}_t\|_2^2 &= \mathbb{E}\left\{\boldsymbol{\xi}_t^{\mathrm{T}}\boldsymbol{\xi}_t + (\mathbb{E}\boldsymbol{\xi}_t)^{\mathrm{T}}\mathbb{E}\boldsymbol{\xi}_t - \boldsymbol{\xi}_t^{\mathrm{T}}\mathbb{E}\boldsymbol{\xi}_t - (\mathbb{E}\boldsymbol{\xi}_t)^{\mathrm{T}}\boldsymbol{\xi}_t\right\} \\ &= \mathbb{E}\|\boldsymbol{\xi}_t\|_2^2 - \mathbb{E}\boldsymbol{\xi}_t^{\mathrm{T}}\mathbb{E}\boldsymbol{\xi}_t.\end{aligned}$$

Now defining $C_t := \mathbb{E}\boldsymbol{\xi}_t^{\mathrm{T}}\mathbb{E}\boldsymbol{\xi}_t$ and rearranging the above equation we get

$$\mathbb{E}\|\boldsymbol{\xi}_t\|_2^2 = \mathbb{E}\|\boldsymbol{\xi}_t - \mathbb{E}\boldsymbol{\xi}_t\|_2^2 + C_t.$$

Now substituting value of $\boldsymbol{\xi}_t$ from (2.7) we get

$$\begin{aligned}\frac{C_t}{\|\mathbf{v}_{t-1}\|_2^2} = \frac{\mathbb{E}\boldsymbol{\xi}_t^{\mathrm{T}}\mathbb{E}\boldsymbol{\xi}_t}{\|\mathbf{v}_{t-1}\|_2^2} &= \frac{1}{\|\mathbf{v}_{t-1}\|_2^2}\left(\boldsymbol{\Sigma}\mathbf{v}_{t-1} - \frac{\mathbf{v}_{t-1}^{\mathrm{T}}\boldsymbol{\Sigma}\mathbf{v}_{t-1}\mathbf{v}_{t-1}}{\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{v}_{t-1}}\right)^{\mathrm{T}}\left(\boldsymbol{\Sigma}\mathbf{v}_{t-1} - \frac{\mathbf{v}_{t-1}^{\mathrm{T}}\boldsymbol{\Sigma}\mathbf{v}_{t-1}\mathbf{v}_{t-1}}{\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{v}_{t-1}}\right) \\ &= \frac{\mathbf{v}_{t-1}^{\mathrm{T}}\boldsymbol{\Sigma}^2\mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2^2} - \left(\frac{\mathbf{v}_{t-1}^{\mathrm{T}}\boldsymbol{\Sigma}\mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2^2}\right)^2. \qquad (2.28)\end{aligned}$$

Since $\boldsymbol{\Sigma}$ is a positive definite matrix we can write its eigenvalue decomposition as

$$\boldsymbol{\Sigma} = \sum_{i=1}^{d}\lambda_i\mathbf{q}_i\mathbf{q}_i^{\mathrm{T}},$$

where $\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_d$ and $\mathbf{q}_1, \ldots, \mathbf{q}_d$ are the eigenvalues and eigenvectors of $\boldsymbol{\Sigma}$ respectively.

It follows that

$$\begin{aligned}
\frac{C_t}{\|\mathbf{v}_{t-1}\|_2^2} &= \sum_{i=1}^{d} \lambda_i^2 \frac{(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_i)^2}{\|\mathbf{v}_{t-1}\|_2^2} - \left(\sum_{i=1}^{d} \lambda_i \frac{(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_i)^2}{\|\mathbf{v}_{t-1}\|_2^2} \right)^2 \\
&= \lambda_1^2 \frac{(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_{t-1}\|_2^2} + \sum_{i=2}^{d} \lambda_i^2 \frac{(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_i)^2}{\|\mathbf{v}_{t-1}\|_2^2} - \left(\lambda_1 \frac{(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_{t-1}\|_2^2} + \sum_{i=2}^{d} \lambda_i \frac{(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_i)^2}{\|\mathbf{v}_{t-1}\|_2^2} \right)^2 \\
&\leq \lambda_1^2 \frac{(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_{t-1}\|_2^2} + \lambda_2^2 \sum_{i=2}^{d} \frac{(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_i)^2}{\|\mathbf{v}_{t-1}\|_2^2} - \lambda_1^2 \frac{(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)^4}{\|\mathbf{v}_{t-1}\|_2^4} \\
&= \lambda_1^2 \frac{(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_{t-1}\|_2^2} \left(1 - \frac{(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_{t-1}\|_2^2} \right) + \lambda_2^2 \left(1 - \frac{(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_{t-1}\|_2^2} \right).
\end{aligned}$$

From definition of Ψ_{t-1} we get

$$\frac{C_t}{\|\mathbf{v}_{t-1}\|_2^2} \leq \Psi_{t-1}\big((1-\Psi_{t-1})\lambda_1^2 + \lambda_2^2\big) \leq \Psi_{t-1}\big(\lambda_1^2 + \lambda_2^2\big) = \Psi_{t-1}\phi.$$

□

Now using Lemma 9 we can prove Lemma 1 in the following.

2.6.1 Proof of Lemma 1

From (2.9), we have

$$\Psi_t = \frac{\|\mathbf{v}_t\|_2^2 - (\mathbf{v}_t^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_t\|_2^2}.$$

Now substituting $\mathbf{v}_t$ from (2.7) we get

$$\begin{aligned}
\Psi_t &= \frac{\|\mathbf{v}_{t-1} + \gamma_t \boldsymbol{\xi}_t\|_2^2 - ((\mathbf{v}_{t-1} + \gamma_t \boldsymbol{\xi}_t)^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_t\|_2^2}, \\
&\overset{(a)}{=} \frac{\|\mathbf{v}_{t-1}\|_2^2 + \gamma_t^2\|\boldsymbol{\xi}_t\|_2^2 - ((\mathbf{v}_{t-1} + \gamma_t \boldsymbol{\xi}_t)^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_t\|_2^2}, \\
&\overset{(b)}{\leq} \frac{\|\mathbf{v}_{t-1}\|_2^2 + \gamma_t^2\|\boldsymbol{\xi}_t\|_2^2 - ((\mathbf{v}_{t-1} + \gamma_t \boldsymbol{\xi}_t)^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_{t-1}\|_2^2}, \\
&= 1 + \gamma_t^2 \frac{\|\boldsymbol{\xi}_t\|_2^2}{\|\mathbf{v}_{t-1}\|_2^2} - \frac{((\mathbf{v}_{t-1} + \gamma_t \boldsymbol{\xi}_t)^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_{t-1}\|_2^2} \\
&= 1 + \gamma_t^2 \frac{\|\boldsymbol{\xi}_t\|_2^2}{\|\mathbf{v}_{t-1}\|_2^2} - \frac{((\mathbf{v}_{t-1} + \gamma_t \boldsymbol{\xi}_t)^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_{t-1}\|_2^2} \\
&= 1 + \gamma_t^2 \frac{\|\boldsymbol{\xi}_t\|_2^2}{\|\mathbf{v}_{t-1}\|_2^2} - \frac{(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)^2 + \gamma_t^2(\boldsymbol{\xi}_t^{\mathrm{T}}\mathbf{q}_1)^2 + 2\gamma_t(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)(\boldsymbol{\xi}_t^{\mathrm{T}}\mathbf{q}_1)}{\|\mathbf{v}_{t-1}\|_2^2} \\
&= 1 - \frac{(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_{t-1}\|_2^2} + \gamma_t^2 \frac{\|\boldsymbol{\xi}_t\|_2^2 - (\boldsymbol{\xi}_t^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_{t-1}\|_2^2} - 2\gamma_t \frac{(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)(\boldsymbol{\xi}_t^{\mathrm{T}}\mathbf{q}_1)}{\|\mathbf{v}_{t-1}\|_2^2} \\
&= \Psi_{t-1} + \gamma_t^2 \frac{\|\boldsymbol{\xi}_t\|_2^2}{\|\mathbf{v}_{t-1}\|_2^2} - 2\gamma_t \frac{(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)(\boldsymbol{\xi}_t^{\mathrm{T}}\mathbf{q}_1)}{\|\mathbf{v}_{t-1}\|_2^2}. \qquad (2.29)
\end{aligned}$$

Here (a) and (b) are due to [65, Lemma A.1], (a) is true because $\mathbf{v}_{t-1}$ is perpendicular to $\boldsymbol{\xi}_t$ and (b) is true because $\|\mathbf{v}_{t-1}\|_2 \leq \|\mathbf{v}_t\|_2$. Now defining $\widehat{\mathbf{v}}_{t-1} = \mathbf{v}_{t-1}/\|\mathbf{v}_{t-1}\|_2$, the second term in the above inequality can be bounded as

$$\begin{aligned}\frac{\|\boldsymbol{\xi}_t\|_2^2}{\|\mathbf{v}_{t-1}\|_2^2} &= \frac{\|\boldsymbol{\xi}_t - \mathbb{E}\boldsymbol{\xi}_t\|_2^2 + \mathbb{E}\boldsymbol{\xi}_t^{\mathrm{T}}\mathbb{E}\boldsymbol{\xi}_t}{\|\mathbf{v}_t\|_2^2}, \\ &\overset{(a)}{\leq} \|\mathbf{A}_{i,t} - \boldsymbol{\Sigma}\|_2^2 + \Psi_{t-1}\rho \leq \|\mathbf{A}_{i,t} - \boldsymbol{\Sigma}\|_F^2 + \Psi_{t-1}\rho. \qquad (2.30)\end{aligned}$$

Here, (a) is due to Lemma 9. Substituting (2.30) in (2.29) we complete proof of Part 1 of Lemma 1. Next, we prove Part 2 of Lemma 1.

$$\begin{aligned}\frac{\|\boldsymbol{\xi}_t\|_2^2}{\|\mathbf{v}_{t-1}\|_2^2} &= \frac{\|(1/B)\sum_{i=1}^{B}\boldsymbol{\xi}_{i,t}\|_2^2}{\|\mathbf{v}_{t-1}\|_2^2} \\ &= \frac{(1/B^2)\|\sum_{i=1}^{B}\boldsymbol{\xi}_{i,t}\|_2^2}{\|\mathbf{v}_{t-1}\|_2^2} \\ &\overset{(a)}{\leq} \frac{(1/B^2)\sum_{i=1}^{B} B\|\boldsymbol{\xi}_{i,t}\|_2^2}{\|\mathbf{v}_{t-1}\|_2^2} \\ &= \frac{\sum_{i=1}^{B}(\mathbf{x}_{i,t}^{\mathrm{T}}\mathbf{v}_{t-1})^2\|\mathbf{x}_{i,t} - (\mathbf{x}_{i,t}^{\mathrm{T}}\widehat{\mathbf{v}}_{t-1})\widehat{\mathbf{v}}_{t-1}\|_2^2}{B\|\mathbf{v}_{t-1}\|_2^2} \\ &\leq \frac{1}{B}\sum_{i=1}^{B}\|\mathbf{x}_{i,t}\|_2^2\|\mathbf{x}_{i,t} - (\mathbf{x}_{i,t}^{\mathrm{T}}\widehat{\mathbf{v}}_{t-1})\widehat{\mathbf{v}}_{t-1}\|_2^2 \\ &= \frac{1}{B}\sum_{i=1}^{B}\|\mathbf{x}_{i,t}\|_2^2(\|\mathbf{x}_{i,t}\|_2^2 - (\mathbf{x}_{i,t}^{\mathrm{T}}\widehat{\mathbf{v}}_{t-1})^2) \\ &\leq \sum_{i=1}^{B}\frac{\|\mathbf{x}_{i,t}\|_2^4}{B} \leq \max_i \|\mathbf{x}_{i,t}\|_2^4 \leq r^4. \qquad (2.31)\end{aligned}$$

Here (a) is by using Cauchy-Schwartz inquality and last inequality is due to the Assumption **[A1]**. Now substituting this in (2.29) completes the proof.

2.6.2 Proof of Lemma 2

$$\begin{aligned}\mathbb{E}\{(z_t - \mathbb{E}\{z_t\})^2|\mathcal{F}_{t-1}\} &= \mathbb{E}\left\{\left(\frac{2\gamma_t(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)(\boldsymbol{\xi}_t^{\mathrm{T}}\mathbf{q}_1)}{\|\mathbf{v}_{t-1}\|_2^2} - \mathbb{E}\left\{\frac{2\gamma_t(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)(\boldsymbol{\xi}_t^{\mathrm{T}}\mathbf{q}_1)}{\|\mathbf{v}_{t-1}\|_2^2}\right\}\right)^2 \middle| \mathcal{F}_{t-1}\right\}, \\ &= \frac{4\gamma_t^2(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_{t-1}\|_2^4}\mathbb{E}\left\{\left(\boldsymbol{\xi}_t^{\mathrm{T}}\mathbf{q}_1 - \mathbb{E}\left\{\boldsymbol{\xi}_t^{\mathrm{T}}\mathbf{q}_1\right\}\right)^2\right\},\end{aligned}$$

substituting value of ξ_t we get

$$\begin{aligned}\mathbb{E}\{(z_t - \mathbb{E}\{z_t\})^2|\mathcal{F}_{t-1}\} &= \frac{4\gamma_t^2(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_{t-1}\|_2^4}\mathbb{E}\Bigg\{\Bigg(\Big(\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}\mathbf{v}_{t-1} - \frac{\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}\mathbf{v}_{t-1}\mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2^2}\Big)^{\mathrm{T}}\mathbf{q}_1 \\ &\qquad - \mathbb{E}\Big\{\Big(\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}\mathbf{v}_{t-1} - \frac{\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}\mathbf{v}_{t-1}\mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2^2}\Big)^{\mathrm{T}}\mathbf{q}_1\Big\}\Bigg)^2\Bigg\}, \\ &= \frac{4\gamma_t^2(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_{t-1}\|_2^4}\mathbb{E}\Bigg\{\Bigg(\Big(\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}\mathbf{v}_{t-1} - \frac{\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}\mathbf{v}_{t-1}\mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2^2}\Big)^{\mathrm{T}}q_1 \\ &\qquad - \mathbf{v}_{t-1}^{\mathrm{T}}\mathbb{E}\{\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}\}\mathbf{q}_1 + \frac{\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{v}_{t-1}^{\mathrm{T}}\mathbb{E}\{\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}\}\mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2^2}\mathbf{q}_1\Bigg)^2\Bigg\}.\end{aligned}$$

Since $\boldsymbol{\Sigma} := \mathbb{E}\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}$ is the covariance matrix, we get

$$\begin{aligned}&\mathbb{E}\{(z_t - \mathbb{E}\{z_t\})^2|\mathcal{F}_{t-1}\} \\ &\quad= \frac{4\gamma_t^2(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_{t-1}\|_2^4}\mathbb{E}\Bigg\{\Bigg(\Big((\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}} - \boldsymbol{\Sigma})\mathbf{v}_{t-1} - \frac{\mathbf{v}_{t-1}^{\mathrm{T}}(\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}} - \boldsymbol{\Sigma})\mathbf{v}_{t-1}\mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2^2}\Big)^{\mathrm{T}}\mathbf{q}_1\Bigg)^2\Bigg\}, \\ &\quad= \frac{4\gamma_t^2(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_{t-1}\|_2^4}\mathbb{E}\Bigg\{\Bigg(\Big(\mathbf{q}_1^{\mathrm{T}}(\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}} - \boldsymbol{\Sigma})\mathbf{v}_{t-1} - \frac{\Big(\mathbf{v}_{t-1}^{\mathrm{T}}(\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}} - \boldsymbol{\Sigma})\mathbf{v}_{t-1}\Big)\mathbf{q}_1^{\mathrm{T}}\mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2^2}\Big)\Bigg)^2\Bigg\} \\ &\quad\le \frac{8\gamma_t^2(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_{t-1}\|_2^4}\mathbb{E}\Bigg\{\Big(\mathbf{q}_1^{\mathrm{T}}(\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}} - \boldsymbol{\Sigma})\mathbf{v}_{t-1}\Big)^2 + \Bigg(\frac{\Big(\mathbf{v}_{t-1}^{\mathrm{T}}(\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}} - \boldsymbol{\Sigma})\mathbf{v}_{t-1}\Big)\mathbf{q}_1^{\mathrm{T}}\mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2^2}\Bigg)^2\Bigg\} \\ &\quad= \frac{8\gamma_t^2(\mathbf{v}_{t-1}^{\mathrm{T}}\mathbf{q}_1)^2}{\|\mathbf{v}_{t-1}\|_2^2}\mathbb{E}\Bigg\{\Bigg(\frac{\mathbf{q}_1^{\mathrm{T}}(\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}} - \boldsymbol{\Sigma})\mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2}\Bigg)^2 + \Bigg(\frac{\mathbf{v}_{t-1}^{\mathrm{T}}(\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}} - \boldsymbol{\Sigma})\mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2^2}\Bigg)^2\Bigg(\frac{\mathbf{q}_1^{\mathrm{T}}\mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2}\Bigg)^2\Bigg\} \\ &\quad\le 8\gamma_t^2\mathbb{E}\Bigg\{\Bigg(\frac{\mathbf{q}_1^{\mathrm{T}}(\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}} - \boldsymbol{\Sigma})\mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2}\Bigg)^2 + \Bigg(\frac{\mathbf{v}_{t-1}^{\mathrm{T}}(\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}} - \boldsymbol{\Sigma})\mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2^2}\Bigg)^2\Bigg\}. \qquad (2.32)\end{aligned}$$

The last inequality in (2.32) is due to the fact that $\left(\frac{\mathbf{q}_1^{\mathrm{T}}\mathbf{v}_{t-1}}{\|\mathbf{v}_{t-1}\|_2}\right)^2 \le 1$. We can see that both the remaining terms are Rayleigh quotients of matrix $(\boldsymbol{\Sigma} - \mathbf{x}_t\mathbf{x}_t^{\mathrm{T}})$ and hence the largest eigenvalue of $(\boldsymbol{\Sigma} - \mathbf{x}_t\mathbf{x}_t^{\mathrm{T}})$ maximizes the both terms. Using this fact we get

$$\mathbb{E}\{(z_t - \mathbb{E}\{z_t\})^2|\mathcal{F}_{t-1}\} \le 16\gamma_t^2\mathbb{E}\{\|\boldsymbol{\Sigma} - \mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}\|_2^2\} \le 16\gamma_t^2\mathbb{E}\{\|\boldsymbol{\Sigma} - \mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}\|_F^2\}.$$

Using Definition 2 we get

$$\mathbb{E}\{(z_t - \mathbb{E}\{z_t\})^2|\mathcal{F}_{t-1}\} \le 16\gamma_t^2\sigma_N^2.$$

2.6.3 Proof of Lemma 3

Now that we have computed the upper bound on variance of z_t we are ready to compute the upper bound on moment generating function of Ψ_t. Using Lemma 1 we can write the moment generating function of Ψ_t as follows

$$\begin{aligned}\mathbb{E}\{\exp(s\Psi_t)|\mathcal{F}_{t-1}\} &\leq \mathbb{E}\Big\{\exp\Big(s\Psi_{t-1} + s\gamma_t^2 r^4 - sz_t\Big)\Big|\mathcal{F}_{t-1}\Big\} \\ &= \exp(s\Psi_{t-1} + s\gamma_t^2 r^4)\mathbb{E}\Big\{\exp\Big(-sz_t\Big)\Big|\mathcal{F}_{t-1}\Big\} \\ &= \exp(s\Psi_{t-1} + s\gamma_t^2 r^4 - s\mathbb{E}\{z_t|\mathcal{F}_{t-1}\})\mathbb{E}\Big\{\exp\Big(-s(z_t - \mathbb{E}\{z_t\})\Big)\Big|\mathcal{F}_{t-1}\Big\}.\end{aligned} \tag{2.33}$$

Now we can bound this using Bennett's inequality (Proposition 2). In order to apply Bennett's inequality, we need to compute the variance and range of the random variable z_t. We have already computed the variance of z_t in Lemma 2. Next we compute the range of $(z_t - \mathbb{E}\{z_t\})$ as follows:

$$\Big|z_t - \mathbb{E}\{z_t\}\Big| \leq 2|z_t| \leq 2\gamma_t\|\mathbf{x}_t\|_2^2 \leq 2\gamma_t r^2 =: h.$$

Here last inequality is due to the Assumption [**A1**]. For parameters σ_N^2 and h using Bennett's inequality we get

$$\mathbb{E}\{\exp(s\Psi_t)|\mathcal{F}_{t-1}\} \leq \exp\left(s\Psi_{t-1} - s\mathbb{E}\{z_t|\mathcal{F}_{t-1}\} + s\gamma_t^2 r^4 + s^2\gamma_t^2\sigma_N^2\left(\frac{e^{sh} - 1 - sh}{(sh)^2}\right)\right).$$

For $L \geq L_1 + L_2$, where L_1 and L_2 are given by (2.12), we show that $(\frac{e^{sh}-1-sh}{sh}) \leq 1$ in Proposition 3, which completes the proof as follows:

$$\mathbb{E}\{\exp(s\Psi_t)|\mathcal{F}_{t-1}\} \leq \exp\left(s\Psi_{t-1} - s\mathbb{E}\{z_t|\mathcal{F}_{t-1}\} + s\gamma_t^2 r^4 + s^2\gamma_t^2\sigma_N^2\right).$$

2.7 Proof of Supporting Lemmas for Intermediate Epochs of Improvement

2.7.1 Proof of Lemma 4

Using Lemma 3, we have

$$\begin{aligned}\mathbb{E}\{e^{s\Psi_t}|\mathcal{F}_{t-1}\} &\leq \exp\left(s\left(\Psi_{t-1} + \gamma_t^2 r^4 - \mathbb{E}\{z_t|\mathcal{F}_{t-1}\} + s\gamma_t^2\sigma_N^2\right)\right) \\ &\overset{(a)}{\leq} \exp\left(s\left(\Psi_{t-1} - 2\gamma_t\Big(\lambda_1 - \lambda_2\Big)\Psi_{t-1}\Big(1 - \Psi_{t-1}\Big) + \gamma_t^2 r^4 + s\gamma_t^2\sigma_N^2\right)\right) \\ &\overset{(b)}{\leq} \exp\left(s\left(\Psi_{t-1} - \frac{c_0\Psi_{t-1}\Big(1 - \Psi_{t-1}\Big)}{t+L} + \frac{c^2 r^4}{(t+L)^2} + \frac{sc^2\sigma_N^2}{(t+L)^2}\right)\right).\end{aligned} \tag{2.34}$$

Here, (a) is due to [65, Lemma A.3] and (b) is by substituting $\gamma_t = c/(t+L) = c_0/2(\lambda_1 - \lambda_2)(t+L)$. Finally, for $\omega \in \Omega_t'$ we have $\Psi_{t-1}(\omega) \leq 1 - \epsilon_j$. Now taking expectation over Ω_t' we get the result.

2.7.2 Proof of Lemma 5

Let $\alpha_t = 1 - \frac{c_0\epsilon_j}{t+L}$ and $\xi_t(s) = \frac{sc^2r^4}{(t+L)^2} + \frac{s^2c^2\sigma_N^2}{(t+L)^2}$. Substituting α_t and $\xi_t(s)$ in Lemma 4, we get

$$\mathbb{E}_t\{e^{s\Psi_t}\} \leq \mathbb{E}_t\{e^{s\alpha_t\Psi_{t-1}}\}\exp\big(\xi_t(s)\big) \leq \mathbb{E}_{t-1}\{e^{s\alpha_t\Psi_{t-1}}\}\exp\big(\xi_t(s)\big). \tag{2.35}$$

Note that the second inequality in (2.35) is due to [65, Lemma 2.8]. Applying this procedure repeatedly yields

$$\begin{aligned}\mathbb{E}_t\{e^{s\Psi_t}\} &\leq \mathbb{E}_{t_j+1}\{\exp\big(s\Psi_{t_j}\alpha_t\dots\alpha_{t_j+1}\big)\}\exp\big(\xi_t(s)\big)\dots\exp\big(\xi_{t_j+1}(s\alpha_t\dots\alpha_{t_j+1})\big)\\ &\leq \mathbb{E}_{t_j+1}\{\exp\big(s\Psi_{t_j}\alpha_t\dots\alpha_{t_j+1}\big)\}\exp\big(\xi_t(s)\big)\dots\exp\big(\xi_{t_j+1}(s)\big).\end{aligned}$$

Substituting values of α_t and $\xi_t(s)$ from here we have

$$\begin{aligned}\mathbb{E}_t\{e^{s\Psi_t}\} \leq \mathbb{E}_{t_j+1}\Big\{&\exp\Big(s\Psi_{t_j}\Big(1-\frac{c_0\epsilon_j}{t+L}\Big)\dots\Big(1-\frac{c_0\epsilon_j}{t_j+L+1}\Big)\Big)\Big\}\\ &\exp\bigg(\Big(sc^2r^4+s^2c^2\sigma_N^2\Big)\Big(\frac{1}{(t+L)^2}+\dots+\frac{1}{(t_j+L+1)^2}\Big)\bigg)\\ \leq \exp\Big(&s(1-\epsilon_j)\exp\Big(-c_0\epsilon_j\Big(\frac{1}{t+L}+\dots+\frac{1}{t_j+L+1}\Big)\Big)\Big)\\ &\exp\bigg(\Big(sc^2r^4+s^2c^2\sigma_N^2\Big)\Big(\frac{1}{(t+L)^2}+\dots+\frac{1}{(t_j+L+1)^2}\Big)\bigg).\end{aligned} \tag{2.36}$$

Here last inequality is true because $\Psi_{t_j}(\omega) \leq 1-\epsilon_j$ for $\omega \in \Omega'_{t_j+1}$ and $1-x \leq e^{-x}$ for $x \leq 1$. Next we bound summations as follows

$$\frac{1}{t+L}+\dots+\frac{1}{t_j+L+1} \geq \int_{t_j+1}^{t+1}\frac{dx}{x+L} = \ln\frac{t+L+1}{t_j+L+1},$$

$$\frac{1}{(t+L)^2}+\dots+\frac{1}{(t_j+L+1)^2} \leq \int_{t_j}^{t}\frac{dx}{(x+L)^2} = \frac{1}{t_j+L}-\frac{1}{t+L}.$$

Substituting these bounds in (2.36) we get the desired result.

2.7.3 Proof of Lemma 6

Using result from Lemma 5, this lemma deals with specific value of t, i.e., $t = t_{j+1}$. For $t = t_{j+1}$, (2.25) gives

$$\mathbb{E}_{t_j+1}\{e^{s\Psi_{t_{j+1}}}\} \leq \exp\left(s(1-\epsilon_j)\left(\frac{t_j+L+1}{t_{j+1}+L+1}\right)^{c_0\epsilon_j} + \left(sc^2r^4+s^2c^2\sigma_N^2\right)\left(\frac{1}{t_j+L}-\frac{1}{t_{j+1}+L}\right)\right), \tag{2.37}$$

using conditions [**C1**] and [**C2**] and the fact that $e^{-2x} \leq 1-x$ for $0 \leq x \leq 3/4$ we get

$$(1-\epsilon_j)\Big(\frac{t_j+L+1}{t_{j+1}+L+1}\Big)^{c_0\epsilon_j} \leq e^{-\epsilon_j}(e^{-5/c_0})^{c_0\epsilon_j} = e^{-6\epsilon_j} \leq 1-3\epsilon_j \leq 1-\epsilon_{j+1}-\epsilon_j.$$

Substituting this in (2.37) we finish the proof.

2.7.4 Proof of Lemma 7

Constructing a super-martingale sequence, M_t in the same way as we did in Theorem 3 and applying Doob's martingale inequality, we get

$$\begin{aligned}
\mathbf{Pr}_{t_j}\Big(\sup_{t\geq t_j}\Psi_t \geq 1-\epsilon_j\Big) &\leq \mathbf{Pr}_{t_j}\Big(\sup_{t\geq t_j} M_t \geq e^{s(1-\epsilon_j)}\Big)\\
&\leq \frac{\mathbb{E}\{M_{t_j}\}}{e^{s(1-\epsilon_j)}}\\
&= \frac{\mathbb{E}\big\{\exp\left(s\Psi_{t_j}+s\tau_{t_j}\right)\big\}}{e^{s(1-\epsilon_j)}}\\
&= \frac{\mathbb{E}\big\{\exp\left(s\Psi_{t_j}\right)\big\}\exp\left(s\tau_{t_j}\right)}{e^{s(1-\epsilon_j)}}.
\end{aligned}$$

Using Lemma 6 results in

$$\begin{aligned}
&\mathbf{Pr}_{t_j}\Big(\sup_{t\geq t_j}\Psi_t \geq 1-\epsilon_j\Big)\\
&\leq \frac{1}{e^{s(1-\epsilon_j)}}\exp\left(s(1-\epsilon_j) - s\epsilon_{j-1} + \Big(sc^2r^4+s^2c^2\sigma_N^2\Big)\Big(\frac{1}{t_{j-1}+L}-\frac{1}{t_j+L}\Big)+s\tau_{t_j}\right).
\end{aligned}$$

Substituting value of τ_j from Theorem 3 (see (2.24)) we have

$$\begin{aligned}
\mathbf{Pr}_{t_j}\Big(\sup_{t\geq t_j}\Psi_t \geq 1-\epsilon_j\Big) &\leq \exp\left(-s\epsilon_{j-1} + \Big(sc^2r^4+s^2c^2\sigma_N^2\Big)\Big(\frac{1}{t_{j-1}+L}-\frac{1}{t_j+L}\Big)\right.\\
&\qquad\left. + s\Big(c^2r^4+sc^2\sigma_N^2\Big)\frac{1}{t_j+L}\right)\\
&= \exp\left(-s\epsilon_{j-1} + s\Big(c^2r^4+sc^2\sigma_N^2\Big)\frac{1}{t_{j-1}+L}\right).
\end{aligned}$$

Substituting $s=(2/\epsilon_0)\ln(4/\delta)$, and using lower bound on L (verification in Proposition 5) we get

$$\mathbf{Pr}_{t_j}\Big(\sup_{t\geq t_j}\Psi_t \geq 1-\epsilon_j\Big) \leq \exp\Big(-\frac{s\epsilon_{j-1}}{2}\Big) = \left(\frac{\delta}{4}\right)^{\epsilon_{j-1}/\epsilon_0} \leq \frac{\delta}{2^{j+1}}.$$

Summing over j we finish the proof.

2.8 Proof of Supporting Lemmas for Final Epoch

Final epoch deals with the case when $\Psi_t \leq 1/2$. Following result captures the rate at which Ψ_t decreases during this phase.

2.8.1 Proof of Lemma 8

From Lemma 1 (Part 1)

$$\Psi_t \leq \Psi_{t-1} + 4\gamma_t^2\Big(\Big\|\frac{1}{B}\sum_{i=1}^{B}\mathbf{A}_{i,t} - \mathbf{\Sigma}\Big\|_F^2 + \Psi_{t-1}\phi\Big) - z_t.$$

Taking expectation over t conditioned on filteration $\mathcal{F}_{t-1}$, we get

$$\mathbb{E}\{\Psi_t|\mathcal{F}_{t-1}\} \leq \Psi_{t-1}(1+\gamma_t^2\phi) + \gamma_t^2\sigma_N^2 - \mathbb{E}\big\{z_t\big|\mathcal{F}_{t-1}\big\}.$$

Here, second term is due to Lemma 9. Now using upper bound on $-\mathbb{E}\big\{z_t\big|\mathcal{F}_{t-1}\big\}$ from [65, Lemma A.4] we get the following

$$\begin{aligned}\mathbb{E}\{\Psi_t|\mathcal{F}_{t-1}\} &\leq \Psi_{t-1}(1+\gamma_t^2\phi) + \gamma_t^2\sigma_N^2 - 2\gamma_t(\lambda_1-\lambda_2)\Psi_{t-1}(1-\Psi_{t-1})\\ &= \Psi_{t-1}\Big(1+\gamma_t^2\phi - 2\gamma_t(\lambda_1-\lambda_2)(1-\Psi_{t-1})\Big) + \gamma_t^2\sigma_N^2.\end{aligned}$$

Substituting $\gamma_t = c_0/(2(t+L)(\lambda_1-\lambda_2))$, using the fact that for $t > t_J$, $\Psi_{t-1} \leq 1/2$ and that we lie in sample space $\Omega_t^{'}$ with probability greater than $1-\delta$ (Theorem 3). Furthermore, $\Omega_t^{'}$ is $\mathcal{F}_{t-1}$-measurable, now taking expectation over $\Omega_t^{'}$ we have

$$\begin{aligned}\mathbb{E}_t\{\Psi_t\} &\leq \mathbb{E}_t\Bigg\{\Psi_{t-1}\Big(1+\frac{c_0^2\phi^2}{4(t+L)^2(\lambda_1-\lambda_2)^2} - \frac{c_0}{2(t+L)}\Big)\Bigg\} + \frac{c^2\sigma_N^2}{(t+L)^2}\\ &= \Bigg(1+\frac{c_0^2\phi^2}{4(t+L)^2(\lambda_1-\lambda_2)^2} - \frac{c_0}{2(t+L)}\Bigg)\mathbb{E}_t\{\Psi_{t-1}\} + \frac{c^2\sigma_N^2}{(t+L)^2}\\ &\leq \Bigg(1+\frac{c_0^2\phi^2}{4(t+L)^2(\lambda_1-\lambda_2)^2} - \frac{c_0}{2(t+L)}\Bigg)\mathbb{E}_{t-1}\{\Psi_{t-1}\} + \frac{c^2\sigma_N^2}{(t+L)^2}.\end{aligned}$$

Proposition 1. *For some constants $a_1, b > 0$, $a_2 > 1$, and $\forall t > t_J$, consider a nonnegative sequence $(u_t : t > t_J)$*

$$u_t \leq \Big(1+\frac{a_1}{(t+L)^2} - \frac{a_2}{t+L}\Big)u_{t-1} + \frac{b}{(t+L)^2},$$

then we have:

$$u_t \leq \Bigg(\frac{L+1}{t+L+1}\Bigg)^{a_2}\exp\Big(\frac{a_1}{L+1}\Big)u_0 + \frac{1}{(t+L+1)}\exp\big(\frac{a_1}{L+1}\big)\Big(\frac{L+2}{L+1}\Big)^2\frac{b}{a_2-1}.$$

Proof. Recursive application of u_t gives:

$$u_t \leq \Bigg(\prod_{i=t_J+1}^{t}\Big(1+\frac{a_1}{(i+L)^2} - \frac{a_2}{i+L}\Big)\Bigg)u_{t_0} + \sum_{i=t_J+1}^{t}\frac{b}{(i+L)^2}\Bigg(\prod_{j=i+1}^{t}\Big(1+\frac{a_1}{(j+L)^2} - \frac{a_2}{j+L}\Big)\Bigg). \tag{2.38}$$

Using [65, Lemma D.1] we can bound the product terms as

$$\prod_{j=i+1}^{t}\Big(1+\frac{a_1}{(j+L)^2}-\frac{a_2}{j+L}\Big) \le \exp\left(\sum_{j=i}^{t}\frac{a_1}{(j+L)^2}-\sum_{j=i}^{t}\frac{a_2}{j+L}\right)$$
$$\le \left(\frac{i+L+1}{t+L+1}\right)^{a_2}\exp\left(\sum_{j=i}^{t}\frac{a_1}{(j+L)^2}\right). \qquad (2.39)$$

Next, we bound the last term here as

$$\exp\left(\sum_{j=i}^{t}\frac{a_1}{(j+L)^2}\right) \le \exp\left(\int_{i+1}^{t+1}\frac{a_1}{(x+L)^2}dx\right) = \exp\Big(\frac{a_1}{i+L+1}-\frac{a_1}{t+L+1}\Big)$$
$$\le \exp\Big(\frac{a_1}{i+L+1}\Big).$$

Substituting this in (2.39) we get

$$\prod_{j=i+1}^{t}\Big(1+\frac{a_1}{(j+L)^2}-\frac{a_2}{j+L}\Big) \le \left(\frac{i+L+1}{t+L+1}\right)^{a_2}\exp\Big(\frac{a_1}{i+L+1}\Big).$$

Substituting this in (2.38) we get:

$$\begin{aligned}
u_t &\le \left(\frac{t_J+L+1}{t+L+1}\right)^{a_2}\exp\Big(\frac{a_1}{t_J+L+1}\Big)u_{t_J} + \sum_{i=t_J+1}^{t}\frac{b}{(i+L)^2}\left(\prod_{j=i+1}^{t}\Big(1+\frac{a_1}{(j+L)^2}-\frac{a_2}{j+L}\Big)\right)\\
&\le \left(\frac{t_J+L+1}{t+L+1}\right)^{a_2}\exp\Big(\frac{a_1}{t_J+L+1}\Big)u_{t_J} + \sum_{i=t_J+1}^{t}\frac{b}{(i+L)^2}\left(\frac{i+L+1}{t+L+1}\right)^{a_2}\exp\big(\frac{a_1}{i+L+1}\big)\\
&\le \left(\frac{t_J+L+1}{t+L+1}\right)^{a_2}\exp\Big(\frac{a_1}{t_J+L+1}\Big)u_{t_J} + \exp\Big(\frac{a_1}{t_J+L+1}\Big)\frac{b}{(t+L+1)^{a_2}}\sum_{i=1}^{t}\frac{(i+L+1)^{a_2}}{(i+L)^2}\\
&\le \left(\frac{t_J+L+1}{t+L+1}\right)^{a_2}\exp\Big(\frac{a_1}{t_J+L+1}\Big)u_{t_J}\\
&\qquad + \exp\Big(\frac{a_1}{t_J+L+1}\Big)\frac{b}{(t+L+1)^{a_2}}\Big(\frac{L+2}{L+1}\Big)^2\sum_{i=1}^{t}(i+L+1)^{a_2-2}.
\end{aligned}$$

Again applying [65, Lemma D.1] we get the final result

$$\begin{aligned}
u_t &\le \left(\frac{t_J+L+1}{t+L+1}\right)^{a_2}\exp\Big(\frac{a_1}{t_J+L+1}\Big)u_{t_J}\\
&\qquad + \exp\Big(\frac{a_1}{t_J+L+1}\Big)\frac{b}{(t+L+1)^{a_2}}\Big(\frac{L+2}{L+1}\Big)^2\frac{(t+L+1)^{a_2-1}}{a_2-1}\\
&= \left(\frac{t_J+L+1}{t+L+1}\right)^{a_2}\exp\Big(\frac{a_1}{t_J+L+1}\Big)u_{t_J} + \frac{1}{(t+L+1)}\exp\big(\frac{a_1}{t_J+L+1}\big)\Big(\frac{L+2}{L+1}\Big)^2\frac{b}{a_2-1}
\end{aligned}$$

□

2.9 Other Results

Proposition 2 (Bennett's Inequality [67])**.** *Consider a zero-mean bounded random variable $|x_i| \leq h$ with variance σ_i^2, then for any $s \in \mathbb{R}$ we have*

$$\mathbb{E}\{e^{sx_i}\} \leq \exp\left(\sigma_i^2 s^2 \Big(\frac{e^{sh} - 1 - sh}{(sh)^2}\Big)\right).$$

Proposition 3. *For a network of N nodes and $t > 0$, we have the following upper bound on conditional moment generating function of Ψ_t*

$$\mathbb{E}\{\exp(s\Psi_t)|\mathcal{F}_{t-1}\} \leq \exp\left(s\Psi_{t-1} - s\mathbb{E}\{z_t|\mathcal{F}_{t-1}\} + s\gamma_t^2 r^4 + s^2\gamma_t^2\sigma_N^2\right), \tag{2.40}$$

Proof. In order to prove this proposition we need to show that using lower bound on t_0 and the upper bound on $\|x_t\|_2$, we have $sh \leq 7/4$. Now depending on value of s we need to prove the result for two cases in the following.

<u>*Case i:*</u> For $s = d/4\epsilon$, substituting value of h we get

$$\begin{aligned} sh &= \frac{d\gamma_t r^2}{2\epsilon} = \frac{dcr^2}{2(t+L)\epsilon} \leq \frac{dcr^2}{2L\epsilon} \leq \frac{dcr^2}{2\epsilon L_1} \\ &\overset{(a)}{\leq} \frac{dcr^2}{2\epsilon} \frac{\epsilon}{8dr^4 \max(1,c^2)\ln(4/\delta)} \leq \frac{1}{16\ln(4/\delta)} \leq \frac{7}{4}. \end{aligned}$$

<u>*Case ii:*</u> For $s = (2/\epsilon_0)\ln(4/\delta)$ we have:

$$\begin{aligned} sh &= \frac{2\ln(4/\delta)cr^2}{\epsilon_0(t+L)} \leq \frac{2\ln(4/\delta)cr^2}{\epsilon_0 L_1} \\ &\leq \frac{2\ln(4/\delta)cr^2}{\epsilon_0} \frac{\epsilon_0}{8r^4\max(1,c^2)\ln\frac{4}{\delta}} \leq \frac{1}{4} \leq \frac{7}{4}. \end{aligned}$$

□

Proposition 4. *For*

$$L \geq \frac{8dr^4\max(1,c^2)}{\epsilon}\ln\frac{4}{\delta} + \frac{8d^2\sigma_N^2\max(1,c^2)}{\epsilon^2}\ln\frac{4}{\delta}, \tag{2.41}$$

we have

$$\frac{c^2}{L}\Big(r^4 + s\sigma_N^2\Big) \leq \frac{\epsilon}{d}. \tag{2.42}$$

Proof. We will prove this by proving the following two statements:

$$\frac{c^2r^4}{t_0} \leq \frac{c^2r^4}{L_1} \leq \frac{\epsilon}{2d} \quad \text{and} \quad \frac{sc^2\sigma_N^2}{L} \leq \frac{sc^2\sigma_N^2}{L_2} \leq \frac{\epsilon}{2d}.$$

We start by proving the first statement:

$$\frac{c^2r^4}{L_1} \leq c^2r^4\frac{\epsilon}{8dr^4\max(1,c^2)\ln\frac{4}{\delta}} \leq \frac{\epsilon}{2d},$$

Now we prove the second statement as follows:

$$\frac{c^2 s\sigma_N^2}{L_2} \le \frac{c^2 d\sigma_N^2}{4\epsilon}\frac{\epsilon^2}{8d^2\sigma_N^2\max(1,c^2)\ln\frac{4}{\delta}} \le \frac{\epsilon}{2d}.$$

□

Proposition 5. *For* $L \ge \frac{8r^4\max(1,c^2)}{\epsilon_0}\ln\frac{4}{\delta} + \frac{8\sigma_N^2\max(1,c^2)}{\epsilon_0^2}\ln\frac{4}{\delta}$*, we have*

(a) $\frac{c^2r^4}{(t_{j-1}+L)} \le \frac{\epsilon_0}{4}$*, and*

(b) $\frac{2c^2\sigma_N^2}{\epsilon_0(t_{j-1}+L)}\ln\frac{4}{\delta} \le \frac{\epsilon_0}{4}$.

Proof.

$$\begin{aligned}\frac{c^2r^4}{(t_{j-1}+L)} \le \frac{2c^2r^4}{L} &\le \frac{2c^2r^4}{L_1}\\ &\le 2c^2r^4\frac{\epsilon_0}{8r^4\max(1,c^2)\ln\frac{4}{\delta}} \le \frac{\epsilon_0}{4}.\end{aligned}$$

Next we prove the second statement as follows:

$$\frac{2c^2\sigma_N^2}{\epsilon_0(t_{j-1}+L)}\ln\frac{4}{\delta} \le \frac{2c^2\sigma_B^2}{\epsilon_0 L}\ln\frac{4}{\delta} \le \frac{2c^2\sigma_N^2}{\epsilon_0 L_2}\ln\frac{4}{\delta} \le \frac{2c^2\sigma_N^2}{\epsilon_0}\ln\frac{4}{\delta}\frac{\epsilon_0^2}{8\sigma_N^2\max(1,c^2)\ln(4/\delta)} \le \frac{\epsilon_0}{4}.$$

□

2.10 Numerical Results

In this section we will provide numerical results to show the effectiveness of distributed mini-batch Krasulina's method (Algorithm 2). To support our theoretical findings, numerical results in this section will highlight the performance of Krasulina's method in terms of important problem parameters for streaming PCA, namely, eigengap ($\lambda_1 - \lambda_2$), dimensionality (d), upper bound on inputs ($\|\mathbf{x}_t\|_2 \le r$), and variance in samples (σ_B^2). In the following we first provide experiments on synthetic data to highlight the impact of $\lambda_1 - \lambda_2$, r, and d on the performance of Algorithm 2, then we will provide detailed experiments on synthetic and real-world data to show the improvement in convergence rate using mini-batches which is the main contribution of this chapter. For mini-batching experiments the goal is to show that for some fixed number of samples T Algorithm 2 can achieve same final error in top eigenvector estimate as the classical sample-by-sample approach if mini-batch size B is not too large.

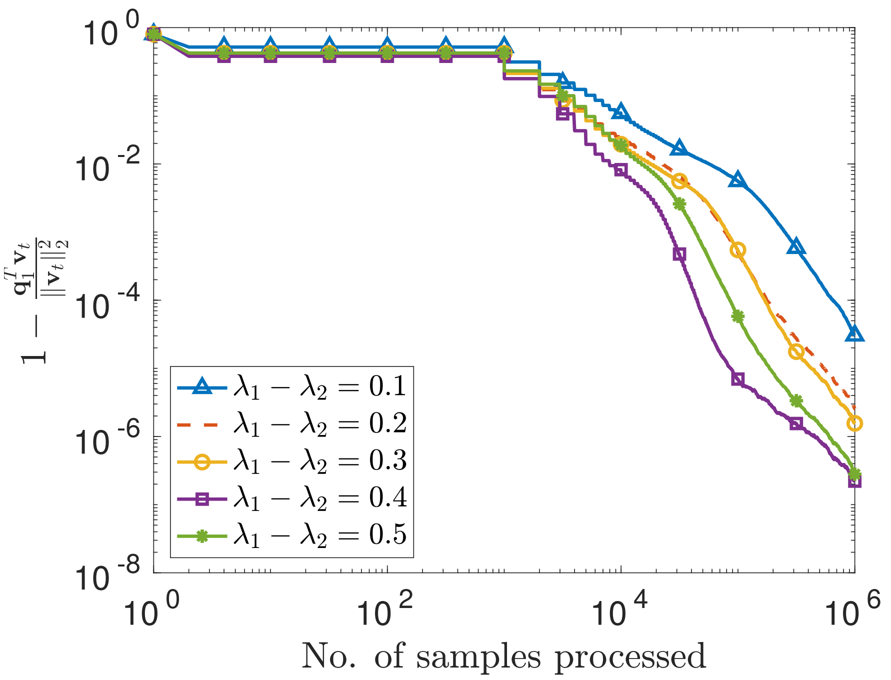


Figure 2.2: Impact eigengap $\lambda_1 - \lambda_2$ on the convergence of Krasulina's method.

2.10.1 Synthetic Data

In the following experiments we generate $T = 10^6$ samples from some probability distribution and for each experiment we perform 200 Monte-Carlo trials. In all the experiments in the following we use step size of the form $\gamma_t = c/t$. To choose value of c we perform simulations with different values of c and plot the results with the value of c giving the best convergence rate. Specific details of setup for each experiment are given in the following as well.

Impact of the eigengap on the performance of Algorithm 2

For this experiment we generate data in $\mathbb{R}^5$ from a normal distribution, $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$. Here, covariance matrix, $\mathbf{\Sigma}$, has largest eigenvalue $\lambda_1 = 1$. We vary the values of eigengap from $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. Values of c giving the best convergence rates are $c = \{180, 110, 90, 70, 60\}$. We can observe from Figure 2.2 that after observing $T = 10^6$ the final gap in error for different values of eigengap is on the order of $O(1/(\lambda_1 - \lambda_2)^2)$ as predicted by the theoretical guarantees.

Impact of dimension on the performance of Algorithm 2

For this experiment we generate data from a normal distribution, $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ with largest eigenvalue of the covariance matrix, $\mathbf{\Sigma}$, to be $\lambda_1 = 1$. We fix the eigengap to 0.2 and vary the

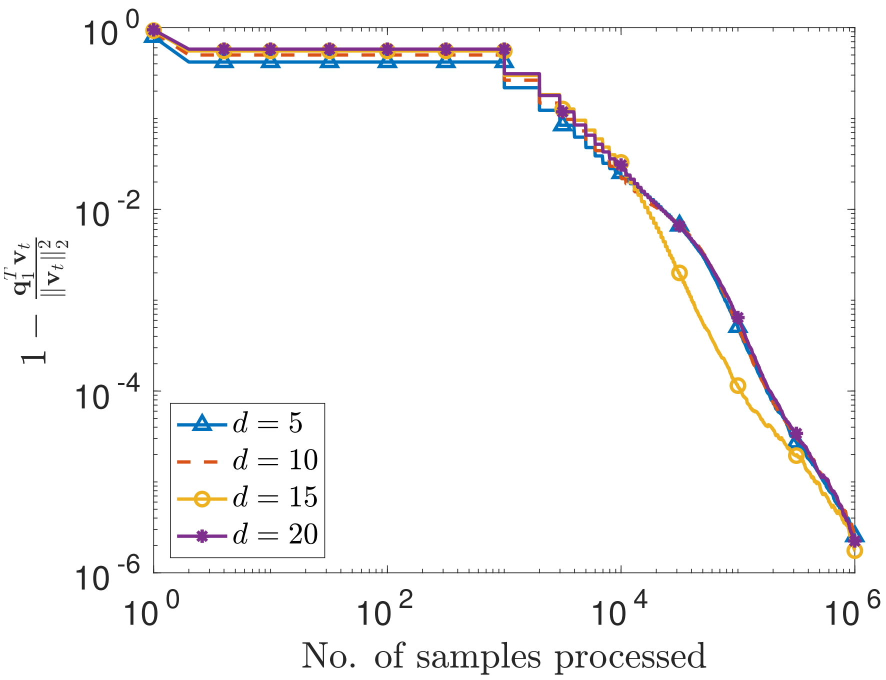


Figure 2.3: Convergence of Krasulina's method with varying dimensionality.

dimension to $d = \{5, 10, 15, 20\}$. Values of c providing the best convergence rate in this case are $\{110, 110, 100, 100\}$. Convergence behavior of Algorithm 2 is give in Figure 2.3. These two observations show that dimension does not impact convergence as well as the choice of step size which is not aligned with our theoretical guarantees. The main implication here is that our theoretical results are not tight in terms of dimensions.

Impact of upper bound on the performance of Algorithm 2

In order to demonstrate the impact of the upper bound $\|x_t\|_2 \leq r$ on the convergence of Krasulina's method we generate data from Uniform distribution $\mathcal{U}(-a, a)$. Values of a are varied from $\{1, 2, 3, 10\}$. These values of a approximately result in $r = \{1.45, 2.9, 4.5, 14.5\}$. Values of c providing best convergence are $c = \{8, 2, 1, 0.08\}$. Convergence behavior in Figure 2.4 shows that if we vary r and all the other parameters are kept constant then the convergence behavior will stay the same. The value of r appears in the convergence results in a lower bound on L (see (2.12)) and in the non-dominant term in the error bound. Hence, although our results are not independent of r its dependence is only through a non-dominant term. Furthermore, the dependence of value of L on upper bound r reflects in the choice of step size parameter c in our experiments here as well.

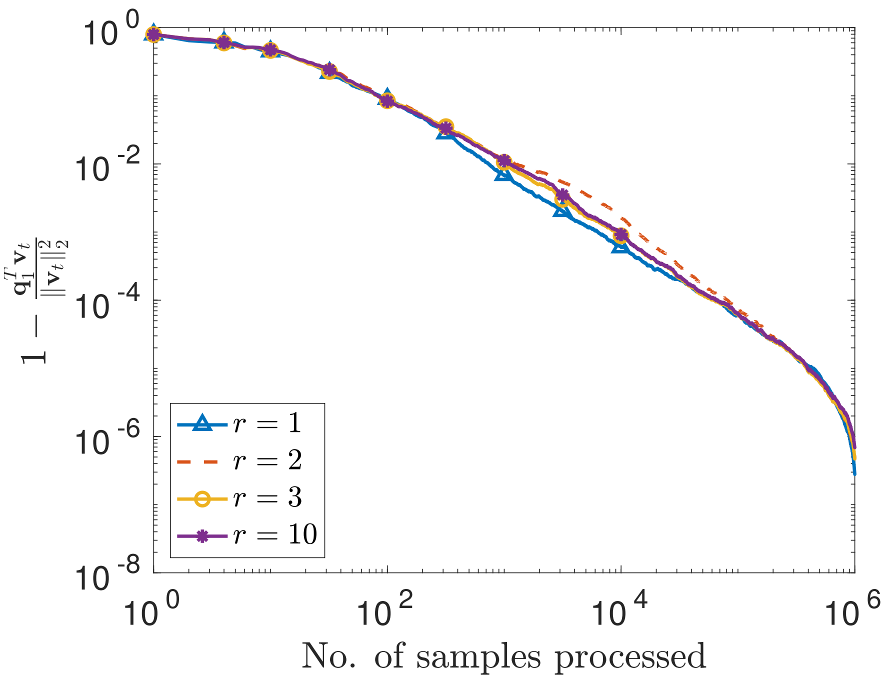


Figure 2.4: Convergence of Krasulina's method with varying upper bound on the inputs.

Impact of minibatching on the performance of Algorithm 2

For this experiment we generate data in $\mathbb{R}^5$ from a normal distribution, $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$. Here, covariance matrix, $\mathbf{\Sigma}$, has largest eigenvalue $\lambda_1 = 1$ and eigengap is, $\lambda_1 - \lambda_2 = 0.2$. We generate $T = 10^6$ samples from this distribution and perform 200 Monte-Carlo trials of experiment. We use mini-batches of sizes $B = 1$, $B = 10$, $B = 100$, $B = 500$, $B = 1000$, and $B = 2000$ in our experiments. For each value of mini-batch we use step size of the form c/t and we provide results here with values of c which give us the best convergence rates in Figure 2.5(a). Values of c resulting in best performance for these step sizes are 70, 80, 80, 90, 110, and 100 respectively. Results of these experiments are shown in Figure 2.5(a). From these results we can see that we achieve the same final error as we increase the batch size from 1 to 100 and it gets worse for $B = 1000$. Hence, as suggested by our analysis, for some fixed number of samples T, we will not achieve optimal error if mini-batch size is too large.

Next, we demonstrate the performance of Algorithm 2 with delays in distributed computation. We use similar data generation setup as in Section 2.10.1 and for a network of 10 nodes ($N = 10$) we use mini-batch of size $B = 100$. For comparison we also plot the error curve for case when we discard $1/N$ of total samples, i.e., we perform mini-batching using 100 samples that are available locally at any given node. We vary the delay in distributed computation such that the number of discarded samples is $\mu = \{10, 100, 200\}$. From Figure 2.5(b) we can see that the estimation error achieved after seeing T samples does not increase much as we increase the number of data samples lost due to resource constraints.

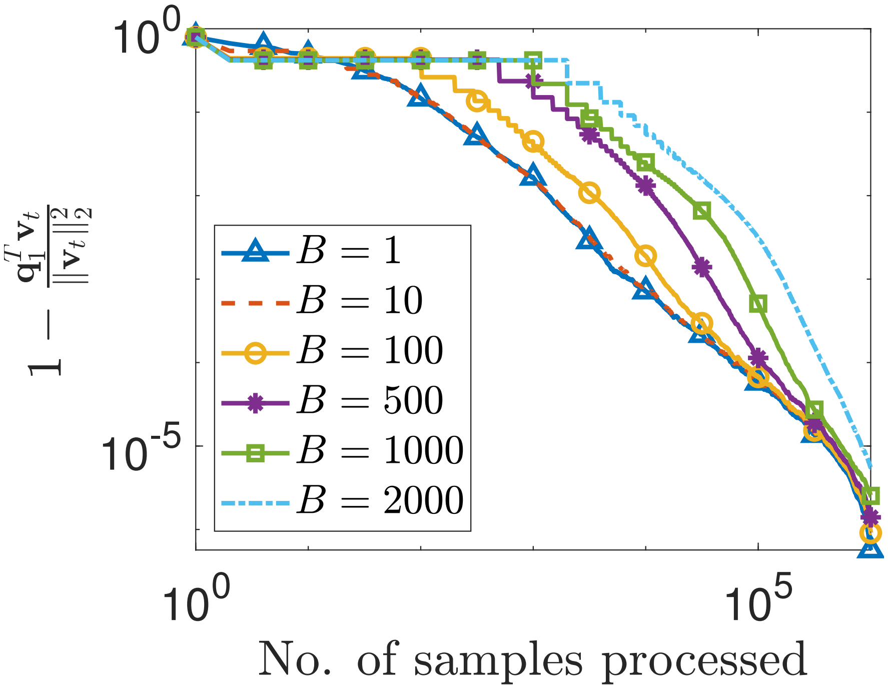
2.10.2 Impact of mini-batching on Real-world Datasets

In this section, we demonstrate the performance of Algorithm 2 over two real-world datasets: the Higgs dataset [2] and MNIST dataset [1]. Among these, Higgs dataset consists of 1.1×10^7 samples with $d = 28$ and the MNIST dataset consists of roughly 6×10^4 samples with $d = 784$.

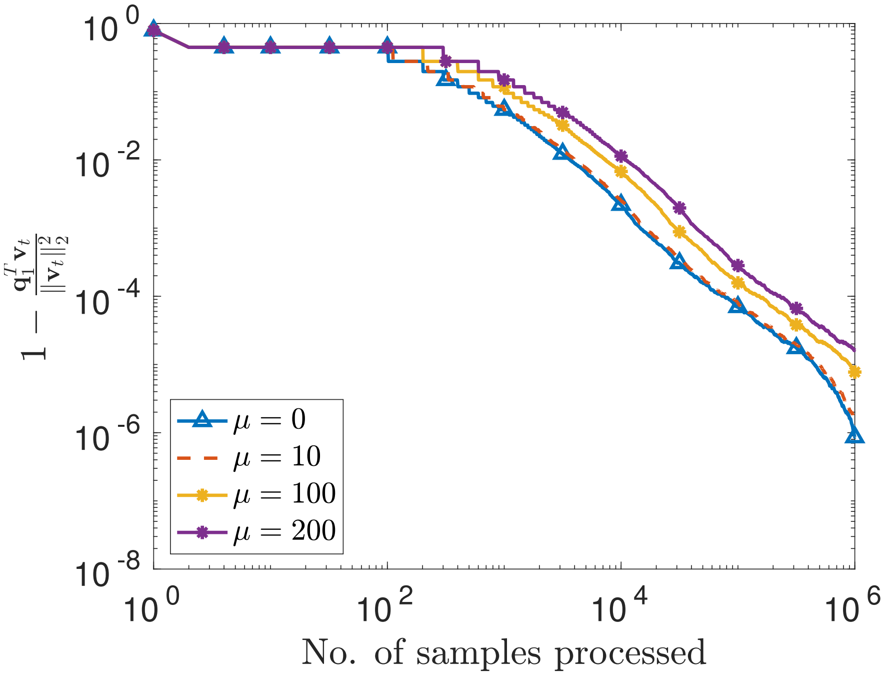
Results for the MNIST dataset are given in Figure 2.7. For the MNIST dataset we use step size $\gamma = c/t$, with $c = 0.6, 0.9, 1.1, 1.5, 1.6$ for mini-batch of sizes $B = \{1, 10, 100, 300, 1000\}$. Results in Figure 2.6 show that for MNIST dataset the final error achieved decreases as we change the batch size from $B = 1$ to $B = 100$ while it starts increasing as we change batch size to $B = 300$ and $B = 1000$. To demonstrate the impact of latency on the convergence of Algorithm 2 for MNIST dataset we fix batch size to $B = 100$ and consider latency such that number of discarded samples is $\mu = \{0, 10, 20, 40, 100\}$. Here, we can see that error increases as increase $\mu = B$ which is what our theory predicted as well. For Higgs dataset we used $c = 0.07$ and mini-batch of sizes, $B = \{1, 10^2, 10^3, 10^4, 2 \times 10^4\}$. We can see the same trend here with suboptimal rate for batches of size $B = 10^4$ and $B = 2 \times 10^4$, while all the other batch-size choices give us $O(1/t)$ rate. Furthermore, with latency in computing network wide average we observe that for batch size $B = 10^3$ we achieve optimal error rate as far as we are discarding up to 10^3 samples per iteration round.

2.11 Conclusion

In this chapter, we proposed algorithms (D-Krasulina and DM-Krasulina) for computing the top eigenvector of a covariance matrix in high rate streaming settings. We further showed theoretically that the proposed algorithms achieve the order optimal convergence rate in high rate streaming settings where the classical single processor approaches will not be optimal anymore. Finally, with the help of synthetic and real world data we demonstrate the efficacy of the proposed methods.

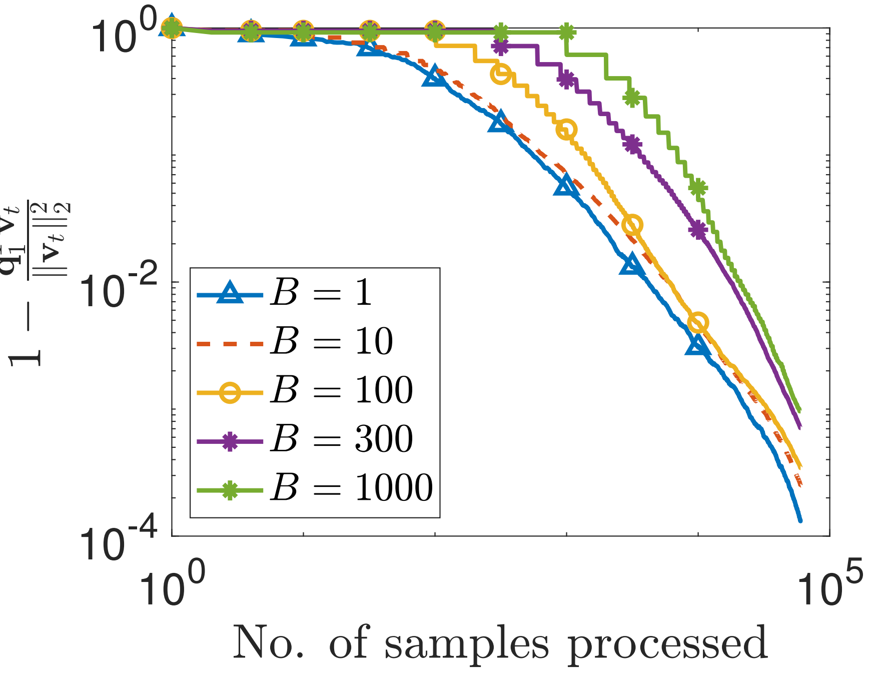


(a) Impact of batch size on convergence of Krasulina's method.

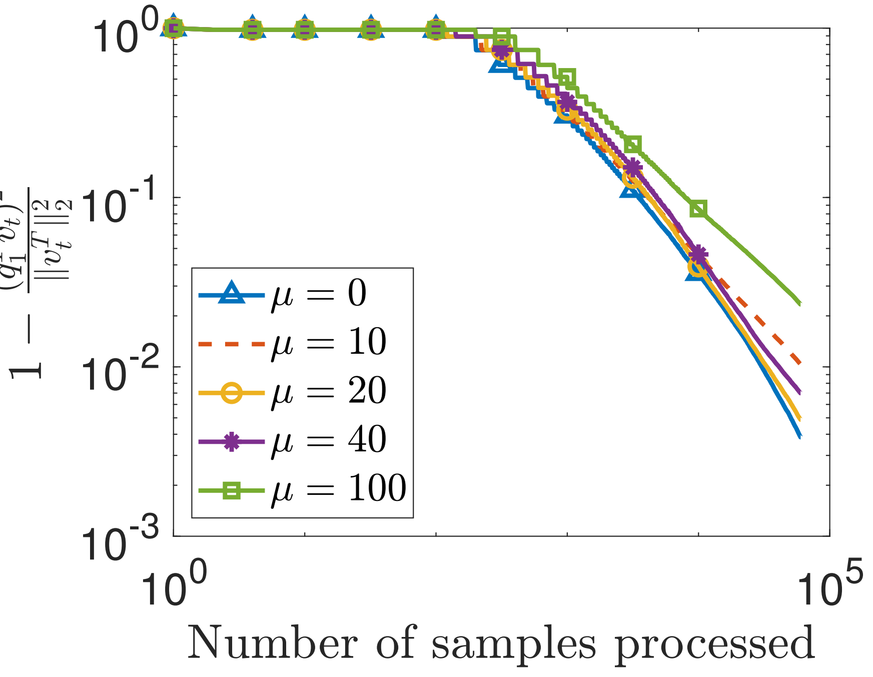


(b) Performance of Krasulina's method for a mini-batch of size $B = 100$ with latency in distributed computation.

Figure 2.5: Impact of batch size on convergence of Krasulina's method for synthetic data.

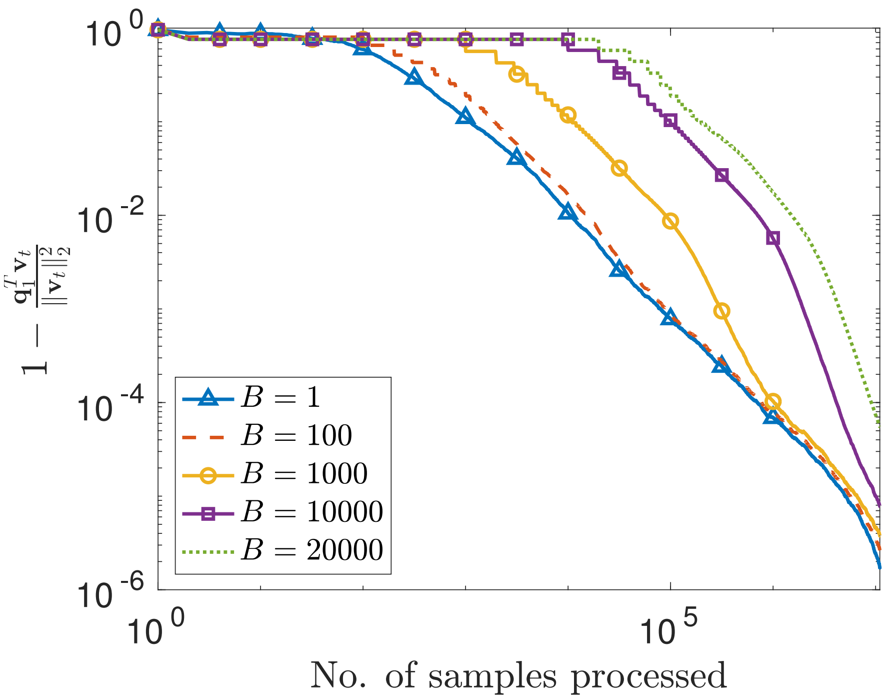


(a) Fast streaming data arriving at a single processor, we only process T/N samples and discard rest of the samples.

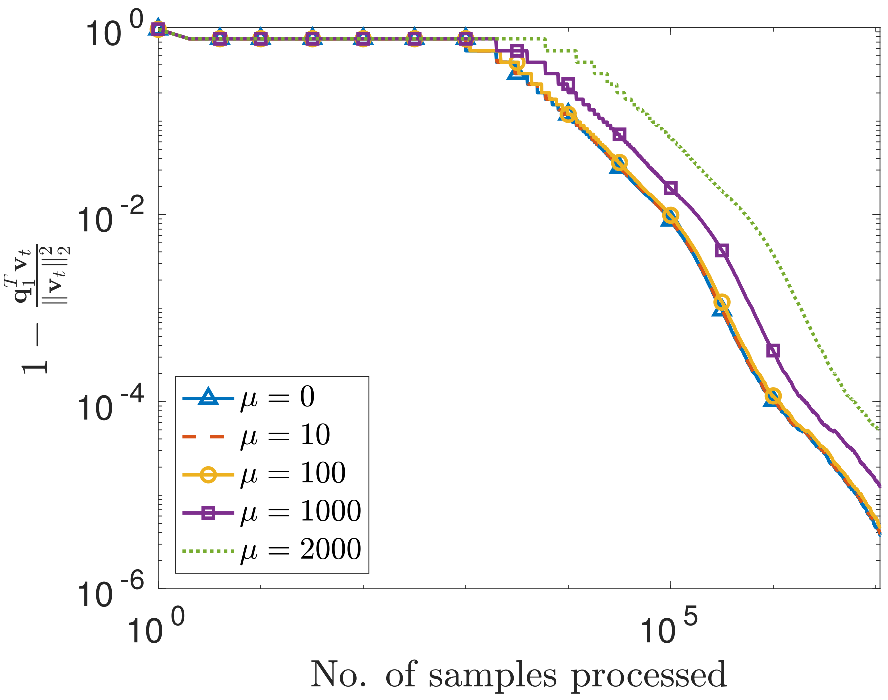


(b) Convergence of distributed Mini-batch Krasulina's method while increasing the latency such that number of discarded samples per iterate update is $\mu = \{0, 10, 20, 40\}$.

Figure 2.6: Impact of batch size on convergence of Krasulina's method for MNIST dataset [1].



(a) Fast streaming data arriving at a single processor, we only process T/N samples and discard rest of the samples.



(b) Convergence of distributed Mini-batch Krasulina's method while increasing the latency such that number of discarded samples per iterate update is $\mu = \{0, 10, 20, 40\}$.

Figure 2.7: Performance of distributed mini-batch Krasulina's method for Higgs dataset [2].

Chapter 3

Cloud K-SVD: A Decentralized Dictionary Learning Algorithm for Big, Distributed Data

This chapter studies the problem of data-adaptive representations for big, distributed data. It is assumed that a number of geographically-distributed, interconnected sites have massive local data and they are interested in collaboratively learning a low-dimensional geometric structure underlying these data. In contrast to previous works on subspace-based data representations, this work focuses on the geometric structure of a union of subspaces (UoS). In this regard, it proposes a decentralized algorithm—which we call cloud K-SVD—for collaborative learning of a UoS structure underlying distributed data of interest. The goal of cloud K-SVD is to learn a common overcomplete dictionary at each individual site such that every sample in the distributed data can be represented through a small number of atoms of the learned dictionary. Cloud K-SVD accomplishes this goal without requiring exchange of individual samples between sites. This makes it suitable for applications where sharing of raw data is discouraged due to either privacy concerns or large volumes of data. This chapter also provides an analysis of cloud K-SVD that gives insights into its properties as well as deviations of the dictionaries learned at individual sites from a centralized solution in terms of different measures of local/global data and topology of interconnections. Finally, the efficacy of cloud K-SVD is illustrated through numerical simulations on real and synthetic data.

3.1 Motivation

Modern information processing is based on the axiom that while real-world data may live in high-dimensional ambient spaces, relevant information within them almost always lies near low-dimensional geometric structures. Knowledge of these (low-dimensional) geometric structures underlying data of interest is central to the success of a multitude of information processing tasks. But this knowledge is unavailable to us in an overwhelmingly large number of applications and a great deal of work has been done in the past to *learn* the geometric structure of data from the data *themselves*. Much of that work, often studied under rubrics such as *principal component analysis* (PCA) [68], *generalized PCA* [69], *hybrid linear modeling* [70], and *dictionary learning* [4, 71, 72], has been focused on centralized settings in which the entire data are assumed available at a single location. In recent years, there has been some effort to extend these works to decentralized settings; see, e.g., [73–84]. The setup considered in some of these works is that each distributed entity is responsible for either some dimensions of the data [73–75] or some part of the learned geometric structure [74, 75, 80]. Other works in this direction also focus on learning under the assumption of data lying near *(linear) subspaces* [73–77], require extensive communications among the distributed entities [78], and ignore some of the technical details associated with processing among distributed entities having interconnections described by graphs of arbitrary, unknown topologies [76–79].

In this chapter, we are interested in a setting in which a number of geographically-distributed sites have massive local data and these sites are interested in collaboratively learning a geometric structure underlying their data by communicating among themselves over public/private networks. The key constraints in this problem that distinguish it from some of the prior works are: (*i*) sites cannot communicate "raw" data among themselves; (*ii*) interconnections among sites are not described by a complete graph; and (*iii*) sites do not have knowledge of the global network topology. All these constraints are reflective of the future of big, distributed data, some of the examples include medical data [22], object detection in videos [9], images on social media [85, 86], etc. . In particular, the first constraint is justified because of the size of local data compilations as well as privacy concerns in the modern age. Similarly, the latter two constraints are justified because linking geographically-distributed sites into a complete graph can be cost-prohibitive and since enterprises tend to be protective of their internal network topologies.

3.1.1 Our Contributions

The first main contribution of this chapter is formulation of a decentralized method, which we call *cloud K-SVD*, that enables data-adaptive representations in distributed settings. In contrast to works that assume a linear geometric structure for data [73–77], cloud K-SVD is based on the premise that data lie near a *union* of low-dimensional subspaces. The *union-of-subspaces* (UoS) model is a nonlinear generalization of the subspace model [87]. The task of learning the UoS underlying data of interest from data themselves is often termed *dictionary learning* [4, 71, 72], which involves data-driven learning of an overcomplete dictionary such that every data sample can be approximated through a small number of atoms of the dictionary. Dictionary learning—when compared to linear data-adaptive representations such as the PCA and the linear discriminant analysis [88]—has been shown to be highly effective for tasks such as compression [71], denoising [89], object recognition [90], and inpainting [91]. Cloud K-SVD, as the name implies, is a decentralized variant of the popular dictionary learning algorithm K-SVD [4] and leverages a classical iterative eigenvector estimation algorithm, the *power method* [31, Ch. 8], and consensus averaging [92] for collaborative dictionary learning.

The second main contribution of this chapter is a rigorous analysis of cloud K-SVD that gives insights into its properties as well as deviations of the dictionaries learned at individual sites from the centralized K-SVD solution in terms of different measures of local/global data and topology of the interconnections. Using tools from linear algebra, convex optimization, matrix perturbation theory, etc., our analysis shows that—under identical initializations—the dictionaries learned by cloud K-SVD come arbitrarily close to the one learned by (centralized) K-SVD as long as appropriate number of power method and consensus iterations are performed in each iteration of cloud K-SVD. Finally, the third main contribution of this chapter involves numerical experiments on synthetic and real-world data that demonstrate both the efficacy of cloud K-SVD and the usefulness of collaborative dictionary learning over dictionary learning only using data available at the individual sites.

3.1.2 Relationship to Previous Work

Some of the earliest works in decentralized processing date back nearly three decades [93, 94]. Since then a number of decentralized methods have been proposed for myriad tasks. Some recent examples of this that do not involve a centralized *fusion center* include decentralized methods for classification [95–97], localization [98, 99], linear regression [100], and (multitask) estimation [83, 84, 101]. But relatively little attention has been paid to the problem of data-driven decentralized learning of the geometric structure of data. Notable exceptions to this include [75–82]. While our work as well as [75–79] rely on consensus averaging for computing the underlying geometric structure, we are explicit in our formulation that perfect consensus under arbitrary, unknown topologies cannot be achieved. In contrast, developments in [75–79] assume that there is no numerical error due to consensus averaging which means performing infinite consensus iterations which is not feasible in practice . Further, [75–77] assume a subspace data model, while [78] advocates the use of consensus averaging for computing sample covariance—an approach that requires extensive communications among the distributed entities.

Our work is most closely related to that in [80–82], which also study dictionary learning in distributed settings. But [80] focuses only on learning parts of the dictionary at each site as opposed to the setup of this chapter in which we are interested in learning a complete dictionary at each site. While this chapter and [81, 82] share the same setup, our work as well as [82] are fundamentally different from [81]. The method proposed in [81] involves learning local dictionaries at different sites and then *diffusing* these local dictionaries to obtain a global dictionary. In contrast, our work and [82] are based on the centralized K-SVD algorithm, which is known to be superior to other dictionary learning methods [4], and involve updating each atom of the local dictionaries in a collaborative fashion. The difference between this work and [82] lies in the fact that cloud K-SVD uses a decentralized variant of the power method to update each atom, whereas [82] relies on decentralized optimization for this purpose. This helps us rigorously analyze the performance of cloud K-SVD, whereas no such analysis is provided in [82].

We conclude by noting that the decentralized power method component of cloud K-SVD has similarities with the work in [76, 102]. However, unlike [76, 102], we do not assume perfect consensus during iterations of the power method, which leaves open the question of convergence of the decentralized variant of the power method. While analyzing cloud K-SVD, we in fact end up addressing this question also. That part of our analysis is reminiscent of the one carried out in [103] in the context of convergence behavior of decentralized eigenanalysis of a network using a power method-like iterative algorithm. However, there are fundamental differences in the analysis of [103] and our work because the exact place where consensus averaging is carried out differs in the two applications.

3.1.3 Notation and Chapter Organization

We use lower-case letters to represent scalars (v), while we use boldface lower-case letters to represent vectors ($\mathbf{v}$), and boldface upper-case letters to represent matrices $\mathbf{V}$. The operator $\text{sgn} : \mathbb{R} \to \{+1, -1\}$ is defined as $\text{sgn}(x) = x/|x|$, while $\text{supp}(\mathbf{v})$ returns indices of the nonzero entries in vector $\mathbf{v}$. Superscript $(\cdot)^{\mathrm{T}}$ denotes the transpose operation, $\|\cdot\|_0$ counts the number of nonzero entries in a vector, $\|\mathbf{v}\|_p$ denotes the usual ℓ_p norm of vector $\mathbf{v}$, and $\langle \mathbf{u}, \mathbf{v} \rangle$ denotes the inner product between vectors $\mathbf{u}$ and $\mathbf{v}$. Given a set $\mathcal{I}$, $\mathbf{v}_{|\mathcal{I}}$ and $\mathbf{A}_{|\mathcal{I}}$ denote a subvector and a submatrix obtained by retaining entries of vector $\mathbf{v}$ and columns of matrix $\mathbf{A}$ corresponding to the indices in $\mathcal{I}$, respectively, while $\|\mathbf{A}\|_2$, $\|\mathbf{A}\|_F$, and $\|\mathbf{A}\|_{\max}$ denote the operator norm, Frobenius norm, and max norm (i.e., maximum absolute value) of matrix $\mathbf{A}$, respectively. Given matrices $\{\mathbf{A}_i \in \mathbb{R}^{n_i \times m_i}\}_{i=1}^N$, $\text{diag}\{\mathbf{A}_1, \ldots, \mathbf{A}_N\}$ denotes a block-diagonal matrix $\mathbf{A} \in \mathbb{R}^{\sum n_i \times \sum m_i}$ that has $\mathbf{A}_i$'s on its diagonal. Finally, given a matrix $\mathbf{A}$, $\mathbf{a}_j$ and $\mathbf{a}_{j,T}$ denote the j^{th} column and the j^{th} row of $\mathbf{A}$, respectively.

The rest of this chapter is organized as follows. In Section 3.2, we formulate the problem of collaborative dictionary learning from big, distributed data. In Section 3.3, we describe the cloud K-SVD algorithm. In Section 3.4, we provide an analysis of cloud K-SVD algorithm. We provide some numerical results in Section 3.5 and proofs of the main theorems are provided in Sections 3.6–3.9. Finally, concluding remarks are given in Section 3.10.

3.2 Problem Formulation

In this chapter, we consider a collection of N geographically-distributed sites that are interconnected to each other according to a fixed topology. Here, we use "site" in the broadest possible sense of the term, with a site corresponding to a single computational system (e.g., sensor, drone, smartphone, tablet, server, database), a collection of co-located computational systems (e.g., data center, computer cluster, robot swarm), etc. Mathematically, we represent this collection and their interconnections through an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{1, 2, \cdots, N\}$ denotes the sites and $\mathcal{E}$ denotes edges in $\mathcal{G}$ with $(i, i) \in \mathcal{E}$, while $(i, j) \in \mathcal{E}$ whenever there is a connection between sites i and j. The only assumption we make about the topology of $\mathcal{G}$ is that it is a connected graph.

Next, we assume each site i has a collection of local data, expressed as a matrix $\mathbf{Y}_i \in \mathbb{R}^{n \times S_i}$ with S_i representing the number of data samples at the i^{th} site. We can express all this distributed data into a single matrix $\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_1 & \ldots & \mathbf{Y}_N \end{bmatrix} \in \mathbb{R}^{n \times S}$, where $S = \sum_{i=1}^N S_i$ denotes the total number of data samples distributed across the N sites; see Figure 3.1 for a schematic representation of this. In this setting, the fundamental objective is for each site to collaboratively learn a low-dimensional geometric structure that underlies the global (distributed) data $\mathbf{Y}$. The basic premises behind collaborative structure learning of global data, as opposed to local structure learning of local data, are manifold. First, since the number of global samples is much larger than the number of local samples, we expect that collaborative learning will outperform local learning for data representations. Second, local learning will be strictly suboptimal for some sites in cases where sampling density, noise level, fraction of outliers, etc., are not uniform across all sites. Collaborative learning, on the other hand, will even out such nonuniformities within local data.

Our main assumption is that the low-dimensional geometric structure underlying the global data corresponds to a union of T_0-dimensional subspaces in $\mathbb{R}^n$, where $T_0 \ll n$. One possible means of learning such a structure is studied under the moniker dictionary learning, which learns an *overcomplete dictionary* $\mathbf{D}$ such that each data sample is well approximated by no more than T_0 columns (i.e., *atoms*) of $\mathbf{D}$ [4, 71, 72]. Assuming the global data $\mathbf{Y}$ is available at a centralized location, this problem of dictionary learning can be expressed as

$$\big(\mathbf{D}, \mathbf{X}\big) = \arg\min_{\mathbf{D},\mathbf{X}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 \text{ s.t. } \forall s, \|\mathbf{x}_s\|_0 \leq T_0, \tag{3.1}$$

where $\mathbf{D} \in \mathbb{R}^{n \times K}$ with $K > n$ is an overcomplete dictionary having unit ℓ_2-norm columns, $\mathbf{X} \in \mathbb{R}^{K \times S}$ corresponds to representation coefficients of the data having no more than $T_0 \ll n$ nonzero coefficients per sample, and $\mathbf{x}_s$ denotes the s^{th} column in $\mathbf{X}$. Note that (3.1) is non-convex in $\big(\mathbf{D}, \mathbf{X}\big)$, although it is convex in $\mathbf{D}$ alone. One of the most popular approaches to solving (3.1) involves alternate minimization in which one alternates between solving (3.1) for $\mathbf{D}$ using a fixed $\mathbf{X}$ and then solving (3.1) for $\mathbf{X}$ using a fixed $\mathbf{D}$ [4, 104].

Unlike classical dictionary learning, however, we do not have the global data $\mathbf{Y}$ available at a centralized location. Data aggregation either at a centralized location or at any one of the individual sites is also impractical due to communications and storage costs of big data. Furthermore, privacy issues may also preclude aggregation of data. Instead, our goal is to have individual sites collaboratively learn dictionaries $\{\widehat{\mathbf{D}}_i\}_{i \in \mathcal{N}}$ from global data $\mathbf{Y}$ such that these *collaborative dictionaries* are close to a dictionary $\mathbf{D}$ that could have been learned from $\mathbf{Y}$ in a centralized fashion. In the following section, we present a decentralized variant of a popular dictionary learning algorithm that accomplishes this goal without exchanging raw data between sites. This is followed by a rigorous analysis of the proposed algorithm in Section 3.4, which establishes that the collaborative dictionaries learned using our proposed algorithm can indeed be made to come arbitrarily close to a centralized dictionary.

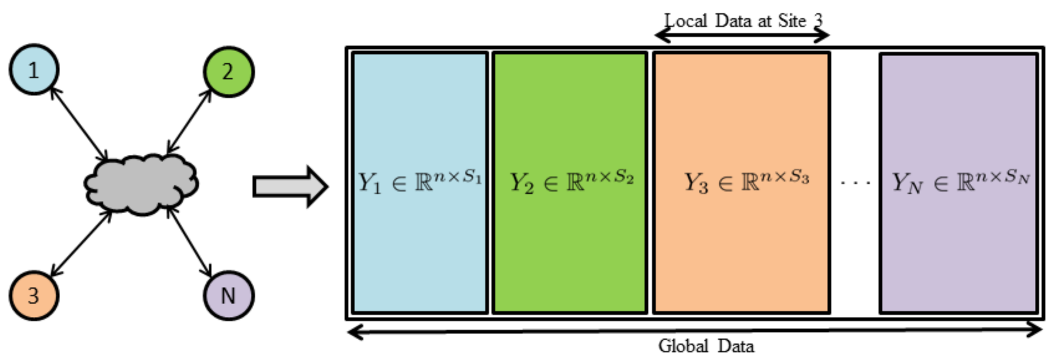

Local Data at Site 3
1
2
3
N
$Y_1 \in \mathbb{R}^{n \times S_1}$
$Y_2 \in \mathbb{R}^{n \times S_2}$
$Y_3 \in \mathbb{R}^{n \times S_3}$
. . .
$Y_N \in \mathbb{R}^{n \times S_N}$
Global Data

Figure 3.1: A schematic representing global data $\mathbf{Y}$ distributed across N sites. Here, n denotes the dimension of each data sample, while S_i denotes the total number of data samples available at the i^{th} site.

3.3 Cloud K-SVD

In this chapter, we focus on the K-SVD algorithm [4] as the basis for collaborative dictionary learning. We have chosen to work with K-SVD because of its iterative nature and its reliance on the singular value decomposition (SVD), both of which enable its exploitation for distributed purposes. In the following, we first provide a brief overview of K-SVD, which is followed by presentation of our proposed algorithm—termed cloud K-SVD—for collaborative dictionary learning.

3.3.1 Dictionary Learning Using K-SVD

The K-SVD algorithm initializes with a (often randomized) dictionary $\mathbf{D}^{(0)}$ and solves (3.1) by iterating between two stages: a *sparse coding stage* and a *dictionary update stage* [4]. Specifically, for a fixed estimate of the dictionary $\mathbf{D}^{(t-1)}$ at the start of iteration $t \geq 1$, the sparse coding stage in K-SVD involves solving (3.1) for $\mathbf{X}^{(t)}$ as follows:

$$\forall s, \ \mathbf{x}_s^{(t)} = \arg\min_{\mathbf{x} \in \mathbb{R}^K} \|\mathbf{y}_s - \mathbf{D}^{(t-1)}\mathbf{x}\|_2^2 \text{ s.t. } \|\mathbf{x}\|_0 \leq T_0, \tag{3.2}$$

where $\mathbf{y}_s$ denotes the s^{th} column of $\mathbf{Y}$. While (3.2) in its stated form has combinatorial complexity, it can be solved approximately by either convexifying (3.2) [105] or using greedy algorithms [106].

After the sparse coding stage, K-SVD fixes $\mathbf{X}^{(t)}$ and moves to the dictionary update stage. The main novelty in K-SVD lies in the manner in which it carries out dictionary update, which involves iterating through the K atoms of $\mathbf{D}^{(t-1)}$ and individually updating the k^{th} atom,

Algorithm 3: Cloud K-SVD for dictionary learning

```
Input: Local data $\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_N$, problem parameters $K$ and $T_0$, and
  doubly-stochastic matrix $\mathbf{W}$.
Initialize: Generate $\mathbf{d}^{\text{ref}} \in \mathbb{R}^n$ and $\mathbf{D}^{\text{init}} \in \mathbb{R}^{n \times K}$ randomly, set $t \leftarrow 0$ and
  $\widehat{\mathbf{D}}_i^{(t)} \leftarrow \mathbf{D}^{\text{init}}, i = 1, \ldots, N$.
1:  while stopping rule do
2:    $t \leftarrow t + 1$
3:    (Sparse Coding) The $i^{th}$ site solves $\forall s, \widetilde{\mathbf{x}}_{i,s}^{(t)} \leftarrow \arg\min_{\mathbf{x} \in \mathbb{R}^K} \|\mathbf{y}_{i,s} - \widehat{\mathbf{D}}_i^{(t-1)}\mathbf{x}\|_2^2$ s.t. $\|\mathbf{x}\|_0 \leq T_0$
4:    for $k = 1$ to $K$ (Dictionary Update) do
5:      $\widehat{\mathbf{E}}_{i,k,R}^{(t)} \leftarrow \mathbf{Y}_i \widetilde{\Omega}_{i,k}^{(t)} - \sum_{j=1}^{k-1} \widehat{\mathbf{d}}_{i,j}^{(t)} \widehat{\mathbf{x}}_{i,j,T}^{(t)} \widetilde{\Omega}_{i,k}^{(t)}$
                 $- \sum_{j=k+1}^{K} \widehat{\mathbf{d}}_{i,j}^{(t-1)} \widetilde{\mathbf{x}}_{i,j,T}^{(t)} \widetilde{\Omega}_{i,k}^{(t)}$
6:      $\widehat{\mathbf{M}}_i \leftarrow \widehat{\mathbf{E}}_{i,k,R}^{(t)} \widehat{\mathbf{E}}_{i,k,R}^{(t)\text{T}}$
7:      (Initialize Distributed Power Method) Generate $\mathbf{q}^{\text{init}}$ randomly, set $t_p \leftarrow 0$ and
        $\widehat{\mathbf{q}}_i^{(t_p)} \leftarrow \mathbf{q}^{\text{init}}$
8:      while stopping rule do
9:        $t_p \leftarrow t_p + 1$
10:       (Initialize Consensus Averaging) Set $t_c \leftarrow 0$ and $\mathbf{z}_i^{(t_c)} \leftarrow \widehat{\mathbf{M}}_i \widehat{\mathbf{q}}_i^{(t_p-1)}$
11:       while stopping rule do
12:         $t_c \leftarrow t_c + 1$
13:         $\mathbf{z}_i^{(t_c)} \leftarrow \sum_{j \in \mathcal{N}_i} \mathbf{w}_{i,j} \mathbf{z}_i^{(t_c-1)}$
14:       end while
15:       $\widehat{\mathbf{v}}_i^{(t_p)} \leftarrow \mathbf{z}_i^{(t_c)} / [\mathbf{W}_1^{t_c}]_i$
16:       $\widehat{\mathbf{q}}_i^{(t_p)} \leftarrow \widehat{\mathbf{v}}_i^{(t_p)} / \|\widehat{\mathbf{v}}_i^{(t_p)}\|_2$
17:     end while
18:     $\widehat{\mathbf{d}}_{i,k}^{(t)} \leftarrow \text{sgn}\left(\langle \mathbf{d}^{\text{ref}}, \widehat{\mathbf{q}}_i^{(t_p)} \rangle\right) \widehat{\mathbf{q}}_i^{(t_p)}$
19:     $\widehat{\mathbf{x}}_{i,k,R}^{(t)} \leftarrow \widehat{\mathbf{d}}_{i,k}^{(t)\text{T}} \widehat{\mathbf{E}}_{i,k,R}^{(t)}$
20:   end for
21: end while
Return: $\widehat{\mathbf{D}}_i^{(t)}, i = 1, 2, \ldots, N$.
```

$k \in 1, \ldots, K$, as follows:

$$\begin{aligned}\mathbf{d}_k^{(t)} &:= \arg\min_{\mathbf{d} \in \mathbb{R}^n} \left\| \Big(\mathbf{Y} - \sum_{j=1}^{k-1} \mathbf{d}_j^{(t)} \mathbf{x}_{j,T}^{(t)} - \sum_{j=k+1}^{K} \mathbf{d}_j^{(t-1)} \mathbf{x}_{j,T}^{(t)}\Big) - \mathbf{d}\mathbf{x}_{k,T}^{(t)} \right\|_F^2 \\ &= \arg\min_{\mathbf{d} \in \mathbb{R}^n} \|\mathbf{E}_k^{(t)} - \mathbf{d}\,\mathbf{x}_{k,T}^{(t)}\|_F^2. \end{aligned} \tag{3.3}$$

Here, $\mathbf{E}_k^{(t)}$ is the representation error for $\mathbf{Y}$ using first $k-1$ atoms of $\mathbf{D}^{(t)}$ and last $k+1, \ldots, K$ atoms of $\mathbf{D}^{(t-1)}$. In order to simplify computations, K-SVD in [4] further defines an ordered set $\omega_k^{(t)} = \{s : 1 \leq s \leq S, \mathbf{x}_{k,T}^{(t)}(s) \neq 0\}$, where $\mathbf{x}_{k,T}^{(t)}(s)$ denotes the s^{th} element of $\mathbf{x}_{k,T}^{(t)}$, and an $S \times |\omega_k^{(t)}|$ binary matrix $\Omega_k^{(t)}$ that has ones in $(\omega_k^{(t)}(s), s)$ locations and zeros everywhere else. Then, defining $\mathbf{E}_{k,R}^{(t)} = \mathbf{E}_k^{(t)} \Omega_k^{(t)}$ and $\mathbf{x}_{k,R}^{(t)} = \mathbf{x}_{k,T}^{(t)} \Omega_k^{(t)}$, it is easy to see from (3.3) that $\mathbf{d}_k^{(t)} = \arg\min_{\mathbf{d} \in \mathbb{R}^n} \left\| \mathbf{E}_{k,R}^{(t)} - \mathbf{d}\,\mathbf{x}_{k,R}^{(t)} \right\|_F^2$. Therefore, solving (3.3) is equivalent to finding the best rank-one approximation of $\mathbf{E}_{k,R}^{(t)}$, which is given by the Eckart–Young theorem as $\mathbf{d}_k^{(t)} \mathbf{x}_{k,R}^{(t)} = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^{\text{T}}$, where $\mathbf{u}_1$ and $\mathbf{v}_1$ denote the largest left- and right-singular vectors of $\mathbf{E}_{k,R}^{(t)}$, respectively, while σ_1 denotes the largest singular value of $\mathbf{E}_{k,R}^{(t)}$. The k^{th} atom of $\mathbf{D}^{(t)}$ can now simply be updated as $\mathbf{d}_k^{(t)} = \mathbf{u}_1$. It is further advocated in [4] that the k^{th} row of the "reduced" coefficient matrix, $\mathbf{x}_{k,R}^{(t)}$, should be simultaneously updated to $\mathbf{x}_{k,R}^{(t)} = \sigma_1 \mathbf{v}_1^{\mathrm{T}}$. The dictionary update stage in K-SVD involves K such applications of the Eckart–Young theorem to update the K atoms of $\mathbf{D}^{(t-1)}$ and the K "reduced" rows of $\mathbf{X}^{(t)}$. The algorithm then moves to the sparse coding stage and continues alternating between the two stages till a stopping criterion (e.g., a prescribed representation error) is reached.

3.3.2 Collaborative Dictionary Learning Using Cloud K-SVD

We now present our decentralized dictionary learning algorithm based on K-SVD. The key to distributing K-SVD is understanding ways in which both the sparse coding and the dictionary update stages can be distributed. To this end, at iteration $t \geq 1$ of decentralized dictionary learning algorithm each site i has a local estimate $\widehat{\mathbf{D}}_i^{(t-1)}$ of the desired dictionary from the previous iteration. In order for the sparse coding stage to proceed, a key observation here is that as long as the local dictionary estimates $\widehat{\mathbf{D}}_i^{(t-1)}$ remain close to each other (established in Section 3.4) each site can "locally" compute representation coefficients of its local data without collaborating with other sites by locally solving Step 3 of Algorithm 3, i.e.,

$$\forall s,\ \widetilde{\mathbf{x}}_{i,s}^{(t)} = \arg\min_{\mathbf{x}\in\mathbb{R}^K} \|\mathbf{y}_{i,s} - \widehat{\mathbf{D}}_i^{(t-1)}\mathbf{x}\|_2^2 \text{ s.t. } \|\mathbf{x}\|_0 \leq T_0, \tag{3.4}$$

where $\mathbf{y}_{i,s}$ and $\widetilde{\mathbf{x}}_{i,s}^{(t)}$ denote the s^{th} sample and its coefficient vector at site i, respectively.

The next challenge in collaborative dictionary learning based on K-SVD arises during the dictionary update stage. Recall that the dictionary update stage in K-SVD involves computing the largest left- and right-singular vectors of the "reduced" error matrix $\mathbf{E}_{k,R}^{(t)} = \mathbf{E}_k^{(t)}\Omega_k^{(t)}, k \in \{1,\dots,K\}$. However, unless the local dictionary estimates $\widehat{\mathbf{D}}_i^{(t-1)}$ happen to be identical, we end up with N such (reduced) error matrices in a distributed setting due to N different local dictionary estimates. To resolve this, we propose to use the following definition of the reduced error matrix in a distributed setting: $\widehat{\mathbf{E}}_{k,R}^{(t)} = \begin{bmatrix}\widehat{\mathbf{E}}_{1,k,R}^{(t)} & \dots & \widehat{\mathbf{E}}_{N,k,R}^{(t)}\end{bmatrix}$, where $\widehat{\mathbf{E}}_{i,k,R}^{(t)} = \mathbf{Y}_i\widetilde{\Omega}_{i,k}^{(t)} - \left(\sum_{j=1}^{k-1}\widehat{\mathbf{d}}_{i,j}^{(t)}\widehat{\mathbf{x}}_{i,j,T}^{(t)} + \sum_{j=k+1}^{K}\widehat{\mathbf{d}}_{i,j}^{(t-1)}\widetilde{\mathbf{x}}_{i,j,T}^{(t)}\right)\widetilde{\Omega}_{i,k}^{(t)}$. Here, $\widetilde{\mathbf{x}}_{i,j,T}^{(t)}$ denotes the j^{th} row of coefficient matrix $\widetilde{\mathbf{X}}_i^{(t)}$ computed at site i during the sparse coding step performed on $\mathbf{Y}_i$ using $\widehat{\mathbf{D}}_i^{(t-1)}$ at the start of iteration t, while $\widehat{\mathbf{x}}_{i,j,T}^{(t)}$ denotes the j^{th} row of the updated coefficient matrix $\widehat{\mathbf{X}}_i^{(t)}$ available at site i due to the update in coefficient matrix performed during the dictionary update step. Furthermore, $\widetilde{\Omega}_{i,k}^{(t)}$ is similar to $\Omega_k^{(t)}$ defined for K-SVD except that it is now defined for only local coefficient matrix $\widetilde{\mathbf{X}}_i^{(t)}$ at site i.

Next, in keeping with the K-SVD derivation in [4], we propose that each of the N sites updates the k^{th} atom of its respective local dictionary and the k^{th} row of its respective "reduced" coefficient matrix, $\widehat{\mathbf{x}}_{i,k,R}^{(t)} = \widehat{\mathbf{x}}_{i,k,T}^{(t)}\widetilde{\Omega}_{i,k}^{(t)}$, by collaboratively computing the dominant left- and right-singular vectors of the *distributed error matrix* $\widehat{\mathbf{E}}_{k,R}^{(t)}$, denoted by $\mathbf{u}_1$ and $\mathbf{v}_1$, respectively.[1] In fact, since $\mathbf{u}_1^{\mathrm{T}}\widehat{\mathbf{E}}_{k,R}^{(t)} = \sigma_1\mathbf{v}_1$ with σ_1 being the largest singular value of $\widehat{\mathbf{E}}_{k,R}^{(t)}$, it follows that if a site has access to the dominant left-singular vector, $\mathbf{u}_1$, of $\widehat{\mathbf{E}}_{k,R}^{(t)}$ then it can simply update the k^{th} row of its respective "reduced" coefficient matrix by setting $\widehat{\mathbf{d}}_{i,k}^{(t)} = \mathbf{u}_1$ and setting $\widehat{\mathbf{x}}_{i,k,R}^{(t)} = \widehat{\mathbf{d}}_{i,k}^{(t)^{\mathrm{T}}}\widehat{\mathbf{E}}_{i,k,R}^{(t)}$. Therefore, we need only worry about collaborative computation of $\mathbf{u}_1$ in this setting. To this end, we define $\widehat{\mathbf{M}}^{(t)} = \widehat{\mathbf{E}}_{k,R}^{(t)}\widehat{\mathbf{E}}_{k,R}^{(t)^{\mathrm{T}}}$ and note that $\mathbf{u}_1$ corresponds to the dominant eigenvector of $\widehat{\mathbf{M}}^{(t)}$. Now express $\widehat{\mathbf{M}}^{(t)}$ as $\widehat{\mathbf{M}}^{(t)} = \sum_{i=1}^{N}\widehat{\mathbf{M}}_i^{(t)}$ and notice that each $\widehat{\mathbf{M}}_i^{(t)} = \widehat{\mathbf{E}}_{i,k,R}^{(t)}\widehat{\mathbf{E}}_{i,k,R}^{(t)^{\mathrm{T}}}$ is a matrix that is readily computable at each local site. Our goal now is computing the dominant eigenvector of $\widehat{\mathbf{M}}^{(t)} = \sum_{i=1}^{N}\widehat{\mathbf{M}}_i^{(t)}$ in a collaborative manner at each site. In order for this, we will make use of decentralized power method, which has been invoked previously in [76, 77, 103] and which corresponds to a decentralized variant of the classical power method for eigenanalysis [31].

Decentralized Power Method: Power method is an iterative procedure for computing eigenvectors of a matrix. It is simple to implement and assuming that the largest eigenvalue λ_1 of a matrix is strictly greater than its second-largest eigenvalue λ_2 it converges to the subspace spanned by the dominant eigenvector at an exponential rate. We are interested in a decentralized variant of the power method to compute the dominant eigenvector of $\widehat{\mathbf{M}}^{(t)} = \sum_{i=1}^{N}\widehat{\mathbf{M}}_i^{(t)}$, where the $\widehat{\mathbf{M}}_i^{(t)}$'s are distributed across N sites. To this end, we proceed as follows.

First, all sites initialize to the same (unit-norm) estimate of the eigenvector $\widehat{\mathbf{q}}_i^{(0)} = \mathbf{q}^{\text{init}}$.[2] At iteration t_p of the decentralized power method, each site computes $\widehat{\mathbf{M}}_i^{(t)}\widehat{\mathbf{q}}_i^{(t_p-1)}$ locally, where $\widehat{\mathbf{q}}_i^{(t_p-1)}$ is the estimate of the dominant eigenvector of $\widehat{\mathbf{M}}^{(t)}$ at the i^{th} site after (t_p-1)-th iteration of the decentralized power method. In the next step, the sites collaboratively compute an approximation $\widehat{\mathbf{v}}_i^{(t_p)}$ of $\sum_i \widehat{\mathbf{M}}_i^{(t)}\widehat{\mathbf{q}}_i^{(t_p-1)}$ in a decentralized manner (Steps 10–14 of Algorithm 3). Finally, each site normalizes its estimate of the dominant eigenvector of $\widehat{\mathbf{M}}^{(t)}$ locally: $\widehat{\mathbf{q}}_i^{(t_p)} = \widehat{\mathbf{v}}_i^{(t_p)}/\|\widehat{\mathbf{v}}_i^{(t_p)}\|_2$.

It is clear from the preceding discussion that the key in decentralized power method is the ability of the sites to collaboratively compute an approximation of $\sum_i \widehat{\mathbf{M}}_i^{(t)}\widehat{\mathbf{q}}_i^{(t_p-1)}$ in each

[1] An alternative is to compute an estimate of $\widehat{\mathbf{E}}_{k,R}^{(t)}$ at each site using consensus averaging, after which individual sites can compute SVD of $\widehat{\mathbf{E}}_{k,R}^{(t)}$ locally. Despite its apparent simplicity, this approach will have significantly greater communication overhead compared to our proposed method.

[2] This can be accomplished, for example, through the use of (local) random number generators initialized with the same seed. Also, note that a key requirement in power method is that $\langle \mathbf{u}_1, \mathbf{q}^{init}\rangle \neq 0$, which holds with probability one in the case of a random initialization.

iteration. In order for this, we make use of the popular consensus averaging method [107]. To perform consensus averaging, we first design a doubly-stochastic weight matrix $\mathbf{W}$ that adheres to the topology of the underlying graph $\mathcal{G}$. In particular, we have that $\mathbf{w}_{i,j} = 0$ whenever $(i,j) \notin \mathcal{E}$. We refer the reader to [107–109] for designing appropriate weight matrices in a decentralized manner without relying on the knowledge of the global network topology. In order to compute $\sum_i \widehat{\mathbf{M}}_i^{(t)} \widehat{\mathbf{q}}_i^{(t_p-1)}$ using consensus averaging each site initializes consensus averaging routine (Steps 10–14 of Algorithm 3) with their local estimates of the top eigenvectors at iteration t_p of the power method, $\mathbf{z}_i^{(0)} = \widehat{\mathbf{M}}_i^{(t)} \widehat{\mathbf{q}}_i^{(t_p-1)}$. Next, we define $\mathcal{N}_i = \{j : (i,j) \in \mathcal{E}\}$ to be the neighborhood of site i and $\mathbf{Z}^{(0)} = \begin{bmatrix}\mathbf{z}_1^{(0)} & \dots & \mathbf{z}_N^{(0)}\end{bmatrix}^{\mathrm{T}}$ be the concatenation of all the local initial eigenvector estimates. Then consensus averaging at iteration, t_c, works by having each site carry out the following update through communications with its neighbors: $\mathbf{z}_i^{(t_c)} = \sum_{j \in \mathcal{N}_i} \mathbf{w}_{i,j} \mathbf{z}_j^{(t_c-1)}$. The dynamics of the overall system in this case evolve as $\mathbf{Z}^{(t_c)} = \mathbf{W}^{t_c} \mathbf{Z}^{(0)}$. It then follows that $\mathbf{Z}_{i,T}^{(t_c)} \xrightarrow{t_c} \mathbf{1}^{\mathrm{T}} \mathbf{Z}^{(0)}/N$ [107], where $\mathbf{Z}_{i,T}^{(t_c)}$ denotes the i^{th} row of $\mathbf{Z}^{(t_c)}$ and $\mathbf{1} \in \mathbb{R}^N$ denotes a (column) vector of all ones. This in particular implies that each site achieves perfect consensus averaging as $t_c \to \infty$ and obtains ${\mathbf{Z}_{i,T}^{(\infty)}}^{\mathrm{T}} = \frac{1}{N} \sum_{j=1}^N \mathbf{z}_j^{(0)} = \frac{1}{N} \sum_{j=1}^N \widehat{\mathbf{M}}_j^{(t)} \widehat{\mathbf{q}}_j^{(t_p-1)}$.

But one cannot perform infinite consensus iterations in practice within each iteration of the decentralized power method. Instead, we assume a finite number of consensus iterations, denoted by T_c, in each power method iteration and make use of the modification of standard consensus averaging proposed in [103] to obtain $\widehat{\mathbf{v}}_i^{(t_p)} = {\mathbf{Z}_{i,T}^{(T_c)}}^{\mathrm{T}} / [\mathbf{W}_1^{T_c}]_i$, where $\mathbf{W}_1^{T_c}$ is the first column of $\mathbf{W}^{T_c}$ and $[\cdot]_i$ denotes the i^{th} entry of a vector. Note that this leads to an error $\epsilon_{i,c}^{(t_p)}$ within $\widehat{\mathbf{v}}_i^{(t_p)}$ at each site for any finite T_c, i.e., $\widehat{\mathbf{v}}_i^{(t_p)} = {\mathbf{Z}_{i,T}^{(T_c)}}^{\mathrm{T}} / [\mathbf{W}_1^{T_c}]_i = \sum_{j=1}^N \widehat{\mathbf{M}}_j \widehat{\mathbf{q}}_j^{(t_p-1)} + \epsilon_{i,c}^{(t_p)}$. After the completion of consensus iterations, each site i in iteration t_p of power method normalizes this vector $\widehat{\mathbf{v}}_i^{(t_p)}$ to get an estimate of the dominant eigenvector of $\widehat{\mathbf{M}}^{(t)}$. Finally, we carry out enough iterations of the distributed power method at each site that the error between successive estimates of the eigenvector falls below a prescribed threshold.

We have now motivated and described the key components of our proposed algorithm and the full collaborative dictionary learning algorithm, termed *cloud K-SVD*, is detailed in Algorithm 3. Notice the initialization of cloud K-SVD differs from K-SVD in the sense that each site also generates a common (random) reference vector $\mathbf{d}^{\mathrm{ref}} \in \mathbb{R}^n$ and stores it locally. The purpose of $\mathbf{d}^{\mathrm{ref}}$ is to ensure that the eigenvectors computed by different sites using the distributed power method all point in the same quadrant, rather than in antipodal quadrants (Step 18 in Algorithm 3). While this plays a role in analysis, it does not have an effect on the workings of cloud K-SVD. Notice also that we have not defined any stopping rules in Algorithm 3. One set of rules could be to run the algorithm for fixed dictionary learning iterations T_d, power method iterations T_p, and consensus iterations T_c. It is worth noting here that algorithms such as cloud K-SVD are often referred to as two time-scale algorithms in the literature. Nonetheless, cloud K-SVD with the stopping rules of finite (T_d, T_p, T_c) can be considered a *quasi* one time-scale algorithm. Accordingly, our analysis of cloud K-SVD assumes these stopping rules.

Remark 4. A careful reading of Algorithm 3 reveals that normalization by $[\mathbf{W}_1^{t_c}]_i$ in Step 15 is redundant due to the normalization in Step 16. We retain the current form of Step 15 however to facilitate the forthcoming analysis.

3.4 Analysis of Cloud K-SVD

Since power method and consensus averaging in Algorithm 3 cannot be performed for an infinite number of iterations, in practice this results in residual errors in each iteration of the algorithm. It is therefore important to understand whether the dictionaries $\{\widehat{\mathbf{D}}_i\}$ returned by cloud K-SVD approach the dictionary that could have been obtained by centralized K-SVD [4]. In order to address this question, we need to understand the behavior of major components of cloud K-SVD, which include sparse coding, dictionary update, and distributed power method within dictionary update. In addition, one also expects that the closeness of $\widehat{\mathbf{D}}_i$'s to the centralized solution will be a function of certain properties of local/global data. We begin our analysis of cloud K-SVD by first stating some of these properties in terms of the centralized K-SVD solution.

3.4.1 Preliminaries

The first thing needed to quantify deviations of the cloud K-SVD dictionaries from the centralized K-SVD dictionary is algorithmic specification of the sparse coding steps in both algorithms. While the sparse coding steps as stated in (3.2) and (3.4) have combinatorial complexity, various low-complexity computational approaches can be used to solve these steps in practice. Our analysis in the following will be focused on the case when sparse coding in both cloud K-SVD and centralized K-SVD is carried out using the *lasso* [110]. Specifically, we assume sparse coding is carried out by solving

$$\mathbf{x}_{i,s} = \arg \min_{\mathbf{x} \in \mathbb{R}^K} \tfrac{1}{2}\|\mathbf{y}_{i,s} - \mathbf{D}\mathbf{x}\|_2^2 + \tau\|\mathbf{x}\|_1 \tag{3.5}$$

with the regularization parameter $\tau > 0$ selected in a way that $\|\mathbf{x}_{i,s}\|_0 \leq T_0 \ll n$. This can be accomplished, for example, by making use of the *least angle regression* algorithm [111]. Note that the lasso also has an alternate constrained form which is given by

$$\mathbf{x}_{i,s} = \arg\min_{\mathbf{x}\in\mathbb{R}^K} \tfrac{1}{2}\|\mathbf{y}_{i,s} - \mathbf{D}\mathbf{x}\|_2^2 \quad \text{s.t.} \quad \|\mathbf{x}\|_1 \le \eta, \tag{3.6}$$

and (3.5) & (3.6) are identical for an appropriate $\eta_\tau = \eta(\tau)$ [112].

Remark 5. While extension of our analysis to other sparse coding methods such as *orthogonal matching pursuit* (OMP) [106] is beyond the scope of this work, such extensions would mainly rely on perturbation analyses of different sparse coding methods. In the case of OMP, for instance, such perturbation analysis is given in [113], which can then be leveraged to extend our lasso-based cloud K-SVD result to OMP-based result.

Our analysis in the following is also based on the assumption that cloud K-SVD and centralized K-SVD are identically initialized, i.e., $\widehat{\mathbf{D}}_i^{(0)} = \mathbf{D}^{(0)}, i = 1, \dots, N$, where $\mathbf{D}^{(t)}, t \ge 0$, in the following denotes the centralized K-SVD dictionary estimate in the t^{th} iteration. While both cloud K-SVD and centralized K-SVD start from the same initial estimates, the cloud K-SVD dictionaries get perturbed in each iteration due to imperfect power method and consensus averaging. In order to ensure these perturbations do not cause the cloud K-SVD dictionaries to diverge from the centralized solution after T_d iterations, we need the dictionary estimates returned by centralized K-SVD in each iteration to satisfy certain properties. Below, we present and motivate these properties.

[P1] Let $\mathbf{x}_{i,s}^{(t)}$ denote the solution of the lasso (i.e., (3.5)) for $\mathbf{D} = \mathbf{D}^{(t-1)}$ and regularization parameter $\tau = \tau^{(t)}, t = 1, \dots, T_d$. Then there exists some $C_1 > 0$ such that the following holds:

$$\min_{t,i,s,j\notin \text{supp}(\mathbf{x}_{i,s}^{(t)})} \tau^{(t)} - \left|\langle \mathbf{d}_j^{(t)}, \mathbf{y}_{i,s} - \mathbf{D}^{(t-1)}\mathbf{x}_{i,s}^{(t)}\rangle\right| > C_1.$$

In our analysis in the following, we will also make use of the smallest regularization parameter among the collection $\{\tau^{(t)}\}_{t=1}^{T_d}$, defined as $\tau_{\min} = \min_t \tau^{(t)}$, and the largest value of parameter η_τ (defined in (3.6)) among the collection $\{\eta_\tau^{(t)} = \eta(\tau^{(t)})\}_{t=1}^{T_d}$, defined as $\eta_{\tau,\max} = \max_t \eta_\tau^{(t)}$.

[P2] Define $\Sigma_{T_0} = \{\mathcal{I} \subset \{1, \dots, K\} : |\mathcal{I}| = T_0\}$. Then there exists some $C_2' > \frac{C_1^4 \tau_{\min}^2}{1936}$ such that the following holds:

$$\min_{t=1,\dots,T_d, \mathcal{I}\in\Sigma_{T_0}} \sigma_{T_0}\left(\mathbf{D}_{|\mathcal{I}}^{(t-1)}\right) \ge \sqrt{C_2'},$$

where $\sigma_{T_0}(\cdot)$ denotes the T_0^{th} (ordered) singular value of a matrix. In our analysis, we will be using the parameter $C_2 = \left(\sqrt{C_2'} - \frac{C_1^2\tau_{\min}}{44}\right)^2$. As explained in the notation section, here, $\mathbf{D}_{|\mathcal{I}}^{(t-1)}$ is the submatrix of $\mathbf{D}^{(t-1)}$ containing only the columns at indices $\mathcal{I}$.

[P3] Let $\lambda_{1,k}^{(t)} > \lambda_{2,k}^{(t)} \geq \dots \lambda_{n,k}^{(t)} \geq 0$ denote the eigenvalues of the centralized "reduced" matrix $\mathbf{E}_{k,R}^{(t)} \mathbf{E}_{k,R}^{(t)^{\mathrm{T}}}, k \in \{1, \dots, K\}$, in the t^{th} iteration, $t \in \{1, \dots, T_d\}$. Then there exists some $C_3' < 1$ such that the following holds:

$$\max_{t,k} \frac{\lambda_{2,k}^{(t)}}{\lambda_{1,k}^{(t)}} \leq C_3'.$$

Now define $C_3 = \max\left\{1, \frac{1}{\min_{t,k} \lambda_{1,k}^{(t)}(1 - C_3')}\right\}$, which we will use in our forthcoming analysis.

We now comment on the rationale behind these three properties. Properties **[P1]** and **[P2]** correspond to sufficient conditions for $\mathbf{x}_{i,s}^{(t)}$ to be a unique solution of (3.5) [114] and guarantee that the centralized K-SVD generates a unique collection of sparse codes in each dictionary learning iteration. Property **[P3]**, on the other hand, ensures that algorithms such as the power method can be used to compute the dominant eigenvector of $\mathbf{E}_{k,R}^{(t)} \mathbf{E}_{k,R}^{(t)^{\mathrm{T}}}$ in each dictionary learning iteration [31]. In particular, **[P3]** is a statement about the worst-case spectral gap of $\mathbf{E}_{k,R}^{(t)} \mathbf{E}_{k,R}^{(t)^{\mathrm{T}}}$. In addition to these properties, our final analytical result for cloud K-SVD will also be a function of a certain parameter of the centralized error matrices $\{\mathbf{E}_k^{(t)}\}_{k=1}^{K}$ generated by the centralized K-SVD in each iteration. We define this parameter in the following for later use. Let $\mathbf{E}_{i,k}^{(t)}, i = 1, \dots, N$, denote part of the centralized error matrix $\mathbf{E}_k^{(t)}$ associated with the data of the i^{th} site in the t^{th} iteration, i.e., $\mathbf{E}_k^{(t)} = \begin{bmatrix} \mathbf{E}_{1,k}^{(t)} & \cdots & \mathbf{E}_{N,k}^{(t)} \end{bmatrix}, k = 1, \dots, K, t = 1, \dots, T_d$. Then

$$C_4 = \max\left\{1, \max_{t,i,k} \|\mathbf{E}_{i,k}^{(t)}\|_2\right\}. \tag{3.7}$$

3.4.2 Main Result

We are now ready to state the main result of this paper. This result is given in terms of the $\|\cdot\|_2$ norm mixing time, T_{mix}, of the Markov chain associated with the doubly-stochastic weight matrix $\mathbf{W}$, defined as

$$T_{mix} = \max_{i=1,\dots,N} \inf_{t \in \mathbb{N}} \left\{ t : \|\mathrm{e}_i^{\mathrm{T}} \mathbf{W}^t - \tfrac{1}{N} \mathbf{1}^{\mathrm{T}}\|_2 \leq \frac{1}{2} \right\}. \tag{3.8}$$

Here, $\mathbf{e}_i \in \mathbb{R}^N$ denotes the i^{th} column of the identity matrix $\mathbf{I}_N$. Note that the mixing time T_{mix} can be upper bounded in terms of inverse of the absolute spectral gap of $\mathbf{W}$, defined as $1 - |\lambda_2(\mathbf{W})|$ with $\lambda_2(\mathbf{W})$ denoting the second largest (in modulus) eigenvalue of $\mathbf{W}$ [115]. As a general rule, better-connected networks can be made to have smaller mixing times compared to sparsely connected networks. We refer the reader to [116] and [115, Chap. 15] for further details on the relationship between T_{mix} and the underlying network topology.

Theorem 5 (Stability of Cloud K-SVD Dictionaries)**.** *Suppose cloud K-SVD (Algorithm 3) and (centralized) K-SVD are identically initialized and both of them carry out T_d dictionary learning iterations. In addition, assume cloud K-SVD carries out T_p power method iterations during the update of each atom and T_c consensus iterations during each power method iteration. Finally, assume the K-SVD algorithm satisfies properties* ***[P1]–[P3]****. Next, define* $\alpha = \max_{t,k} \sum_{i=1}^N \|\widehat{\mathbf{E}}_{i,k,R}^{(t)} \widehat{\mathbf{E}}_{i,k,R}^{(t)^\mathrm{T}}\|_2$, $\beta = \max_{t,t_p,k} \frac{1}{\left\|\widehat{\mathbf{E}}_{k,R}^{(t)} \widehat{\mathbf{E}}_{k,R}^{(t)^\mathrm{T}} \mathbf{q}_{\mathbf{c},t,k}^{(t_p)}\right\|_2}$, $\gamma = \max_{t,k} \sqrt{\sum_{i=1}^N \|\widehat{\mathbf{E}}_{i,k,R}^{(t)} \widehat{\mathbf{E}}_{i,k,R}^{(t)^\mathrm{T}}\|_F^2}$, $\nu = \max_{t,k} \frac{\widehat{\lambda}_{2,k}^{(t)}}{\widehat{\lambda}_{1,k}^{(t)}}$, $\widehat{\theta}_k^{(t)} \in [0, \pi/2]$ *as* $\widehat{\theta}_k^{(t)} = \arccos\left(\frac{\left|\left\langle \mathbf{u}_{1,k}^{(t)}, \mathbf{q}^{\mathrm{init}} \right\rangle\right|}{\|\mathbf{u}_{1,k}^{(t)}\|_2 \|\mathbf{q}^{\mathrm{init}}\|_2}\right)$, $\mu = \max\left\{1, \max_{k,t} \tan(\widehat{\theta}_k^{(t)})\right\}$, *and* $\zeta = K\sqrt{2S_{\max}}\left(\frac{6\sqrt{KT_0}}{\tau_{\min} C_2} + \eta_{\tau,\max}\right)$, *where* $S_{\max} = \max_i S_i$, $\mathbf{u}_{1,k}^{(t)}$ *is the dominant eigenvector of* $\widehat{\mathbf{E}}_{k,R}^{(t)} \widehat{\mathbf{E}}_{k,R}^{(t)^\mathrm{T}}$, $\widehat{\lambda}_{1,k}^{(t)}$ *and* $\widehat{\lambda}_{2,k}^{(t)}$ *are first and second largest eigenvalues of* $\widehat{\mathbf{E}}_{k,R}^{(t)} \widehat{\mathbf{E}}_{k,R}^{(t)^\mathrm{T}}$ *respectively, and* $\mathbf{q}_{\mathbf{c},t,k}^{(t_p)}$ *denotes the iterates of a centralized power method initialized with* $\mathbf{q}^{\mathrm{init}}$ *for estimation of the dominant eigenvector of* $\widehat{\mathbf{E}}_{k,R}^{(t)} \widehat{\mathbf{E}}_{k,R}^{(t)^\mathrm{T}}$*. Then, assuming* $\min_{t,k} \left|\langle \mathbf{u}_{1,k}^{(t)}, \mathbf{q}^{\mathrm{init}} \rangle\right| > 0$, *and fixing any* $\epsilon \in \left(0, \min\left\{(10\alpha^2\beta^2)^{-1/3T_p}, (\frac{1-\nu}{4})^{1/3}\right\}\right)$ *and* $\delta_d \in \left(0, \min\left\{\frac{1}{\sqrt{2}}, \frac{C_1^2 \tau_{\min}}{44\sqrt{2K}}\right\}\right)$, *we have*

$$\max_{\substack{i=1,\dots,N \\ k=1,\dots,K}} \left\|\widehat{\mathbf{d}}_{i,k}^{(T_d)} \widehat{\mathbf{d}}_{i,k}^{(T_d)^\mathrm{T}} - \mathbf{d}_k^{(T_d)} \mathbf{d}_k^{(T_d)^\mathrm{T}}\right\|_2 \leq \delta_d \tag{3.9}$$

as long as the number of power method iterations

$$T_p \geq \frac{2(T_d K - 2)\log(8C_3 C_4^2 N + 5) + (T_d - 1)\log(1 + \zeta) + \log(8C_3 C_4 \mu N \sqrt{n} \delta_d^{-1})}{\log[(\nu + 4\epsilon^3)^{-1}]}$$

and the number of consensus iterations $T_c = \Omega\big(T_p T_{mix} \log(2\alpha\beta\epsilon^{-1}) + T_{mix} \log(\alpha^{-1}\gamma\sqrt{N})\big)$.

The proof of this theorem is given in Section 3.8. We now comment on the major implications of Theorem 5. First, the theorem establishes that the decentralized dictionaries $\{\widehat{\mathbf{D}}_i^{(T_d)}\}$ can indeed remain arbitrarily close to the centralized dictionary $\mathbf{D}^{(T_d)}$ after T_d dictionary learning iterations (cf. 3.9). Second, the theorem shows that this can happen as long as the number of decentralized power method iterations T_p scale in a certain manner. In particular, Theorem 5 calls for this scaling to be at least linear in $T_d K$ (modulo the $\log N$ multiplication factor), which is the total number of SVDs that K-SVD needs to perform in T_d dictionary learning iterations. On the other hand, T_p need only scale logarithmically with $S_{\max}$, which is significant in the context of big data problems. Other main problem parameters that affect the scaling of T_p include T_0, n, and δ_d^{-1}, all of which enter the scaling relation in a logarithmic fashion. Finally, Theorem 5 dictates that the number of consensus iterations T_c should also scale at least linearly with $T_p T_{mix}$ (modulo some log factors) for the main result to hold. Notice that the effect of network topology on the number of consensus iterations is captured through the dependence of T_c on the mixing time T_{mix}. Combining all the iterative steps we end up performing $T_d \times T_p \times T_c$ communications throughout the dictionary learning process which in terms of number of samples scales as $\Omega(\log^2 S_{\max})$ which makes cloud K-SVD a good candidate for big data settings. In summary, Theorem 5 guarantees that the decentralized dictionaries learned by cloud K-SVD can remain close to the centralized dictionary without requiring excessive numbers of power method and consensus averaging iterations.

We now discuss briefly the main analytical challenges that one needs to address to prove Theorem 5. In the first dictionary learning iteration ($t = 1$), we have $\{\widehat{\mathbf{D}}_i^{(t-1)} \equiv \mathbf{D}^{(t-1)}\}$ due to identical initializations. While this means both K-SVD and cloud K-SVD result in identical sparse codes for $t = 1$, the distributed dictionaries begin to deviate from the centralized dictionary after this step. The perturbations in $\{\widehat{\mathbf{d}}_{i,k}^{(1)}\}$ happen due to the finite numbers of power method and consensus averaging iterations for $k = 1$, whereas they happen for $k > 1$ due to this reason as well as due to the earlier perturbations in $\{\widehat{\mathbf{d}}_{i,j}^{(1)}, \widehat{x}_{i,j,T}^{(1)}\}, j < k$. In subsequent dictionary learning iterations ($t > 1$), therefore, cloud K-SVD starts with already perturbed distributed dictionaries $\{\widehat{\mathbf{D}}_i^{(t-1)}\}$. This in turn also results in deviations of the sparse codes computed by K-SVD and cloud K-SVD, which then adds another source of perturbations in $\{\widehat{\mathbf{d}}_{i,k}^{(t)}\}$ during the dictionary update steps. To summarize, imperfect power method and consensus averaging in cloud K-SVD introduce errors in the top eigenvector estimates of (centralized) $\mathbf{E}_{1,R}^{(1)}\mathbf{E}_{1,R}^{(1)^{\mathrm{T}}}$ at individual sites, which then accumulate for $(k,t) \neq (1,1)$ to also cause errors in estimate $\widehat{\mathbf{E}}_{k,R}^{(t)}\widehat{\mathbf{E}}_{k,R}^{(t)^{\mathrm{T}}}$ of the matrix $\mathbf{E}_{k,R}^{(t)}\mathbf{E}_{k,R}^{(t)^{\mathrm{T}}}$ available to cloud K-SVD. Collectively, these two sources of errors cause deviations of the distributed dictionaries from the centralized dictionary and the proof of Theorem 5 mainly relies on our ability to control these two sources of errors. In the following we provide the roadmap towards addressing the above mentioned challenges and hence proving Theorem 5.

3.4.3 Roadmap to Theorem 5

The first main result needed for the proof of Theorem 5 looks at the errors in the estimates of the dominant eigenvector $\mathbf{u}_1$ of an arbitrary symmetric matrix $\mathbf{M} = \sum_{i=1}^{N} \mathbf{M}_i$ obtained at individual sites using imperfect power method and consensus averaging when the $\mathbf{M}_i$'s are distributed across the N sites (cf. Section 3.3.2). The following result effectively helps us control the errors in cloud K-SVD dictionaries due to Steps 7–17 in Algorithm 3.

Theorem 6 (Stability of Decentralized Power Method)**.** *Consider any symmetric matrix* $\mathbf{M} = \sum_{i=1}^{N} \mathbf{M}_i$ *with dominant eigenvector* $\mathbf{u}_1$ *and eigenvalues* $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$*. Suppose each* $\mathbf{M}_i, i = 1, \ldots, N$*, is only available at the* i^{th} *site in our network and let* $\widehat{\mathbf{q}}_i$ *denote an estimate of* $\mathbf{u}_1$ *obtained at site* i *after* T_p *iterations of the distributed power method (Steps*

7–17 in Algorithm 3). Next, define $\alpha_p = \sum_{i=1}^N \|\mathbf{M}_i\|_2$, $\beta_p = \max_{t_p=1,\ldots,T_p} \frac{1}{\|\mathbf{M}\mathbf{q}_{\mathbf{c}}^{(t_p)}\|_2}$, *and* $\gamma_p = \sqrt{\sum_{i=1}^N \|\mathbf{M}_i\|_F^2}$, *where* $\mathbf{q}_{\mathbf{c}}^{(t_p)}$ *denotes the iterates of a centralized power method initialized with* $\mathbf{q}^{\text{init}}$*. Then, fixing any* $\epsilon \in \big(0, (10\alpha_p^2\beta_p^2)^{-1/3T_p}\big)$*, we have*

$$\max_{i=1,\ldots,N} \left\|\mathbf{u}_1\mathbf{u}_1^{\mathrm{T}} - \widehat{\mathbf{q}}_i\widehat{\mathbf{q}}_i^{\mathrm{T}}\right\|_2 \leq \tan(\theta) \left|\frac{\lambda_2}{\lambda_1}\right|^{T_p} + 4\epsilon^{3T_p}, \tag{3.10}$$

as long as $|\langle \mathbf{u}_1, \mathbf{q}^{\text{init}}\rangle| > 0$ *and the number of consensus iterations within each iteration of the distributed power method (Steps 10–14 in Algorithm 3) satisfies* $T_c = \Omega\big(T_pT_{mix}\log(2\alpha_p\beta_p\epsilon^{-1}) + T_{mix}\log(\alpha_p^{-1}\gamma_p\sqrt{N})\big)$*. Here,* θ *denotes the angle between* $\mathbf{u}_1$ *and* $\mathbf{q}^{\text{init}}$*, defined as:*

$$\theta = \arccos(|\langle \mathbf{u}_1, \mathbf{q}^{\text{init}}\rangle| / (\|\mathbf{u}_1\|_2\|\mathbf{q}^{\text{init}}\|_2)).$$

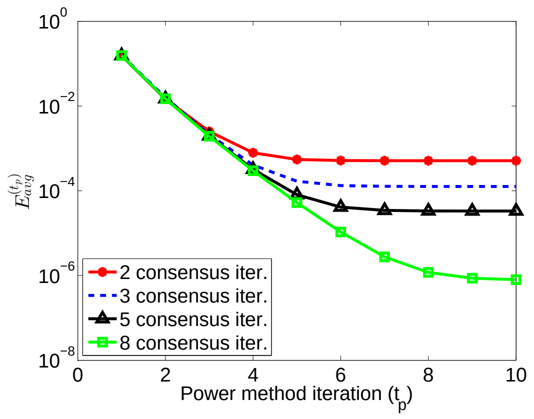
The proof of this theorem is given in Section 3.6. Theorem 6 states that $\widehat{\mathbf{q}}_i \xrightarrow{T_p} \pm\mathbf{u}_1$ geometrically at each site as long as enough consensus iterations are performed in each iteration of the distributed power method. In the case of a finite number of distributed power method iterations, (3.10) in Theorem 6 tells us that the maximum error in estimates of the dominant eigenvector is bounded by the sum of two terms, with the first term due to finite number of power method iterations and the second term due to finite number of consensus iterations.

The second main result needed to prove Theorem 5 looks at the errors between individual blocks of the reduced distributed error matrix $\widehat{\mathbf{E}}_{k,R}^{(t)} = \left[\widehat{\mathbf{E}}_{1,k,R}^{(t)}, \cdots, \widehat{\mathbf{E}}_{N,k,R}^{(t)}\right]$ and the reduced centralized error matrix $\mathbf{E}_{k,R}^{(t)} = \left[\mathbf{E}_{1,k,R}^{(t)}, \cdots, \mathbf{E}_{N,k,R}^{(t)}\right]$ for $k \in \{1, \cdots, K\}$ and $t \in \{1, \cdots, T_d\}$. This result helps us control the error in step 5 of Algorithm 3 and, together with Theorem 6, characterizes the major sources of errors in cloud K-SVD in relation to centralized K-SVD. The following theorem provides a bound on error in $\mathbf{E}_{i,k,R}^{(t)}$.

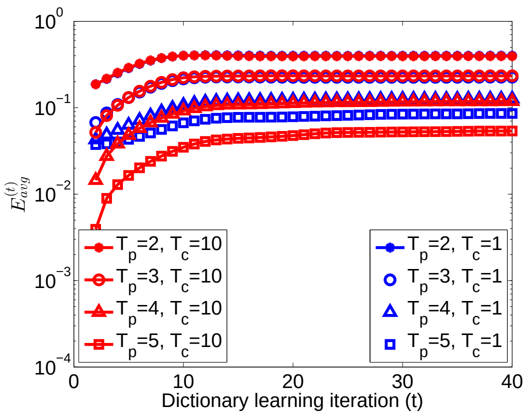
Theorem 7 (Perturbation in the matrix $\widehat{\mathbf{E}}_{i,k,R}^{(t)}$). *Recall the definitions of* $\Omega_k^{(t)}$ *and* $\widetilde{\Omega}_{i,k}^{(t)}$ *from Section 3.3.1 and Section 3.3.2, respectively. Next, express* $\Omega_k^{(t)} = \text{diag}\{\Omega_{1,k}^{(t)}, \cdots, \Omega_{N,k}^{(t)}\}$*, where* $\Omega_{i,k}^{(t)}$ *corresponds to the data samples associated with the* i^{th} *site, and define* $\mathbf{B}_{i,k,R}^{(t)} = \widehat{\mathbf{E}}_{i,k,R}^{(t)} - \mathbf{E}_{i,k,R}^{(t)}$*. Finally, let* ζ, μ, ν, ϵ, *and* δ_d *be as in Theorem 5, define* $\varepsilon = \mu\nu^{T_p} + 4\epsilon^{3T_p}$*, and assume* $\varepsilon \leq \frac{\delta_d}{8N\sqrt{n}C_3(1+\zeta)^{T_d-1}C_4^2(8C_3NC_4^2+5)^{2(T_dK-2)}}$*. Then, if we perform* T_p *power method iterations and* $T_c = \Omega\big(T_pT_{mix}\log(2\alpha\beta\epsilon^{-1}) + T_{mix}\log(\alpha^{-1}\gamma\sqrt{N})\big)$ *consensus iterations in cloud K-SVD and assume* ***[P1]–[P3]*** *hold, we have for* $i \in \{1, \ldots, N\}$, $t \in \{1, \cdots, T_d\}$*, and* $k \in \{1, \cdots, K\}$

$$\|\mathbf{B}_{i,k,R}^{(t)}\|_2 \leq \begin{cases} 0, & \textit{for } t = 1, k = 1, \\ \varepsilon(1+\zeta)^{t-1}C_4(8C_3NC_4^2+5)^{(t-1)K+k-2}, & \textit{o.w.} \end{cases}$$

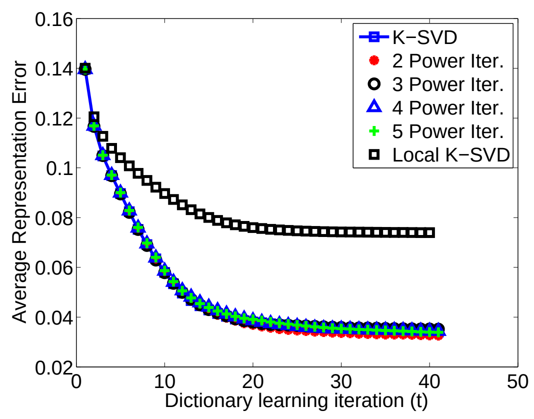
Proof of Theorem 7 along with the proofs of supporting lemmas is given in Section 3.7. Theorem 7 tells us that the error in matrix $\mathbf{E}_{i,k,R}^{(t)}$ can be made arbitrarily small through a



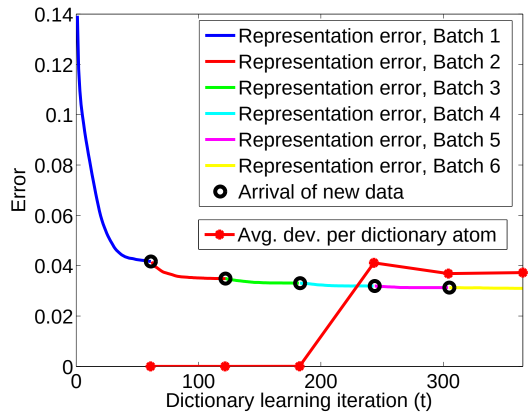
(a) Distributed Power Method



(b) Cloud K-SVD Error



(c) Average Representation Error



(d) Online K-SVD

Figure 3.2: Performance of cloud K-SVD on synthetic data. (a) Average error in eigenvector estimates of distributed power method. (b) Average error in dictionary atoms returned by cloud K-SVD. (c) Average representation error of cloud K-SVD. (d) Average representation error and average deviation per dictionary atom (from centralized dictionary learned in a full-batch setting) of K-SVD in an online setting as a function of dictionary learning iterations.

suitable choice of T_p and ϵ as long as all of the assumptions of Theorem 5 are satisfied. The proof of Theorem 5, given in Section 3.8, relies on these two aforementioned theorems. In particular, the proof of Theorem 5 shows that the assumption on ε in Theorem 7 is satisfied as long as we are performing power method iterations and consensus iterations as required by Theorem 5.

3.5 Numerical Experiments

We present numerical results in this section for demonstrating the usefulness of cloud K-SVD and also validating some of our theoretical results. In the first set of experiments, synthetic data is used to demonstrate efficacy of cloud K-SVD for data representation. Furthermore, behavior of decentralized power method (Steps 7–17 in Algorithm 3) as a function of the number of consensus iterations and deviations in cloud K-SVD dictionaries from centralized dictionary as a function of number of power method iterations are also shown with the help of simulations. In the second set of experiments, MNIST dataset is used to motivate an application of cloud K-SVD that can benefit from collaboration between distributed sites.

3.5.1 Experiments Using Synthetic Data

These experiments correspond to a total of $N = 100$ sites, with each site having $S_i = 500$ local samples in $\mathbb{R}^{20}$ (i.e., $n = 20$). Interconnections between the sites are randomly generated using an Erdős–Rényi graph with parameter $p = 0.5$. In order to generate synthetic data at individual sites, we first generate a dictionary with $K = 50$ atoms, $\mathbf{D} \in \mathbb{R}^{20\times 50}$, with columns uniformly distributed on the unit sphere in $\mathbb{R}^{20}$. Next, we randomly select a 45-column subdictionary of $\mathbf{D}$ for each site and then generate samples for that site using a linear combination of $T_0 = 3$ randomly selected atoms of this subdictionary, followed by addition of white Gaussian noise with variance $\sigma^2 = 0.01$. All data samples in our experiments are also normalized to have unit ℓ_2 norms. Sparse coding in these experiments is performed using an implementation of OMP provided in [117]. Finally, in order to carry out consensus averaging, we generate a doubly-stochastic weight matrix $\mathbf{W}$ according to the local-degree weights method described in [107, Section 4.2].

In our first set of experiments we illustrate the convergence behavior of the decentralized power method component within cloud K-SVD (Steps 7–17 in Algorithm 3) as a function of the number of consensus iterations. The results of these experiments, which are reported in Figure 3.2(a), correspond to five different values of the number of consensus iterations (3, 4, 5, 10, 15) within each iteration of the decentralized power method. Specifically, let $\mathbf{q}$ denote the principal eigenvector of the matrix $\sum_{i=1}^{N} \widehat{\mathbf{M}}_i$ in Algorithm 3 (Step 6) computed using Matlab (ver. 2014a) and $\widehat{\mathbf{q}}_i^{(t_p)}$ denote an estimate of $\mathbf{q}$ obtained at site i after the t_p^{th} iteration of the decentralized power method. Then Figure 3.2(a) plots $\mathbf{E}_{\text{eig}}^{(t_p)}$, which is the average of $\|\mathbf{q}\mathbf{q}^{\mathrm{T}} - \widehat{\mathbf{q}}_i^{(t_p)}\widehat{\mathbf{q}}_i^{(t_p)\mathrm{T}}\|_2$ over all sites $i \in \{1, \dots, N\}$, dictionary update steps $k \in \{1, \dots, K\}$, dictionary learning iterations T_d, and 100 Monte-Carlo trials, as a function of the number of decentralized power method iterations t_p. It can be seen from this figure that the distributed power method of Algorithm 3 hits an *error floor* with increasing number of decentralized power method iterations, where the floor is fundamentally determined by the number of consensus iterations within each power method iteration, as predicted by Theorem 6.

Using the same setup our second set of experiments demonstrate the effectiveness of collaboratively learning a dictionary using cloud K-SVD, as opposed to each site learning a *local dictionary* from its local data using the canonical K-SVD algorithm (referred to as *local K-SVD* in the following). Moreover, these experiments also demonstrate the variations in cloud K-SVD results when we change the number of power method iterations (T_p) and consensus iterations (T_c). In Figure 3.2(c), we plot average representation error, defined as $\frac{1}{nS}\sum_{i=1}^{N}\sum_{j=1}^{S_i}\|\mathbf{y}_{i,j}-\mathbf{D}\mathbf{x}_{i,j}\|_2$, as a function of the number of dictionary learning iterations for three dictionary learning methods, namely, centralized (canonical) K-SVD, cloud K-SVD, and local K-SVD. It can be seen from this figure, which corresponds to an average of 100 Monte-Carlo trials, that cloud K-SVD and centralized K-SVD have similar performance and both of them perform better than local K-SVD. In particular, the local K-SVD error is ≈ 0.06 after 40 iterations, while it is ≈ 0.03 for cloud K-SVD and centralized K-SVD. Notice that changes in the number of power method iterations induce relatively minor changes in the representation error of cloud K-SVD. Next, Figure 3.2(b) highlights the average error in dictionary atoms learned using cloud K-SVD as compared to centralized K-SVD. For this experiment, number of consensus iterations are either $T_c = 1$ or $T_c = 10$, and for each of these values, the number of power method iterations used are $T_p = 2, 3, 4, 5$. These experiments show the effect of changing T_p and T_c on the error in collaborative dictionaries. This error is averaged over all dictionary atoms and sites in each iteration for 100 Monte-Carlo trials, defined as $\mathbf{E}^{(t)}_{\text{average}} = \frac{1}{NK}\sum_{k=1}^{K}\sum_{i=1}^{N}\|d_k^{(t)}\mathbf{d}_k^{(t)^{\mathrm{T}}} - \widehat{\mathbf{d}}_{i,k}^{(t)}\widehat{\mathbf{d}}_{i,k}^{(t)^{\mathrm{T}}}\|_2$. Results in Figure 3.2(b) show that this error in dictionary atoms increases sharply at the start, but it stabilizes after some iterations. Important point to note here is that as we increase the number of power method iterations and consensus iterations we get smaller average error in dictionary atoms as predicted by our analysis.

Next, we discuss the usage of cloud K-SVD in online settings. Since it has already been demonstrated that cloud K-SVD achieves performance similar to that of K-SVD, we focus here on the representation error of centralized K-SVD in online settings. The setup corresponds to a mini-batch of 500 training samples being periodically generated at each site and the assumption that each site has a buffer limit of 1000 samples. Thus only samples from the last two periods can be used for dictionary learning. After arrival of each new mini-batch of training samples, we use the dictionary learned in the last period to warm-start (centralized) K-SVD and carry out 60 dictionary learning iterations. Figure 3.2(d) shows the representation error of the learned dictionary in this case, along with the deviation per dictionary atom when compared to a dictionary learned using full-batch centralized K-SVD. These results are plotted as a function of dictionary learning iterations for six periods, where the ending of a period is marked by a circle. The representation error curve in this figure shows that K-SVD takes more time to converge, but it (and thus cloud K-SVD) is a viable option for online settings. Similarly, the deviation curve shows that while the dictionary error initially increases with the arrival of more data, it stabilizes afterward. Note that further improvements in these results can be obtained by using methods like [118] for active sample selection.

Finally, we perform experiments to report actual values of the parameters C_1–C_4. To this end, we generate samples belonging to $\mathbb{R}^{17}$, where each sample is a linear combination of $T_0 = 3$ atoms of a dictionary $D \in \mathbb{R}^{17\times 40}$. We perform sparse coding in these experiments using the lasso package in Matlab 2014a, while we perform dictionary learning using K-SVD. Average values obtained for parameters C_1–C_4 over 100 Monte-Carlo trials in this case are 0.0586, 0.1633, 4.544, and 1.5947, respectively. Using cloud K-SVD, average values of μ and ν are 9000 and 0.3242, respectively. Based on these values, we get $T_p \approx 16,000$. This suggests that the constants in our bounds are rather loose, and our analysis should mainly be used to provide scaling guidelines.

3.5.2 Classification of MNIST Images

In this section, we demonstrate an application of cloud K-SVD in solving classification problem for a real-world dataset. Some of the methods for discriminative dictionary learning (dictionary learning for classification) in centralized settings include [119–121], while some of the decentralized methods are [122, 123]. The purpose of our experiments here is to show the advantage of collaborating with other sites in the network using cloud K-SVD on real-world data as compared to using only the local data at any given site. It is important to note here that K-SVD is not the best method for solving image classification and that the purpose of these experiments is to show that cloud K-SVD can have similar performance as the centralized K-SVD.

For evaluation of cloud K-SVD on real world dataset, we perform classification of digits $\{0, 3, 5, 8, 9\}$ from MNIST dataset [124]. For each digit 6000 samples are used, where 5000 samples are used for training purposes and remaining 1000 for testing purposes. The data are five-times randomly split into training and test samples. For cloud K-SVD, Erdős–Rényi graph with connectivity parameter $p = 0.5$ (probability of edge between any pair of vertices in the network) is used to generate a network with 10 sites and data is equally distributed among them. Before performing dictionary learning, data is down sampled from $\mathbb{R}^{784}$ to $\mathbb{R}^{256}$. After downsampling, a separate dictionary is learned for each digit using centralized K-SVD, cloud K-SVD, and K-SVD using only local data. Each dictionary has dimensions $\mathbb{R}^{256\times 400}$, i.e.,

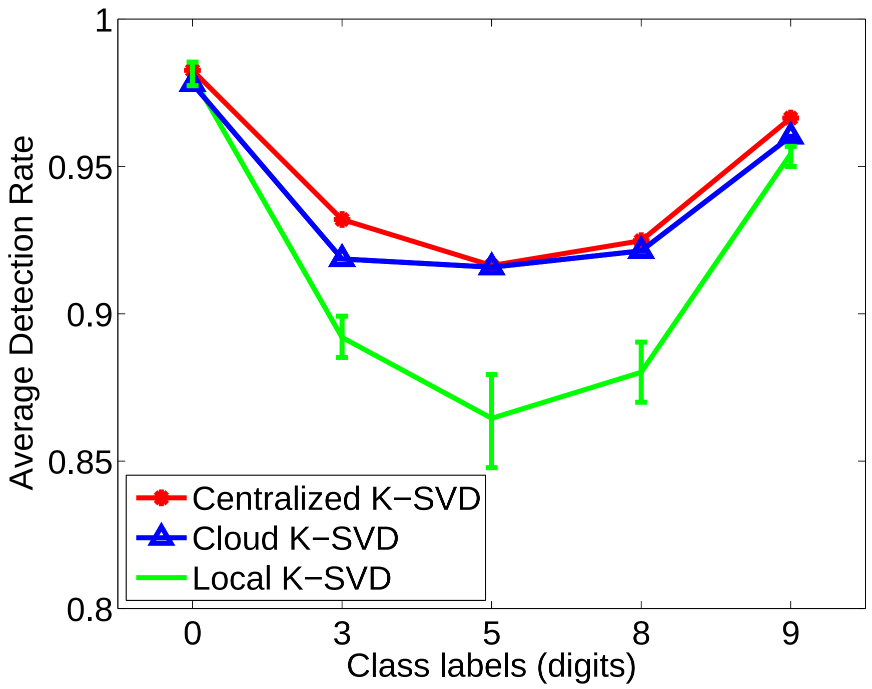


Figure 3.3: Average detection rate for five classes of MNIST dataset using centralized K-SVD, cloud K-SVD, and local K-SVD.

$K = 400$, and sparsity level of $T_0 = 10$ is used. Minimum residue based rule [125, Section II-A] is used for classification, more details on which are given in the following paragraph.

Let $\{\mathbf{D}_c\}_{c=1}^5$ be the set of dictionaries for 5 classes and let $\mathbf{D} = \begin{bmatrix}\mathbf{D}_1 & \mathbf{D}_2 & \mathbf{D}_3 & \mathbf{D}_4 & \mathbf{D}_5\end{bmatrix}$ be the complete dictionary. For any test sample $\mathbf{y}_s$, we perform sparse coding using dictionary D with sparsity constraint of $T_0 = 10$ to get coefficients $\mathbf{x}_s \in \mathbb{R}^{2000}$. Then we partition $\mathbf{x}_s$ into five segment $\{\mathbf{x}_{s,c}\}_{c=1}^5$, where $\mathbf{x}_{s,c}$ are the coefficients corresponding to dictionary $\mathbf{D}_c$ of class c. Next we define residue for class c as $r_c = \|\mathbf{y}_s - \mathbf{D}_c\mathbf{x}_{s,c}\|_2$. Finally, the detected class is given by $c^* = \arg\min_c r_c$. Performance of each method (centralized K-SVD, cloud K-SVD, and local K-SVD) is measured in terms of average detection rate on the test samples, which is defined as:

$$R_c = \frac{\text{Number of samples in class } c \text{ detected correctly}}{\text{Total number of samples of class } c}.$$

Results of this experiment are given in Figure 3.3. We see that centralized K-SVD and cloud K-SVD have comparable performance. But in the case of local K-SVD where we only use the local data for learning representations, classification rate deteriorates considerably. The error bars in local K-SVD show the highest and lowest detection rates achieved among the 10 sites, which highlights the variation in effectiveness of models learned across different sites when using only the local data.

3.6 Proof of Theorem 6

The proof of this theorem relies on a lemma that guarantees that if the estimates obtained at different sites using the decentralized power method are close to the estimate obtained using the centralized power method at the start of a power method iteration then the decentralized estimates remain close to the centralized estimate at the end of that iteration. To prove such a lemma, we first need a result from the literature that characterizes the convergence behavior of *vector* consensus averaging as a function of the number of consensus iterations.

Proposition 6. *[103, Theorem 5] Consider the $n \times 1$ vector sum $\mathbf{z} = \sum_{i=1}^{N} \mathbf{z}_i^{(0)}$ and suppose each vector $\mathbf{z}_i^{(0)}, i = 1, \ldots, N$, is only available at the i^{th} site in our network. Let $\mathbf{b}$ be a vector whose entries are the sum of absolute values of the initial vectors $\mathbf{z}_i^{(0)}$ (i.e., j^{th} entry of $\mathbf{b}$ is $\mathbf{b}_j = \sum_{i=1}^{N} |\mathbf{z}_{i,j}^{(0)}|$) and $\mathbf{z}_i^{(t_c)}$ be the $n \times 1$ vector obtained at the i^{th} site after t_c consensus iterations. Then, fixing any $\delta > 0$, we have that $\left\| \frac{\mathbf{z}_i^{(t_c)}}{[\mathbf{W}_1^{t_c}]_i} - \mathbf{z} \right\|_2 \leq \delta \|\mathbf{b}\|_2 \ \forall i$ as long as the number of consensus iterations satisfies $t_c = \Omega(T_{mix} \log \delta^{-1})$.*

We use Proposition 6 to state and prove the desired lemma.

Lemma 10. *Suppose we are at the start of $(t_p + 1) \leq T_p$ power method iteration. Let $\mathbf{q_c}$ and $\mathbf{q}_{i,d}$ denote the outputs of centralized power method and decentralized power method at i^{th} site after t_p iterations, respectively. Similarly, let $\mathbf{q}'_\mathbf{c}$ and $\mathbf{q}'_{i,d}$ denote the outputs of centralized power method and decentralized power method at i^{th} site after $t_p + 1$ iterations, respectively. Next, fix an $\epsilon \in (0, 1)$, define $\delta = \frac{\alpha_p}{\gamma_p \sqrt{N}} \big(\frac{\epsilon}{2\alpha_p \beta_p} \big)^{3T_p}$, and assume that $\forall i$, $\|\mathbf{q_c} - \mathbf{q}_{i,d}\|_2 + \frac{\delta \gamma_p \sqrt{N}}{\alpha_p} \leq \frac{1}{2\alpha_p \beta_p^2 (2\alpha_p + \delta \gamma_p \sqrt{N})}$. Then, assuming $\Omega(T_{mix} \log \delta^{-1})$ consensus iterations, we have that*

$$\forall i, \ \|\mathbf{q}'_\mathbf{c} - \mathbf{q}'_{i,d}\|_2 \leq (2\alpha_p \beta_p)^3 \left(\max_{i=1,\ldots,N} \|\mathbf{q_c} - \mathbf{q}_{i,d}\|_2 + \frac{\delta \gamma_p \sqrt{N}}{\alpha_p} \right).$$

Proof. Define $\mathbf{v} = \mathbf{M}\mathbf{q_c}$ and $\widehat{\mathbf{v}} = \sum_{i=1}^{N} \mathbf{M}_i \mathbf{q}_{i,\mathsf{d}}$. Next, fix any $i \in \{1, \ldots, N\}$ and let $\widehat{\mathbf{v}}_i$ be the vector obtained at the i^{th} site in Step 15 of Algorithm 3 during the $(t_p + 1)$ iteration of decentralized power method. Notice that $\widehat{\mathbf{v}}_i$ can be expressed as $\widehat{\mathbf{v}}_i = \widehat{\mathbf{v}} + \epsilon_{i,c}$, where $\epsilon_{i,c}$ denotes the error introduced in $\widehat{\mathbf{v}}$ at the i^{th} site due to finite number of consensus iterations. Next, define $r = \|\mathbf{v}\|_2$ and $\widehat{r}_i = \|\widehat{\mathbf{v}}_i\|_2$ and notice that $\mathbf{q}'_\mathsf{c} - \mathbf{q}'_{i,\mathsf{d}} = \mathbf{v}(r^{-1} - \widehat{r}_i^{-1}) + (\mathbf{v} - \widehat{\mathbf{v}}_i)\widehat{r}_i^{-1}$. It therefore follows from the triangle inequality that

$$\|\mathbf{q}'_\mathsf{c} - \mathbf{q}'_{i,\mathsf{d}}\|_2 \leq \|\mathbf{v}\|_2 |r^{-1} - \widehat{r}_i^{-1}| + \|\mathbf{v} - \widehat{\mathbf{v}}_i\|_2 \widehat{r}_i^{-1}. \tag{3.11}$$

We now need to bound $\|\mathbf{v}\|_2$, $|r^{-1} - \widehat{r}_i^{-1}|$, $\|\mathbf{v} - \widehat{\mathbf{v}}_i\|_2$, and $\widehat{r}_i^{-1}$. To this end, notice that $\mathbf{v} - \widehat{\mathbf{v}}_i = \Big[\sum_{i=1}^{N} \mathbf{M}_i (\mathbf{q}_\mathsf{c} - \mathbf{q}_{i,\mathsf{d}})\Big] - \epsilon_{i,c}$. It also follows from Proposition 6 and some manipulations that $\|\epsilon_{i,c}\|_2 \leq \delta\gamma_p\sqrt{N}$. We therefore obtain

$$\|\mathbf{v} - \widehat{\mathbf{v}}_i\|_2 \leq \sum_{i=1}^{N} \|\mathbf{M}_i\|_2 \|\mathsf{q}_\mathsf{c} - \mathbf{q}_{i,\mathsf{d}}\|_2 + \delta\gamma\sqrt{N}. \tag{3.12}$$

Next, notice $|r^{-1} - \widehat{r_i}^{-1}| = |r - \widehat{r}_i| r^{-1}\widehat{r_i}^{-1}$ and further it can be shown that $|r - \widehat{r}_i| \leq r^{-1}|\widehat{r}_i^2 - r^2|$. Now, $|\widehat{r}_i^2 - r^2| = |\widehat{\mathbf{v}}_i^{\mathrm{T}}\widehat{\mathbf{v}}_i - \mathbf{v}^{\mathrm{T}}\mathbf{v}| \leq \|\widehat{\mathbf{v}}_i - \mathbf{v}\|_2(\|\widehat{\mathbf{v}}_i\|_2 + \|\mathbf{v}\|_2)$. Since $\widehat{\mathbf{v}}_i = \widehat{\mathbf{v}} + \epsilon_{i,c}$, it can also be shown that $\|\widehat{\mathbf{v}}_i\|_2 \leq \alpha_p + \delta\gamma_p\sqrt{N}$. In addition, we have $\|\mathbf{v}\|_2 \leq \alpha_p$. Combining these facts with (3.12), we get

$$\begin{aligned} |\widehat{r}_i^2 - r^2| &\leq (2\alpha_p + \delta\gamma_p\sqrt{N}) \left(\sum_{i=1}^{N} \|M_i\|_2 \|\mathsf{q}_\mathsf{c} - \mathbf{q}_{i,\mathsf{d}}\|_2 + \delta\gamma_p\sqrt{N} \right) \\ &\leq (2\alpha_p + \delta\gamma_p\sqrt{N}) \left(\alpha_p \max_i \|\mathsf{q}_\mathsf{c} - \mathbf{q}_{i,\mathsf{d}}\|_2 + \delta\gamma_p\sqrt{N} \right). \end{aligned} \tag{3.13}$$

We can now use this inequality to obtain $|r^{-1} - \widehat{r_i}^{-1}| \leq \widehat{r_i}^{-1}\beta_p^2(2\alpha_p + \delta\gamma_p\sqrt{N})(\alpha_p \max_i \|\mathsf{q}_\mathsf{c} - \mathbf{q}_{i,\mathsf{d}}\|_2 + \delta\gamma_p\sqrt{N})$.

The only remaining quantity we need to bound is $\widehat{r_i}^{-1}$. To this end, notice that $|r - \widehat{r}_i| \geq (r^{-1})^{-1} - (\widehat{r_i}^{-1})^{-1}$. Since $|r - \widehat{r}_i| \leq r^{-1}|\widehat{r}_i^2 - r^2|$, we obtain from (3.13) that

$$(r^{-1})^{-1} - (\widehat{r_i}^{-1})^{-1} \leq \alpha_p r^{-1}(2\alpha_p + \delta\gamma_p\sqrt{N}) \left(\max_i \|\mathsf{q}_\mathsf{c} - \mathbf{q}_{i,\mathsf{d}}\|_2 + \frac{\delta\gamma_p\sqrt{N}}{\alpha_p} \right).$$

It then follows from the lemma's assumptions along with some algebraic manipulations that $\widehat{r_i}^{-1} \leq 2\beta_p$. Finally, plugging the bounds on $\widehat{r_i}^{-1}$, $|r^{-1} - \widehat{r_i}^{-1}|$, $\|\mathbf{v}\|_2$, and $\|\mathbf{v} - \widehat{\mathbf{v}}_i\|_2$ in (3.11), we obtain

$$\begin{aligned} \|\mathsf{q}'_\mathsf{c} - \mathbf{q}'_{i,\mathsf{d}}\|_2 &\leq 2\alpha_p\beta_p^3 \left(\alpha_p \max_i \|\mathsf{q}_\mathsf{c} - \mathbf{q}_{i,\mathsf{d}}\|_2 + \delta\gamma_p\sqrt{N} \right) \left(2\alpha_p + \delta\gamma_p\sqrt{N} \right) \\ &\quad + 2\beta_p \left(\alpha_p \max_i \|\mathsf{q}_\mathsf{c} - \mathbf{q}_{i,\mathsf{d}}\|_2 + \delta\gamma_p\sqrt{N} \right) \\ &= \left(4\alpha_p^3\beta_p^3 + 2\alpha_p^3\beta_p^3 \frac{\delta\gamma_p\sqrt{N}}{\alpha_p} + 2\alpha_p\beta_p \right) \left(\max_i \|q_\mathsf{c} - q_{i,\mathsf{d}}\|_2 + \frac{\delta\gamma_p\sqrt{N}}{\alpha_p} \right). \end{aligned}$$

Finally, $\frac{\delta\gamma_p\sqrt{N}}{\alpha_p} \leq \left(\frac{\epsilon}{2}\right)^{3T_p} < 1$ since (i) $\delta = \frac{\alpha_p}{\gamma_p\sqrt{N}}\left(\frac{\epsilon}{2\alpha_p\beta_p}\right)^{3T_p}$, (ii) $\epsilon < 1$, and (iii) $\alpha_p r^{-1} \geq 1$, which implies $\alpha_p\beta_p \geq 1$. Plugging this into the above expression and noting that $\alpha_p\beta_p \leq \alpha_p^3\beta_p^3$, we obtain the claimed result. □

Lemma 10 provides an understanding of the error accumulation in the decentralized power method due to finite number of consensus iterations in each power method iteration. And while the factor of $(2\alpha_p\beta_p)^3$ in the lemma statement might seem discouraging, the fact that the decentralized power method starts with a zero error helps keep the total error in control. We now formally argue this in the proof of Theorem 6 below.

Proof of Theorem 6. We begin by defining $\mathbf{q_c}$ as the estimate of $\mathbf{u}_1$ obtained using T_p iterations of the centralized power method that is initialized with the same $\mathbf{q}^{\text{init}}$ as the decentralized power method. Next, fix an $i \in \{1, \ldots, N\}$ and notice that

$$\left\|\mathbf{u}_1\mathbf{u}_1^{\mathrm{T}} - \widehat{\mathbf{q}}_i\widehat{\mathbf{q}}_i^{\mathrm{T}}\right\|_2 \leq \|\mathbf{u}_1\mathbf{u}_1^{\mathrm{T}} - \mathbf{q_c}\mathbf{q_c}^{\mathrm{T}}\|_2 + \|\mathbf{q_c}\mathbf{q_c}^{\mathrm{T}} - \widehat{\mathbf{q}}_i\widehat{\mathbf{q}}_i^{\mathrm{T}}\|_2. \tag{3.14}$$

The convergence rate of the centralized power method is well studied and can be expressed as [31]

$$\|\mathbf{u}_1\mathbf{u}_1^{\mathrm{T}} - \mathbf{q_c}\mathbf{q_c}^{\mathrm{T}}\|_2 \leq \tan(\theta)\left|\frac{\lambda_2}{\lambda_1}\right|^{T_p}. \tag{3.15}$$

In order to bound $\|\mathbf{q_c}\mathbf{q_c}^{\mathrm{T}} - \widehat{\mathbf{q}}_i\widehat{\mathbf{q}}_i^{\mathrm{T}}\|_2$, we make use of Lemma 10. To invoke this lemma, we first need to show that the main assumption of the lemma holds for all iterations $t_p \leq (T_p - 1)$. We start with $t_p = 0$ for this purpose and note that $\mathbf{q_c}^{(0)} = \widehat{\mathbf{q}}_i^{(0)} = \mathbf{q}^{init}$, which trivially implies $\|\mathbf{q_c}^{(0)} - \widehat{\mathbf{q}}_i^{(0)}\|_2 + \frac{\delta\gamma_p\sqrt{N}}{\alpha_p} \leq (\frac{\epsilon}{2})^{3T_p}$, where δ is as defined in Lemma 10. Further, under the assumptions of the theorem, it can be shown through elementary algebra that $\left(\frac{\epsilon}{2}\right)^{3T_p} \leq \frac{1}{2\alpha_p\beta_p^2(2\alpha_p+\delta\gamma_p\sqrt{N})}$. We now invoke mathematical induction and claim that the main assumption of Lemma 10 is satisfied for all $t_p \leq m < T_p$. Then we obtain from a recursive application of the statement of the lemma that for $t_p = (m+1)$, we have

$$\begin{aligned}
&\|\mathbf{q_c}^{(m+1)} - \widehat{\mathbf{q}}_i^{(m+1)}\|_2 + \frac{\delta\gamma_p\sqrt{N}}{\alpha_p} \\
&\quad\leq \frac{\delta\gamma_p\sqrt{N}}{\alpha_p}\sum_{i=0}^{m}(2\alpha_p\beta_p)^{3i} \overset{(a)}{\leq} 2\cdot\frac{\delta\gamma_p\sqrt{N}}{\alpha_p}(2\alpha_p\beta_p)^{3m} \\
&\quad= 2\cdot\epsilon^{3T_p}\frac{(2\alpha_p\beta_p)^{3m}}{(2\alpha_p\beta_p)^{3T_p}} \overset{(b)}{\leq} \frac{1}{2\alpha_p\beta_p^2(2\alpha_p+\delta\gamma_p\sqrt{N})},
\end{aligned} \tag{3.16}$$

where (a) follows from the geometric sum and the fact that $(2\alpha_p\beta_p)^3 > 2$, while (b) follows from the theorem assumptions and the fact that $m < T_p$. We have now proved that the main assumption of Lemma 10 holds for all $t_p \leq (T_p - 1)$. In order to compute $\|\mathbf{q_c}\mathbf{q_c}^{\mathrm{T}} - \widehat{\mathbf{q}}_i\widehat{\mathbf{q}}_i^{\mathrm{T}}\|_2$, therefore, we can recursively apply the result of this lemma up to the T_p^{th} iteration to obtain

$$\|\mathbf{q_c} - \widehat{\mathbf{q}}_i\|_2 \leq \frac{\delta\gamma_p\sqrt{N}}{\alpha_p}\sum_{i=0}^{T_p}(2\alpha_p\beta_p)^{3i} \overset{(c)}{\leq} 2\epsilon^{3T_p}, \tag{3.17}$$

where (c) follows from the same arguments as in (3.16). The proof of the theorem now follows by noting the fact that $\|\mathbf{q_c}\mathbf{q_c}^{\mathrm{T}} - \widehat{\mathbf{q}}_i\widehat{\mathbf{q}}_i^{\mathrm{T}}\|_2 \leq (\|\mathbf{q_c}\|_2 + \|\widehat{\mathbf{q}}_i\|_2)\|\mathbf{q_c} - \widehat{\mathbf{q}}_i\|_2 \leq 4\epsilon^{3T_p}$. $\square$

3.7 Proof of Theorem 7

Notice from Algorithm 3 that sparse coding is always performed before update of the first dictionary atom. However, we do not perform sparse coding before updating any other dictionary atom. Due to this distinction, we answer how error is accumulated in matrix $\mathbf{E}_{i,k,R}^{(t)}$ for first dictionary atom differently than for any other dictionary atom. In the following, we first provide an overview of how to bound $\|\mathbf{B}_{i,k+1,R}^{(t)}\|_2$ when we know a bound on $\|\mathbf{B}_{i,k,R}^{(t)}\|_2$. Then we will talk about bounding $\|\mathbf{B}_{i,1,R}^{(t+1)}\|_2$ when we know bounds on $\{\|\mathbf{B}_{i,j,R}^{(t)}\|_2\}_{j=1}^{K}$.

Recall from Step. 5 in Algorithm 3 that

$$\widehat{\mathbf{E}}_{i,k,R}^{(t)} = \mathbf{Y}_i\widetilde{\Omega}_{i,k}^{(t)} - \sum_{j=1}^{k-1}\widehat{\mathbf{d}}_{i,j}^{(t)}\widehat{\mathbf{x}}_{i,j,T}^{(t)}\widetilde{\Omega}_{i,k}^{(t)} - \sum_{j=k+1}^{K}\widehat{\mathbf{d}}_{i,j}^{(t-1)}\widetilde{\mathbf{x}}_{i,j,T}^{(t)}\widetilde{\Omega}_{i,k}^{(t)}.$$

Now, if one assumes that $\widetilde{\Omega}_{k}^{(t)} = \Omega_{k}^{(t)}$, which we will argue is true, then the error in $\mathbf{E}_{i,k,R}^{(t)}$ is due to errors in $\{\mathbf{x}_{i,j,T,R}^{(t)}\}_{j=1}^{K}$ and $\{\mathbf{d}_{j}^{(t)}\}_{j=1}^{K}$. Infact, we will show that $\|\mathbf{B}_{i,k+1,R}^{(t)}\|_2$ can be bounded by knowing bounds on errors in $\widehat{\mathbf{d}}_{i,k}^{(t)}$ and $\mathbf{x}_{i,k,T,R}^{(t)}$ only. Next, recall from Step. 19 in Algorithm 3 that $\widehat{\mathbf{x}}_{i,k,R}^{(t)} = \widehat{\mathbf{d}}_{i,k}^{(t)}\widehat{\mathbf{E}}_{i,k,R}^{(t)^{\mathrm{T}}}$, which means we only need to know a bound on $\mathbf{d}_{k}^{(t)}$ to bound $\|\mathbf{B}_{i,k+1,R}^{(t)}\|_2$. Another challenge for us will be to bound error in $\mathbf{d}_{k}^{(t)}$ from a given bound on $\|\mathbf{B}_{i,k,R}^{(t)}\|_2$. We will accomplish this by noting that there are two sources of error in $\widehat{\mathbf{d}}_{k}^{(t)}$. The first source is the difference in eigenvectors of $\widehat{\mathbf{E}}_{k,R}^{(t)}\widehat{\mathbf{E}}_{k,R}^{(t)^{\mathrm{T}}}$ and $\mathbf{E}_{k,R}^{(t)}\mathbf{E}_{k,R}^{(t)^{\mathrm{T}}}$. We will bound this difference using Proposition 8 in Section 3.9. In order to use this proposition, we will need a bound on $\|\widehat{\mathbf{E}}_{k,R}^{(t)}\widehat{\mathbf{E}}_{k,R}^{(t)^{\mathrm{T}}} - \mathbf{E}_{k,R}^{(t)}\mathbf{E}_{k,R}^{(t)^{\mathrm{T}}}\|_F$, which we will also prove using a given bound on $\|\mathbf{B}_{i,k,R}^{(t)}\|_2$ (Lemma 11). The second source of error in $\widehat{\mathbf{d}}_{k}^{(t)}$ is the error in eigenvector computation, which in our case is due to the decentralized power method. It follows from Theorem 6 and statement of Theorem 7 that this error is bounded by ε. Combining these two sources of error, we will first bound the error in $\widehat{\mathbf{d}}_{k}^{(t)}$ (Lemma 12), and then using this we will finally bound $\|\mathbf{B}_{i,k+1,R}^{(t)}\|_2$ (Lemma 14).

In order to bound $\|\mathbf{B}_{i,1,R}^{(t+1)}\|_2$ when we know bounds on $\{\|\mathbf{B}_{i,j,R}^{(t)}\|_2\}_{j=1}^{K}$, the difference from previous case is that now we cannot write sparse code $\{\widehat{\mathbf{x}}_{i,j,T}^{(t+1)}\}_{j=1}^{K}$ in terms of dictionary atoms $\{\widehat{\mathbf{d}}_{i,j}^{(t)}\}_{j=1}^{K}$. Therefore, in addition to bounding errors in dictionary atoms $\{\widehat{\mathbf{d}}_{i,j}^{(t)}\}_{j=1}^{K}$, we also need to bound errors in sparse codes due to perturbations in dictionaries after iteration t. Since we know $\{\|\mathbf{B}_{i,k,R}^{(t)}\|_2\}_{j=1}^{K}$, we can use the bounds on $\{\widehat{\mathbf{d}}_{i,j}^{(t)}\}_{j=1}^{K}$ derived earlier (Lemma 12). Next, using error bounds on $\{\widehat{\mathbf{d}}_{i,j}^{(t)}\}_{j=1}^{K}$, we can use Proposition 7 in Section 3.9 to bound errors in $\{\widehat{\mathbf{x}}_{i,j,T}^{(t+1)}\}_{j=1}^{K}$. Finally, using these error bounds on $\{\widehat{\mathbf{d}}_{i,j}^{(t)}\}_{j=1}^{K}$ and $\{\widehat{\mathbf{x}}_{i,j,T}^{(t+1)}\}_{j=1}^{K}$ we will bound $\|\mathbf{B}_{i,1,R}^{(t+1)}\|_2$ (Lemma 13). This will be followed by the remaining proof of Theorem 7.

Our first result in support of Theorem 7 shows that the assumption of Proposition 8 in Section 3.9 is satisfied under certain conditions, which will make it possible for us to bound the difference in the principal eigenvector of $\mathbf{E}_{k,R}^{(t)}\mathbf{E}_{k,R}^{(t)^{\mathrm{T}}}$ and $\widehat{\mathbf{E}}_{k,R}^{(t)}\widehat{\mathbf{E}}_{k,R}^{(t)^{\mathrm{T}}}$.

Lemma 11. *Let $\Omega_{i,k}^{(t)}$, $\widetilde{\Omega}_{i,k}^{(t)}$, ε and ζ be as defined in Theorem 7. Fix δ_d as in Theorem 5, and suppose (i)* ***[P1]*–*[P3]*** *are satisfied, (ii) $\Omega_{i,k}^{(t)} = \widetilde{\Omega}_{i,k}^{(t)}$, and (iii) $\varepsilon \leq \frac{\delta_d}{8N\sqrt{n}C_3(1+\zeta)^{T_d-1}C_4^2(8C_3NC_4^2+5)^{2(T_dK-2)}}$. Then $\forall i \in \{1,\dots,N\}$ and for any $t \in \{1,\cdots,T_d\}$ and $k \in \{1,\cdots,K\}$, if*

$$\|\mathbf{B}_{i,k,R}^{(t)}\|_2 \leq \begin{cases} 0, & t=1, k=1, \\ \varepsilon(1+\zeta)^{t-1}C_4(8C_3NC_4^2+5)^{(t-1)K+k-2}, & o.w, \end{cases}$$

then $\Delta\mathbf{M}_k^{(t)} = \mathbf{E}_{k,R}^{(t)}\mathbf{E}_{k,R}^{(t)^{\mathrm{T}}} - \widehat{\mathbf{E}}_{k,R}^{(t)}\widehat{\mathbf{E}}_{k,R}^{(t)^{\mathrm{T}}}$ is bounded as $\|\Delta\mathbf{M}_k^{(t)}\|_F \leq \frac{1}{5C_3}$.

Proof. Since our starting dictionaries are same, therefore, for $(t,k) = (1,1)$ we have $\mathbf{E}_{1,R}^{(1)} = \widehat{\mathbf{E}}_{1,R}^{(1)}$, which means $\Delta\mathbf{M}_k = 0$. Hence, claim is true for $(t,k) = (1,1)$. In the following, proof is provided for the claim for case $(t,k) \neq 1$.

Substituting $\mathbf{B}_{i,k,R}^{(t)}$ in the definition of $\Delta\mathbf{M}_k^{(t)}$, we get

$$\Delta\mathbf{M}_k^{(t)} = \sum_{i=1}^{N} \mathbf{E}_{i,k,R}^{(t)}\mathbf{B}_{i,k,R}^{(t)^{\mathrm{T}}} + \mathbf{B}_{i,k,R}^{(t)}\mathbf{E}_{i,k,R}^{(t)^{\mathrm{T}}} + \mathbf{B}_{i,k,R}^{(t)}\mathbf{B}_{i,k,R}^{(t)^{\mathrm{T}}}.$$

Simple algebraic manipulations, along with submultiplicativity of matrix 2-norm, result in

$$\begin{aligned} \|\Delta\mathbf{M}_k^{(t)}\|_2 &\leq 2\sum_{i=1}^{N}\left(\|\mathbf{E}_{i,k,R}^{(t)}\|_2\|\mathbf{B}_{i,k,R}^{(t)}\|_2 + \|\mathbf{B}_{i,k,R}^{(t)}\|_2^2\right) \\ &\leq 2N\max_i\left(C_4\|\mathbf{B}_{i,k,R}^{(t)}\|_2 + \|\mathbf{B}_{i,k,R}^{(t)}\|_2^2\right), \end{aligned} \tag{3.18}$$

where the last inequality is due to (3.7). Now, using the assumptions on bound of $\|\mathbf{B}_{i,k,R}^{(t)}\|_2$ and ε, we get

$$\begin{aligned} \|\Delta\mathbf{M}_k^{(t)}\|_2 &\leq 2N\varepsilon(1+\zeta)^{t-1}\Big(C_4^2(8C_3NC_4^2+5)^{(t-1)K+k-2} \\ &\quad +\varepsilon(1+\zeta)^{t-1}C_4^2(8C_3NC_4^2+5)^{2(t-1)K+2k-4}\Big) \\ &\leq 2N\varepsilon(1+\zeta)^{t-1}\Big(C_4^2(8C_3NC_4^2+5)^{(t-1)K+k-2} \\ &\quad +\frac{1}{8N\sqrt{n}C_3}\frac{(1+\zeta)^{t-1}C_4^2(8C_3NC_4^2+5)^{2(t-1)K+2k-4}\delta_d}{(1+\zeta)^{T_d-1}C_4^2(8C_3NC_4^2+5)^{2(T_dK-2)K}}\Big) \\ &\overset{(a)}{\leq} 2N\varepsilon(1+\zeta)^{t-1}\left(C_4^2(8C_3NC_4^2+5)^{(t-1)K+k-2} + \frac{1}{8N\sqrt{n}C_3}\right) \\ &\leq 4N\varepsilon(1+\zeta)^{t-1}C_4^2(8C_3NC_4^2+5)^{(t-1)K+k-2}, \end{aligned}$$

where (a) is true because $\frac{(1+\zeta)^{t-1}C_4^2(8C_3NC_4^2+5)^{2(t-1)K+2k-4}\delta_d}{(1+\zeta)^{T_d-1}C_4^2(8C_3NC_4^2+5)^{2(T_dK-2)}} \leq 1$. Finally, using once again the assumption on ε, performing algebraic manipulations and using the fact that $\delta_d \leq 1$, we get

$$\begin{aligned} \|\Delta\mathbf{M}_k^{(t)}\|_2 &\leq \frac{(8C_3NC_4^2+5)^{(t-1)K+k-2}}{2\sqrt{n}C_3(8C_3NC_4^2+5)^{2(T_dK-2)}} \\ &\leq \frac{1}{\sqrt{n}(8C_3NC_4^2+5)^{(T_dK-2)}} \leq \frac{1}{\sqrt{n}(5C_3)}. \end{aligned}$$

Now using the fact that $\text{rank}(\Delta\mathbf{M}_k^{(t)}) \leq n$, we get $\|\Delta\mathbf{M}_k^{(t)}\|_F \leq \sqrt{\text{rank}(\Delta\mathbf{M}_k^{(t)})}\|\Delta\mathbf{M}_k^{(t)}\|_2 \leq \sqrt{n}\|\Delta\mathbf{M}_k^{(t)}\|_2 \leq \frac{1}{5C_3}$. □

We are now ready to prove that if we know a bound on $\|\mathbf{B}_{i,k,R}^{(t)}\|_2$ then we can bound the error in dictionary atom $\widehat{\mathbf{d}}_{i,k}^{(t)}$. This result is given in the following lemma.

Lemma 12. *Let $\Omega_{i,k}^{(t)}$, $\widetilde{\Omega}_{i,k}^{(t)}$, ε and ζ be as defined in Theorem 7, also perform T_c consensus iterations as given in Theorem 7. Now fix δ_d as in Theorem 5, and suppose (i)* ***[P1]–[P3]*** *are satisfied, (ii) $\Omega_{i,k}^{(t)} = \widetilde{\Omega}_{i,k}^{(t)}$, and (iii) $\varepsilon \leq \frac{\delta_d}{8N\sqrt{n}C_3(1+\zeta)^{T_d-1}C_4^2(8C_3NC_4^2+5)^{2(T_dK-2)}}$. Then for all $i \in \{1,\dots,N\}$ and for any $t \in \{1,2,\cdots,T_d\}$ and $k \in \{1,2,\cdots,K\}$ if we know*

$$\|\mathbf{B}_{i,k,R}^{(t)}\|_2 \leq \begin{cases} 0, & t=1, k=1, \\ \varepsilon(1+\zeta)^{t-1}C_4(8C_3NC_4^2+5)^{(t-1)K+k-2}, & o.w, \end{cases}$$

then, $\|\widehat{\mathbf{d}}_{i,k}^{(t)}\widehat{\mathbf{d}}_{i,k}^{(t)^{\mathrm{T}}} - \mathbf{d}_k^{(t)}\mathbf{d}_k^{(t)^{\mathrm{T}}}\|_2 \leq \varepsilon(1+\zeta)^{t-1}(8C_3NC_4^2+5)^{(t-1)K+k-1}$.

Proof. To prove this lemma we first need to decompose error in dictionary atom into two different components i.e., error in principal eigenvector due to perturbation in $\mathbf{E}_{k,R}^{(t)}\mathbf{E}_{k,R}^{(t)^{\mathrm{T}}}$ and error due to decentralized power method. Let $\mathbf{d}_k^{(t)}$ be the updated k^{th} atom of centralized dictionary at iteration t, which is the principal eigenvector of $\mathbf{E}_{k,R}^{(t)}\mathbf{E}_{k,R}^{(t)^{\mathrm{T}}}$. In cloud K-SVD, $\widehat{\mathbf{d}}_{i,k}^{(t)}$ corresponds to the principal eigenvector estimate of $\widehat{\mathbf{E}}_{k,R}^{(t)}\widehat{\mathbf{E}}_{k,R}^{(t)^{\mathrm{T}}}$ obtained at the i^{th} site. Let us denote the true principal eigenvector of $\widehat{\mathbf{E}}_{k,R}^{(t)}\widehat{\mathbf{E}}_{k,R}^{(t)^{\mathrm{T}}}$ by $\widetilde{\mathbf{d}}_k^{(t)}$ and let $\widehat{\mathbf{d}}_{i,k}^{(t)}$ be the eigenvector of $\widehat{\mathbf{E}}_{k,R}^{(t)}\widehat{\mathbf{E}}_{k,R}^{(t)^{\mathrm{T}}}$ computed using decentralized power method at the i^{th} site. Using this notation, notice that $\|\mathbf{d}_k^{(t)}\mathbf{d}_k^{(t)^{\mathrm{T}}} - \widehat{\mathbf{d}}_{i,k}^{(t)}\widehat{\mathbf{d}}_{i,k}^{(t)^{\mathrm{T}}}\|_2 \leq \|\mathbf{d}_k^{(t)}\mathbf{d}_k^{(t)^{\mathrm{T}}} - \widetilde{\mathbf{d}}_k^{(t)}\widetilde{\mathbf{d}}_k^{(t)^{\mathrm{T}}}\|_2 + \|\widetilde{\mathbf{d}}_k^{(t)}\widetilde{\mathbf{d}}_k^{(t)^{\mathrm{T}}} - \widehat{\mathbf{d}}_{i,k}^{(t)}\widehat{\mathbf{d}}_{i,k}^{(t)^{\mathrm{T}}}\|_2$, where the first term is due to perturbation in $\mathbf{E}_{k,R}^{(t)}\mathbf{E}_{k,R}^{(t)^{\mathrm{T}}}$ and the second term is due to imperfect power method and consensus iterations. We can now use Theorem 6 to obtain

$$\begin{aligned}\|\mathbf{d}_k^{(t)}\mathbf{d}_k^{(t)^{\mathrm{T}}} - \widehat{\mathbf{d}}_{i,k}^{(t)}\widehat{\mathbf{d}}_{i,k}^{(t)^{\mathrm{T}}}\|_2 &\leq \|\mathbf{d}_k^{(t)}\mathbf{d}_k^{(t)^{\mathrm{T}}} - \widetilde{\mathbf{d}}_k^{(t)}\widetilde{\mathbf{d}}_k^{(t)^{\mathrm{T}}}\|_2 + \tan(\widehat{\theta}_k^{(t)})\left(\frac{\widehat{\lambda}_{2,k}^{(t)}}{\widehat{\lambda}_{1,k}^{(t)}}\right)^{T_p} + 4\epsilon^{3T_p} \\ &\overset{(a)}{\leq} \|\mathbf{d}_k^{(t)}\mathbf{d}_k^{(t)^{\mathrm{T}}} - \widetilde{\mathbf{d}}_k^{(t)}\widetilde{\mathbf{d}}_k^{(t)^{\mathrm{T}}}\|_2 + \mu\nu^{T_p} + 4\epsilon^{3T_p} \\ &\overset{(b)}{=} \|\mathbf{d}_k^{(t)}\mathbf{d}_k^{(t)^{\mathrm{T}}} - \widetilde{\mathbf{d}}_k^{(t)}\widetilde{\mathbf{d}}_k^{(t)^{\mathrm{T}}}\|_2 + \varepsilon,\end{aligned}$$

where (a) is due to definition of parameters μ and ν in Theorem 5, and (b) is due to definition of ε in Theorem 7.

Next, for symmetric matrices $\mathbf{M}_k^{(t)} = \sum_i \mathbf{E}_{i,k,R}^{(t)}\mathbf{E}_{i,k,R}^{(t)^{\mathrm{T}}}$ and $\widehat{\mathbf{M}}_k^{(t)} = \sum_i \widehat{\mathbf{E}}_{i,k,R}^{(t)}\widehat{\mathbf{E}}_{i,k,R}^{(t)^{\mathrm{T}}}$ such that $\widehat{\mathbf{M}}_k^{(t)} = \mathbf{M}_k^{(t)} + \Delta\mathbf{M}_k^{(t)}$, we can use Lemma 11 and Proposition 8 to find a bound on deviation in principal eigenvector of $\mathbf{M}_k^{(t)}$ due to perturbation $\Delta\mathbf{M}_k^{(t)}$. Since we have from Lemma 11 that

$\|\Delta\mathbf{M}_k^{(t)}\|_F \leq \frac{1}{5C_3}$, it follows from Proposition 8 that

$$\begin{aligned}\|\mathbf{d}_k^{(t)}\mathbf{d}_k^{(t)^{\mathrm{T}}} - \widehat{\mathbf{d}}_{i,k}^{(t)}\widehat{\mathbf{d}}_{i,k}^{(t)^{\mathrm{T}}}\|_2 &\leq 4C_3\|\Delta\mathbf{M}_k^{(t)}\|_2 + \varepsilon \\ &\leq 8C_3N\max_i\left(C_4\|\mathbf{B}_{i,k,R}^{(t)}\|_2 + \|\mathbf{B}_{i,k,R}^{(t)}\|_2^2\right) + \varepsilon, \qquad (3.19)\end{aligned}$$

where the last inequality is due to (3.18). Now using the bound on $\|\mathbf{B}_{i,k,R}^{(t)}\|_2$ in the lemma statement, it can be shown using some algebraic manipulations that

$$\begin{aligned}\|\widehat{\mathbf{d}}_{i,k}^{(t)}\widehat{\mathbf{d}}_{i,k}^{(t)^{\mathrm{T}}} - \mathbf{d}_k^{(t)}\mathbf{d}_k^{(t)^{\mathrm{T}}}\|_2 \leq \varepsilon(1+\zeta)^{t-1}C_4\Big(&8C_3NC_4^2(8C_3NC_4^2+5)^{(t-1)K+k-2} \\ &+8\varepsilon(1+\zeta)^{t-1}C_4C_3N(8C_3NC_4^2+5)^{2(t-1)K+2k-4}+1\Big).\end{aligned}$$

The claim in the lemma now follows by replacing the bound on ε in the parentheses of the above inequality, followed by some manipulations. $\square$

The next lemma shows that if we know bounds on errors in $\{\widehat{\mathbf{E}}_{i,k,R}^{(t)}\}_{k=1}^K$ for any t then we can bound the error in $\widehat{\mathbf{E}}_{i,1,R}^{(t+1)}$.

Lemma 13. *Let $\Omega_{i,k}^{(t)}$, $\widetilde{\Omega}_{i,k}^{(t)}$, ε and ζ be as defined in Theorem 7, also perform T_c consensus iterations as given in Theorem 7. Now fix δ_d as in Theorem 5 and suppose (i)* ***[P1]–[P3]*** *are satisfied, (ii) $\Omega_{i,k}^{(t+1)} = \widetilde{\Omega}_{i,k}^{(t+1)}$, (iii) $\Omega_{i,k}^{(t)} = \widetilde{\Omega}_{i,k}^{(t)}$, and (iv) $\varepsilon \leq \frac{\delta_d}{8N\sqrt{n}C_3(1+\zeta)^{T_d-1}C_4^2(8C_3NC_4^2+5)^{2(T_dK-2)}}$, then for any $t \in \{1,\cdots,T_d-1\}$, and for all $k \in \{1,\cdots,K\}$ and $i \in \{1,\cdots,N\}$, if $\|\mathbf{B}_{i,k,R}^{(t)}\|_2 \leq \varepsilon(1+\zeta)^{t-1}C_4(8C_3C_4^2N+5)^{(t-1)K+k-2}$ then, $\|\mathbf{B}_{i,1,R}^{(t+1)}\|_2 \leq \varepsilon(1+\zeta)^tC_4(8C_3C_4^2N+5)^{tK-1}$.*

Proof. The error in $\widehat{\mathbf{E}}_{i,1,R}^{(t+1)}$ is due to error in dictionary in the previous iteration t and sparse coding at the start of iteration $(t+1)$. Specifically, $\mathbf{B}_{i,1}^{(t+1)} = \mathbf{E}_{i,1}^{(t+1)} - \widehat{\mathbf{E}}_{i,1}^{(t+1)} = \mathbf{Y}_i - \sum_{j=2}^K \mathbf{d}_j^{(t)}\mathbf{x}_{i,j,T}^{(t+1)} - \mathbf{Y}_i + \sum_{j=2}^K \widehat{\mathbf{d}}_{i,j}^{(t)}\widetilde{\mathbf{x}}_{i,j,T}^{(t+1)}$. It then follow that

$$\|\mathbf{B}_{i,1}^{(t+1)}\|_2 \leq \sum_{j=2}^K \|\widehat{\mathbf{d}}_{i,j}^{(t)}\widetilde{\mathbf{x}}_{i,j,T}^{(t+1)} - \mathbf{d}_j^{(t)}\mathbf{x}_{i,j,T}^{(t+1)}\|_2 \leq \sum_{j=1}^K \|\widehat{\mathbf{d}}_{i,j}^{(t)}\widetilde{\mathbf{x}}_{i,j,T}^{(t+1)} - \mathbf{d}_j^{(t)}\mathbf{x}_{i,j,T}^{(t+1)}\|_2.$$

In reality we are interested in finding a bound on $\|\mathbf{B}_{i,1,R}^{(t+1)}\|_2$. But since $\Omega_{i,k}^{(t+1)} = \widetilde{\Omega}_{i,k}^{(t+1)}$ we can define $\mathbf{B}_{i,1,R}^{(t+1)}$ as $\mathbf{B}_{i,1,R}^{(t+1)} = \left(\sum_{j=2}^K\left(\mathbf{Y}_i - \widehat{\mathbf{d}}_{i,j}^{(t)}\widetilde{\mathbf{x}}_{i,j,T}^{(t+1)}\right) - \sum_{j=2}^K\left(\mathbf{Y}_i - \mathbf{d}_j^{(t)}\mathbf{x}_{i,j,T}^{(t+1)}\right)\right)\Omega_{i,1}^{(t+1)}$. It can be seen from this definition that $\mathbf{B}_{i,1,R}^{(t+1)}$ is a submatrix of $\mathbf{B}_{i,1}^{(t+1)}$, which implies

$$\|\mathbf{B}_{i,1,R}^{(t+1)}\|_2 \leq \|\mathbf{B}_{i,1}^{(t+1)}\|_2 \leq \sum_{j=1}^K \|\widehat{\mathbf{d}}_{i,j}^{(t)}\widetilde{\mathbf{x}}_{i,j,T}^{(t+1)} - \mathbf{d}_j^{(t)}\mathbf{x}_{i,j,T}^{(t+1)}\|_2. \qquad (3.20)$$

Now, defining $\widehat{\mathbf{d}}_{i,j}^{(t)} = \mathbf{d}_j^{(t)} + \mathbf{e}_{i,j}^{(t)}$, where $\mathbf{e}_{i,j}^{(t)}$ denotes the error in dictionary atom $\mathbf{d}_j^{(t)}$, and substituting this in (3.20) we get

$$\begin{aligned}
\|\mathbf{B}_{i,1,R}^{(t+1)}\|_2 &\le K \max_j \Big(\|\mathbf{d}_j^{(t)} \widetilde{\mathbf{x}}_{i,j,T}^{(t+1)} - \mathbf{d}_j^{(t)} \mathbf{x}_{i,j,T}^{(t+1)}\|_2 + \|\mathbf{e}_{i,j} \widetilde{\mathbf{x}}_{i,j,T}^{(t+1)}\|_2 \Big) \\
&\le K \max_j \Big(\|\widetilde{\mathbf{x}}_{i,j,T}^{(t+1)} - \mathbf{x}_{i,j,T}^{(t+1)}\|_2 + \|\widehat{\mathbf{d}}_{i,j}^{(t)} - \mathbf{d}_j^{(t)}\|_2 \|\widetilde{\mathbf{x}}_{i,j,T}^{(t+1)}\|_2 \Big) \\
&= K \max_j \Big(\|\widetilde{\mathbf{x}}_{i,j,T}^{(t+1)} - \mathbf{x}_{i,j,T}^{(t+1)}\|_2 + \|\widehat{\mathbf{d}}_{i,j}^{(t)} - \mathbf{d}_j^{(t)}\|_2 \|\widetilde{\mathbf{x}}_{i,j,T}^{(t+1)} + \mathbf{x}_{i,j,T}^{(t+1)} - \mathbf{x}_{i,j,T}^{(t+1)}\|_2 \Big) \\
&\le K \max_j \Big(\|\widetilde{\mathbf{x}}_{i,j,T}^{(t+1)} - \mathbf{x}_{i,j,T}^{(t+1)}\|_2 (1 + \|\widehat{\mathbf{d}}_{i,j}^{(t)} - \mathbf{d}_j^{(t)}\|_2) + \|\widehat{\mathbf{d}}_{i,j}^{(t)} - \mathbf{d}_j^{(t)}\|_2 \|\mathbf{x}_{i,j,T}^{(t+1)}\|_2 \Big). \quad (3.21)
\end{aligned}$$

Now, let $\mathbf{X}^{(t+1)} = \begin{bmatrix} \mathbf{X}_1^{(t+1)} & \mathbf{X}_2^{(t+1)} & \dots & \mathbf{X}_N^{(t+1)} \end{bmatrix} \in \mathbb{R}^{K \times S}$ be the sparse coding matrix associated with the centralized K-SVD (see, e.g, Sec 3.3.1). Notice that $\mathbf{x}_{i,j,T}^{(t+1)}$ is the j^{th} row of $\mathbf{X}_i^{(t+1)}$. It then follows that

$$\|\mathbf{x}_{i,j,T}^{(t+1)}\|_2 \le \sqrt{S_i} \|\mathbf{X}_i^{(t+1)}\|_{\max} \le \sqrt{S_i} \|\mathbf{X}_i^{(t+1)}\|_1.$$

We therefore obtain under P1 that $\|\mathbf{x}_{i,j,T}^{(t+1)}\|_2 \le \sqrt{S_{\max}} \eta_{\tau,\max}$. Next, using the bound on $\|\mathbf{B}_{i,k,R}^{(t)}\|_2$ and applying Lemma 12, we get

$$\|\widehat{\mathbf{d}}_{i,k}^{(t)} \widehat{\mathbf{d}}_{i,k}^{(t)^{\mathrm{T}}} - \mathbf{d}_k^{(t)} \mathbf{d}_k^{(t)^{\mathrm{T}}}\|_2 \le \varepsilon (1+\zeta)^{t-1} C_4 (8 C_3 N C_4^2 + 5)^{(t-1)K+k-1}.$$

Now, under the assumption that both cloud K-SVD and centralized K-SVD use the same $\mathbf{d}^{\mathrm{ref}}$, we have $\widehat{\mathbf{d}}_{i,k}^{(t)^{\mathrm{T}}} \mathbf{d}_k^{(t)} \ge 0$ and therefore it follows from Lemma 16 in Section 3.9 that

$$\begin{aligned}
\|\widehat{\mathbf{d}}_{i,k}^{(t)} - \mathbf{d}_k^{(t)}\|_2 &\le \varepsilon \sqrt{2} (1+\zeta)^{t-1} C_4 (8 C_3 N C_4^2 + 5)^{(t-1)K+k-1} \\
&\overset{(a)}{\le} \sqrt{2} \delta_d \overset{(b)}{\le} 1, \quad (3.22)
\end{aligned}$$

where (a) follows from the assumption on ε and (b) is true for any fixed δ_d as defined in Theorem 5. Using this bound we can write

$$\begin{aligned}
\|\mathbf{D}^{(t)} - \widehat{\mathbf{D}}_i^{(t)}\|_2 \le \|\mathbf{D}^{(t)} - \widehat{\mathbf{D}}_i^{(t)}\|_F &= \sqrt{\sum_{j=1}^{K} \|\widehat{\mathbf{d}}_{i,j}^{(t)} - \mathbf{d}_j^{(t)}\|_2^2} \\
&\le \sqrt{K} \max_{j \in \{1,\cdots,K\}} \|\widehat{\mathbf{d}}_{i,j}^{(t)} - \mathbf{d}_j^{(t)}\|_2 \\
&\le \sqrt{2K} (1+\zeta)^{t-1} \varepsilon C_4 (8 C_3 N C_4^2 + 5)^{tK-1}. \quad (3.23)
\end{aligned}$$

Furthermore, using lemma assumption on ε we get

$$\|\mathbf{D}^{(t)} - \widehat{\mathbf{D}}_i^{(t)}\|_2 \le \sqrt{2K} \delta_d = \min\left\{ \sqrt{K}, \frac{C_1^2 \tau_{\min}}{44} \right\}. \quad (3.24)$$

We can now use (3.24) and Proposition 7 in Section 3.9 to bound $\|\mathbf{x}_{i,j,T}^{(t+1)} - \widetilde{\mathbf{x}}_{i,j,T}^{(t+1)}\|_2$ in (3.21). Notice that Proposition 7 assumes the error in dictionary to be smaller than $\frac{C_1^2 \tau_{\min}}{44}$, which is satisfied by (3.24). Other assumptions of Proposition 7 are satisfied due to **[P1]** and **[P2]**. Therefore, we get $\forall\, i \in \{1,\dots,N\}$ and $j \in \{1,\dots,S_i\}$,

$$\|\mathbf{x}_{i,j}^{(t+1)} - \widetilde{\mathbf{x}}_{i,j}^{(t+1)}\|_2 \leq \frac{3\sqrt{T_0}}{\tau_{\min} C_2}\|\mathbf{D}^{(t)} - \widehat{\mathbf{D}}_i^{(t)}\|_2. \tag{3.25}$$

Now defining $\mathbf{X}_i^{(t+1)}$ and $\widetilde{\mathbf{X}}_i^{(t+1)}$ as before, we note that

$$\begin{aligned}\|\mathbf{x}_{i,j,T}^{(t+1)} - \widetilde{\mathbf{x}}_{i,j,T}^{(t+1)}\|_2 &\leq \sqrt{S_{\max}}\|\mathbf{X}_i^{(t+1)} - \widetilde{\mathbf{X}}_i^{(t+1)}\|_{\max}\\ &\leq \sqrt{S_{\max}} \max_{j\in\{1,\dots,S_i\}} \|\mathbf{x}_{i,j}^{(t+1)} - \widetilde{\mathbf{x}}_{i,j}^{(t+1)}\|_2\\ &\leq \frac{3\sqrt{2KS_{\max}T_0}}{\tau_{\min}C_2}\varepsilon(1+\zeta)^{t-1}C_4(8C_3NC_4^2+5)^{tK-1}, \end{aligned}\tag{3.26}$$

where the last inequality follows from (3.25) and (3.23). Now using bounds on $\|\mathbf{x}_{i,j,T}^{(t+1)}\|_2$ and (3.26) we get the following from (3.21):

$$\begin{aligned}\|\mathbf{B}_{i,1,R}^{(t+1)}\|_2 &\overset{(c)}{\leq} 2K\max_j \|\widetilde{\mathbf{x}}_{i,j,T}^{(t+1)} - \mathbf{x}_{i,j,T}^{(t+1)}\|_2 + \max_j \|\widehat{\mathbf{d}}_{i,j}^{(t)} - \mathbf{d}_j^{(t)}\|_2\|\mathbf{x}_{i,j,T}^{(t)}\|_2\\ &\overset{(d)}{\leq} 2K\frac{3\sqrt{S_{\max}T_0}}{\tau_{\min}C_2}\sqrt{2K}\varepsilon(1+\zeta)^{t-1}C_4(8C_3NC_4^2+5)^{tK-1}\\ &\quad + \varepsilon\sqrt{2}(1+\zeta)^{t-1}C_4(8C_3NC_4^2+5)^{tK-1}\sqrt{S_{\max}}\eta_{\tau,\max}\\ &\overset{(e)}{\leq} \varepsilon(1+\zeta)^{t}C_4(8C_3NC_4^2+5)^{tK-1}.\end{aligned}$$

Here, (c)–(d) follow by application of (3.22) and (3.23), and (e) is by definition of ζ. □

The last lemma that we need bounds $\|\mathbf{B}_{i,k+1,R}^{(t)}\|_2$ when we have a bound on $\|\mathbf{B}_{i,k,R}^{(t)}\|_2$.

Lemma 14. *Let $\Omega_{i,k}^{(t)}$, $\widetilde{\Omega}_{i,k}^{(t)}$, ε, and ζ be as defined in Theorem 7, also perform T_c consensus iterations as given in Theorem 7. Now fix δ_d as in Theorem 5, and suppose (i)* ***[P1]–[P3]*** *are satisfied, (ii) $\Omega_{i,k}^{(t)} = \widetilde{\Omega}_{i,k}^{(t)}$, and (iii) $\varepsilon \leq \frac{\delta_d}{8N\sqrt{n}C_3(1+\zeta)^{T_d-1}C_4^2(8C_3NC_4^2+5)^{2(T_dK-2)}}$. For any fixed $k\in\{1,\cdots,K\}$, $t\in\{1,\cdots,T_d\}$, and all $i\in\{1,\cdots,N\}$, if $\|\mathbf{B}_{i,k,R}^{(t)}\|_2 \leq \varepsilon(1+\zeta)^{t-1}C_4(8C_3C_4^2N+5)^{(t-1)K+k-2}$ then $\|\mathbf{B}_{i,k+1,R}^{(t)}\|_2 \leq \varepsilon(1+\zeta)^{t-1}C_4(8C_3C_4^2N+5)^{(t-1)K+k-1}$.*

Proof. Recall once again that we can write

$$\begin{aligned}\mathbf{B}_{i,k+1,R}^{(t)} &= \widehat{\mathbf{E}}_{i,k+1,R}^{(t)} - \mathbf{E}_{i,k+1,R}^{(t)}\\ &= \sum_{j=k+2}^{K}\Big(\mathbf{d}_j^{(t-1)}\mathbf{x}_{i,j,R}^{(t)} - \widehat{\mathbf{d}}_{i,j}^{(t-1)}\widetilde{\mathbf{x}}_{i,j,R}^{(t)}\Big) - \sum_{j=1}^{k}\Big(\widehat{\mathbf{d}}_{i,j}^{(t)}\widehat{\mathbf{x}}_{i,j,R}^{(t)} - \mathbf{d}_j^{(t)}\mathbf{x}_{i,j,R}^{(t)}\Big),\end{aligned}$$

now using relation $\widehat{\mathbf{x}}_{i,k,R}^{(t)} = \widehat{\mathbf{d}}_{i,k}^{(t)^{\mathrm{T}}}\widehat{\mathbf{E}}_{i,k,R}^{(t)}$ and doing some rearrangements we get,

$$\mathbf{B}_{i,k+1,R}^{(t)} = \widehat{\mathbf{d}}_{i,k}^{(t)}\widehat{\mathbf{d}}_{i,k}^{(t)^{\mathrm{T}}}\widehat{\mathbf{E}}_{i,k,R}^{(t)} - \mathbf{d}_k^{(t)}\mathbf{d}_k^{(t)^{\mathrm{T}}}\mathbf{E}_{i,k,R}^{(t)} - \Big(\mathbf{d}_{k+1}^{(t-1)}\mathbf{x}_{i,k+1,R}^{(t)} - \widehat{\mathbf{d}}_{i,k+1}^{(t-1)}\widetilde{\mathbf{x}}_{i,k+1,R}^{(t)}\Big) + \mathbf{B}_{i,k,R}^{(t)}.$$

It then follows that

$$\begin{aligned}
\|\mathbf{B}^{(t)}_{i,k+1,R}\|_2 &\leq \|\mathbf{B}^{(t)}_{i,k,R}\|_2 + \left\|\widehat{\mathbf{d}}^{(t)}_{i,k}\widehat{\mathbf{d}}^{(t)^{\mathrm{T}}}_{i,k}(\mathbf{E}^{(t)}_{i,k,R} + \mathbf{B}^{(t)}_{i,k,R}) - \mathbf{d}^{(t)}_{k}\mathbf{d}^{(t)^{\mathrm{T}}}_{k}\mathbf{E}^{(t)}_{i,k,R}\right\|_2 \\
&\quad + \left\|\mathbf{d}^{(t-1)}_{k+1}\mathbf{x}^{(t)}_{i,k+1,R} - \widehat{\mathbf{d}}^{(t-1)}_{i,k+1}\widetilde{\mathbf{x}}^{(t)}_{i,k+1,R}\right\|_2 \\
&\overset{(a)}{\leq} 2\|\mathbf{B}^{(t)}_{i,k,R}\|_2 + \|\widehat{\mathbf{d}}^{(t)}_{i,k}\widehat{\mathbf{d}}^{(t)^{\mathrm{T}}}_{i,k} - \mathbf{d}^{(t)}_{k}\mathbf{d}^{(t)^{\mathrm{T}}}_{k}\|_2 C_4 + \left\|\mathbf{d}^{(t-1)}_{k+1}\mathbf{x}^{(t)}_{i,k+1,R} - \widehat{\mathbf{d}}^{(t-1)}_{i,k+1}\widetilde{\mathbf{x}}^{(t)}_{i,k+1,R}\right\|_2 \\
&\overset{(b)}{\leq} \varepsilon C_4(1+\zeta)^{t-1}\Big((8C_3NC_4^2+5)^{(t-1)K+k-2}(8C_3NC_4^2+3) \\
&\quad + \varepsilon 8C_3NC_4^2(1+\zeta)^{t-1}(8C_3NC_4^2+5)^{2(t-1)K+2k-4} + \frac{1}{(1+\zeta)^{t-1}}\Big).
\end{aligned}$$

Here (a) is due to the fact that $\mathbf{E}^{(t)}_{i,k,R}$ is a submatrix of $\mathbf{E}^{(t)}_{i,k}$ and the definition of C_4 in (3.7), (b) is obtained by applying (3.19), using assumption on $\|\mathbf{B}^{(t)}_{i,k,R}\|_2$ and finally using the same procedure as in Lemma 13 after (3.20) to bound $\sum_{j=1}^{K}\left\|\mathbf{d}^{(t-1)}_{j}\mathbf{x}^{(t)}_{i,j,T,R} - \widetilde{\mathbf{d}}^{(t-1)}_{i,j}\widetilde{\mathbf{x}}^{(t)}_{i,j,T,R}\right\|_2$. The proof of the lemma now follows by using the assumption on ε and some algebraic manipulations.

□

The proof of Theorem 7 now can be given by combining Lemmas 11– 14. Since these lemmas require the supports of both centralized and decentralized problems to be the same, the main challenge in proving Theorem 7 lies in showing this fact.

Proof of Theorem 7. We will prove this theorem by mathematical induction over t. To be specific, we will prove the following two cases:

1. For base case, we will show that the claim holds for $\|\mathbf{B}^{(1)}_{i,k,R}\|_2$ $\forall k \in \{1,2,\cdots,K\}$.
2. For induction step we assume that for any $\mathbf{q} \in \{1,2,\cdots,T_d-1\}$ the claim is true for $\|\mathbf{B}^{(q)}_{i,k,R}\|_2$ $\forall k \in \{1,2,\cdots,K\}$ and $\Omega^{(q)}_{i,k} = \widetilde{\Omega}^{(q)}_{i,k}$. Then we need to show that $\Omega^{(q+1)}_{i,k} = \widetilde{\Omega}^{(q+1)}_{i,k}$ and claim holds for $\|\mathbf{B}^{(q+1)}_{i,k,R}\|_2$ $\forall k \in \{1,2,\cdots,K\}$.

Base case: $t = 1\ \forall\ k \in \{1,2,\cdots,K\}$ To prove the base case, we will do mathematical induction over k by fixing $t = 1$. Hence, the first thing we need to prove is that the bound is true for $\|\mathbf{B}^{(1)}_{i,1,R}\|_2$. Since both cloud K-SVD and Centralized K-SVD start with the same initial dictionary, we have $\mathbf{d}^{(0)}_{j} = \widehat{\mathbf{d}}^{(0)}_{i,j},\ \forall\ j \in \{1,2,\cdots,K\}$. Therefore, we get $\Omega^{(1)}_{i,j} = \widetilde{\Omega}^{(1)}_{i,j},\ \forall\ j \in \{1,2,\cdots,K\}$. It then follows that

$$\mathbf{B}^{(1)}_{i,1,R} = \mathbf{E}^{(1)}_{i,1,R} - \widehat{\mathbf{E}}^{(1)}_{i,1,R} = \sum_{j=1}^{K-1}\left(\mathbf{d}^{(0)}_{j}\mathbf{x}^{(1)}_{i,j,T}\Omega^{(1)}_{i,j} - \widehat{\mathbf{d}}^{(0)}_{j}\widetilde{\mathbf{x}}^{(1)}_{i,j,T}\widetilde{\Omega}^{(1)}_{i,j}\right) = 0,$$

thereby proving the claim.

Next, for induction argument we fix $k = p \in \{1, \ldots, K-1\}$ for $t = 1$. Then we need to show that it holds for $k = p+1$. Using the induction assumption we have $\|\mathbf{B}_{i,p,R}^{(1)}\|_2 \leq \varepsilon C_4(8C_3NC_4^2+5)^{p-2}$. Since $\Omega_{i,j}^{(1)} = \widetilde{\Omega}_{i,j}^{(1)}$, we have $\mathbf{B}_{i,p+1,R}^{(1)} = \widehat{\mathbf{E}}_{i,p+1,R}^{(1)} - \mathbf{E}_{i,p+1,R}^{(1)}$. This results in

$$\begin{aligned}
\|\mathbf{B}_{i,p+1,R}^{(1)}\|_2 &= \Big\| \sum_{j=p+2}^{K} \Big(\mathbf{d}_j^{(0)}\mathbf{x}_{i,j,R}^{(1)} - \widehat{\mathbf{d}}_{i,j}^{(0)}\widetilde{\mathbf{x}}_{i,j,R}^{(1)}\Big) - \sum_{j=1}^{p}\Big(\widehat{\mathbf{d}}_{i,j}^{(1)}\widehat{\mathbf{x}}_{i,j,R}^{(1)} - \mathbf{d}_j^{(1)}\mathbf{x}_{i,j,R}^{(1)}\Big)\Big\|_2 \\
&\overset{(a)}{=} \Big\| - \sum_{j=1}^{p}\Big(\widehat{\mathbf{d}}_{i,j}^{(1)}\widehat{\mathbf{x}}_{i,j,R}^{(1)} - \mathbf{d}_j^{(1)}\mathbf{x}_{i,j,R}^{(1)}\Big)\Big\|_2 \\
&= \Big\|\widehat{\mathbf{d}}_{i,p}^{(1)}\widehat{\mathbf{x}}_{i,p,R}^{(1)} - \mathbf{d}_p^{(1)}\mathbf{x}_{i,p,R}^{(1)} + \sum_{j=1}^{p-1}\Big(\widehat{\mathbf{d}}_{i,j}^{(1)}\widehat{\mathbf{x}}_{i,j,R}^{(1)} - \mathbf{d}_j^{(1)}\mathbf{x}_{i,j,R}^{(1)}\Big)\Big\|_2 \\
&= \Big\|\widehat{\mathbf{d}}_{i,p}^{(1)}\widehat{\mathbf{x}}_{i,p,R}^{(1)} - \mathbf{d}_p^{(1)}\mathbf{x}_{i,p,R}^{(1)} + \mathbf{B}_{i,p,R}^{(1)}\Big\|_2, \qquad (3.27)
\end{aligned}$$

where (a) is true because $\mathbf{d}_j^{(0)}\mathbf{x}_{i,j,R}^{(1)} - \widehat{\mathbf{d}}_{i,j}^{(0)}\widetilde{\mathbf{x}}_{i,j,R}^{(0)} = 0$. Substituting $\widehat{\mathbf{x}}_{i,p,R}^{(1)} = \widehat{\mathbf{d}}_{i,p}^{(1)^{\mathrm{T}}}\widehat{\mathbf{E}}_{i,p,R}^{(1)}$, we get

$$\begin{aligned}
\|\mathbf{B}_{i,p+1,R}^{(1)}\|_2 &\leq 2\|\mathbf{B}_{i,p,R}^{(1)}\|_2 + \|\widehat{\mathbf{d}}_{i,p}^{(1)}\widehat{\mathbf{d}}_{i,p}^{(1)^{\mathrm{T}}} - \mathbf{d}_p^{(1)}\mathbf{d}_p^{(1)^{\mathrm{T}}}\|_2\|\mathbf{E}_{i,p,R}^{(1)}\|_2 \\
&\overset{(b)}{\leq} 2\|\mathbf{B}_{i,p,R}^{(1)}\|_2 + \|\widehat{\mathbf{d}}_{i,p}^{(1)}\widehat{\mathbf{d}}_{i,p}^{(1)^{\mathrm{T}}} - \mathbf{d}_p^{(1)}\mathbf{d}_p^{(1)^{\mathrm{T}}}\|_2 C_4 \\
&\overset{(c)}{\leq} 2\|\mathbf{B}_{i,p,R}^{(1)}\|_2 + C_4\Big(8C_3N\max_i\Big(\|\mathbf{B}_{i,p,R}^{(1)}\|_2 C_4 + \|\mathbf{B}_{i,p,R}^{(1)}\|_2^2\Big) + \varepsilon\Big) \\
&\overset{(d)}{\leq} \varepsilon C_4\left((8C_3NC_4^2+5)^{p-2}(8C_3NC_4^2+2) + \varepsilon 8C_3NC_4^2(8C_3NC_4^2+5)^{2p-4} + 1\right).
\end{aligned}$$

Here, (b) is true since $\mathbf{E}_{i,k,R}^{(t)}$ is a submatrix of $\mathbf{E}_{i,k}^{(t)}$ and due to the definition of C_4 in (3.7), (c) is due to (3.19) and (d) follows from using the bound on $\|\mathbf{B}_{i,p,R}^{(1)}\|_2$ and some manipulations. Now using the assumption on ε, we get $\|\mathbf{B}_{i,p+1,R}^{(1)}\|_2 \leq \varepsilon C_4(8C_3NC_4^2+5)^{p-1}$.

Induction step:

(Bound on $\|\mathbf{B}_{i,k,R}^{(q)}\|_2$ holds for $t = q \in \{1, \ldots, T_d - 1\}$ and $\forall\, k \in \{1, 2, \cdots, K\}$.)

We need to show the bound holds for $\|\mathbf{B}_{i,k,R}^{(q+1)}\|_2 \;\forall\, k \in \{1, \ldots, K\}$. To show this, we will be using induction argument over k by fixing $t = q+1$. As base case we bound $\|\mathbf{B}_{i,1,R}^{(q+1)}\|_2$. To bound $\|\mathbf{B}_{i,1,R}^{(q+1)}\|_2$, we will be using Lemma 13, which assumes $\widetilde{\Omega}_{i,1}^{(q+1)} = \Omega_{i,1}^{(q+1)}$. Using the induction assumptions, we get the following bound on error in dictionary $\widehat{\mathbf{D}}_i^{(q)}$ using Lemma 12 and performing same steps as we carried out in Lemma 13 to get (3.22) and (3.23): $\|\mathbf{D}^{(q)} - \widehat{\mathbf{D}}_i^{(q)}\|_2 \leq \varepsilon\sqrt{2K}C_4(8C_3NC_4^2+5)^{(q-1)K+k-2}$. Using the assumption on ε, we then have $\|\mathbf{D}^{(q)} - \widehat{\mathbf{D}}_i^{(q)}\|_2 \leq \delta_d\sqrt{2K}$. It then follows from arguments similar to the ones made in Lemma 13 that $\Omega_{i,1}^{(q+1)} = \widetilde{\Omega}_{i,1}^{(q+1)}$. We can now use Lemma 13 to bound $\|\mathbf{B}_{i,1,R}^{(q+1)}\|_2 \leq \varepsilon(1+\zeta)^q C_4(8NC_3C_4^2+5)^{qK-1}$. Having proved the base case, we now suppose that the claim is true for some $k = p \in \{1, \ldots, K-1\}$. We then need to show it holds for $\|\mathbf{B}_{i,p+1,R}^{(q+1)}\|_2$. That claim, however, simply follows from Lemma 14. This concludes the proof of theorem. $\square$

3.8 Proof of Theorem 5

To prove Theorem 5 we need an upper bound on error in matrices $\widehat{\mathbf{E}}_{i,k,R}^{(t)}$, which is given by Theorem 7. Applying Theorem 7 to get a bound on the error in dictionary atom $\widehat{\mathbf{d}}_{i,k}^{(t)}$ is a trivial task. But before using Theorem 7, we need to show that our assumption on ε is indeed satisfied. In the following, we will prove that the assumption on ε is satisfied if we perform T_p power method and T_c consensus iterations that are given according to the statement of Theorem 5.

Proof of Theorem 5. After T_d iterations of cloud K-SVD, error in any k^{th} dictionary atom $\widehat{\mathbf{d}}_{i,k}^{(T_d)}$ at site i is a function of the error in $\widehat{\mathbf{E}}_{i,k,R}^{(T_d)}$. Specifically, notice from (3.19) that we can write

$$\|\mathbf{d}_k^{(T_d)}\mathbf{d}_k^{(T_d)^{\mathrm{T}}} - \widehat{\mathbf{d}}_{i,k}^{(T_d)}\widehat{\mathbf{d}}_{i,k}^{(T_d)^{\mathrm{T}}}\|_2 \le 8NC_3 \max_i (\|\mathbf{B}_{i,k,R}^{(T_d)}\|_2 C_4 + \|\mathbf{B}_{i,k,R}^{(T_d)}\|_2^2) + \varepsilon. \tag{3.28}$$

We can now upper bound $\|\mathbf{B}_{i,k,R}^{(T_d)}\|_2$ in (3.28) using Theorem 7, but we first need to show that the statement of Theorem 5 implies the assumption on ε in Theorem 7 is satisfied. That is, we need to show $\varepsilon \le \frac{\delta_d}{8N\sqrt{n}C_3(1+\zeta)^{T_d-1}C_4(8C_3NC_4^2+5)^{2(T_dK-2)}}$. Recall that by definition $\varepsilon = \mu\nu^{T_p} + 4\epsilon^{3T_p}$. Substituting this, we must show that

$$\mu\nu^{T_p} + 4\epsilon^{3T_p} \le \frac{\delta_d}{8N\sqrt{n}C_3(1+\zeta)^{T_d-1}C_4(8C_3NC_4^2+5)^{2(T_dK-2)}}.$$

Since $\nu > 0$ and $\epsilon > 0$, therefore, $\mu\nu^{T_p} + 4\epsilon^{3T_p} < \mu(\nu + 4\epsilon^3)^{T_p}$. It is therefore sufficient to show that $\mu(\nu+4\epsilon^3)^{T_p} \le \frac{\delta_d}{8N\sqrt{n}C_3(1+\zeta)^{T_d-1}C_4(8C_3NC_4^2+5)^{2(T_dK-2)}}$ for our selected values of T_p and T_c. Showing that, however, is a simple exercise and is left out for brevity. It therefore follows from Theorem 7 that $\|\mathbf{d}_k^{(T_d)}\mathbf{d}_k^{(T_d)^{\mathrm{T}}} - \widehat{\mathbf{d}}_k^{(T_d)}\widehat{\mathbf{d}}_k^{(T_d)^{\mathrm{T}}}\|_2 \le \varepsilon(1+\zeta)^{T_d-1}(8C_3NC_4^2+5)^{(T_d-1)K+k-1}$. Substituting the upper bound on ε, we get $\|\mathbf{d}_k^{(T_d)}\mathbf{d}_k^{(T_d)^{\mathrm{T}}} - \widehat{\mathbf{d}}_k^{(T_d)}\widehat{\mathbf{d}}_k^{(T_d)^{\mathrm{T}}}\|_2 \le \delta_d$. $\square$

3.9 Other results

In this section, we collect some supporting results that are used in the proofs of our main results.

Lemma 15 (Perturbation of singular values)**.** *Let $\mathbf{D}_2$ be a perturbed version of dictionary $\mathbf{D}_1$ such that $\|\mathbf{D}_1 - \mathbf{D}_2\|_2 \le \epsilon_2$ and let Σ_{T_0} be as defined in Section 3.4.1. Then assuming $\min_{\mathcal{I}\in\Sigma_{T_0}} \sigma_{T_0}\left(\mathbf{D}_{1|\mathcal{I}}\right) \ge \sqrt{C_2'} > \epsilon_2$, we have $\min_{\mathcal{I}\in\Sigma_{T_0}} \sigma_{T_0}\left(\mathbf{D}_{2|\mathcal{I}}\right) \ge \sqrt{C_2'} - \epsilon_2$.*

Proof. Using [126, Theorem 1], perturbation in T_0^{th} singular value of $\mathbf{D}_{1_{|\mathcal{I}}}$ can be bounded as $|\sigma_{T_0}\left(\mathbf{D}_{1_{|\mathcal{I}}}\right) - \sigma_{T_0}\left(\mathbf{D}_{2_{|\mathcal{I}}}\right)| \leq \|\mathbf{D}_{1_{|\mathcal{I}}} - \mathbf{D}_{2_{|\mathcal{I}}}\|_2 \leq \|\mathbf{D}_1 - \mathbf{D}_2\|_2 \leq \epsilon_2$. Using reverse triangular inequality, we therefor get $\forall \mathcal{I} \in \Sigma_{T_0}, \epsilon_2 \geq |\sigma_{T_0}\left(\mathbf{D}_{1_{|\mathcal{I}}}\right)| - |\sigma_{T_0}\left(\mathbf{D}_{2_{|\mathcal{I}}}\right)| \geq \sqrt{C_2'} - |\sigma_{T_0}\left(D_{2_{|\mathcal{I}}}\right)|$, which leads to the claimed result. □

Proposition 7 (Stability of sparse coding)**.** *[127, Theorem 1] Let* $\mathbf{D}_2$ *be a perturbed version of dictionary* $\mathbf{D}_1$ *such that* $\|\mathbf{D}_1 - \mathbf{D}_2\|_2 \leq \epsilon_2$*. Given any sample* $\mathbf{y} \in \mathbb{R}^n$*, suppose sparse codes* $\mathbf{x} \in \mathbb{R}^K$ *and* $\widehat{\mathbf{x}} \in \mathbb{R}^K$ *are computed by solving the lasso problem (3.5) using* $\mathbf{D}_1$ *and* $\mathbf{D}_2$*, respectively. Next, let* $\min_{j \notin supp(\mathbf{x})} \tau - |\langle \mathbf{d}_{1,j}, \mathbf{y} - \mathbf{D}_1 x\rangle| > C_1$*, where* $\mathbf{d}_{1,j}$ *denotes the* j^{th} *atom of* $\mathbf{D}_1$*, and suppose* $\mathbf{D}_1$ *satisfies P2. Then, as long as* $\epsilon_2 \leq \frac{C_1^2 \tau}{44}$*, we have that* $supp(\mathbf{x}) = supp(\widehat{\mathbf{x}})$ *and* $\|\mathbf{x} - \widehat{\mathbf{x}}\|_2 \leq \frac{3\|\mathbf{D}_1 - \mathbf{D}_2\|_2 \sqrt{T_0}}{\tau C_2}$*, where* $T_0 = |supp(\mathbf{x})|$*.*

Note that [127, Theorem 1] also requires $\mathbf{D}_2$ to satisfy **[P2]**. Proposition 7 in its current form, however, is a simple consequence of [127, Theorem 1] and Lemma 15.

Proposition 8 (Perturbation of principal eigenvector)**.** *[31, Chap. 8] Let* $\mathbf{A} \in \mathbb{R}^{n \times n}$ *be a symmetric matrix and define* $\widehat{\mathbf{A}} = \mathbf{A} + \mathbf{E}$ *to be a perturbed, but symmetric version of* $\mathbf{A}$*. Define* $\mathbf{Q} = \begin{bmatrix} \mathbf{q}_1 & | & \mathbf{Q}_2 \end{bmatrix}$ *to be an orthogonal matrix comprising eigenvectors of* $\mathbf{A}$*, where* $\mathbf{q}_1$ *denotes the principal eigenvector of* $\mathbf{A}$*. Next, define* $\mathbf{Q}^{\mathrm{T}}\mathbf{A}\mathbf{Q} = \begin{bmatrix} \lambda & 0 \\ 0 & \Lambda_2 \end{bmatrix}$ *and* $\mathbf{Q}^{\mathrm{T}}\mathbf{E}\mathbf{Q} = \begin{bmatrix} \epsilon & \mathbf{e}^{\mathrm{T}} \\ \mathbf{e} & \mathbf{E}_{22} \end{bmatrix}$*. Then, using* $eig(\Lambda_2)$ *to denote the* $(n-1)$ *smallest eigenvalues of* $\mathbf{A}$*, it follows that if* $g = \min_{\varrho \in eig(\Lambda_2)} |\lambda - \varrho| > 0$*, and* $\|\mathbf{E}\|_F \leq \frac{g}{5}$ *then there exists* $\mathbf{p} \in \mathbb{R}^{n-1}$ *satisfying* $\|\mathbf{p}\|_2 \leq \frac{4}{g}\|\mathbf{e}\|_2$*, such that* $\widehat{\mathbf{q}_1} = \frac{\mathbf{q}_1 + \mathbf{Q}_2\mathbf{p}}{\sqrt{1 + \mathbf{p}^{\mathrm{T}}\mathbf{p}}}$ *is a unit 2-norm principal eigenvector for* $\widehat{\mathbf{A}}$*. Moreover,* $\|\mathbf{q}_1\mathbf{q}_1^{\mathrm{T}} - \widehat{\mathbf{q}}_1\widehat{\mathbf{q}}_1^{\mathrm{T}}\|_2 \leq \frac{4}{g}\|\mathbf{e}\|_2$*.*

Lemma 16 (Errors in vectors and their outerproducts)**.** *For two unit* ℓ_2*-norm vectors* $\mathbf{u}$ *and* $\mathbf{v}$ *if* $\|\mathbf{u}\mathbf{u}^{\mathrm{T}} - \mathbf{v}\mathbf{v}^{\mathrm{T}}\|_2 \leq \epsilon$ *and* $\mathbf{u}^{\mathrm{T}}\mathbf{v} \geq 0$ *then* $\|\mathbf{u} - \mathbf{v}\|_2 \leq \sqrt{2}\epsilon$*.*

Proof. Let $\theta = \angle(\mathbf{u}, \mathbf{v})$ and notice that $\|\mathbf{u}\mathbf{u}^{\mathrm{T}} - \mathbf{v}\mathbf{v}^{\mathrm{T}}\|_2 = \sin\theta$. This implies $1 - \cos^2\theta = \sin^2\theta = \|\mathbf{u}\mathbf{u}^{\mathrm{T}} - \mathbf{v}\mathbf{v}^{\mathrm{T}}\|_2^2 \leq \epsilon^2$. Since $\mathbf{u}$ and $\mathbf{v}$ are unit norm and $\mathbf{u}^{\mathrm{T}}\mathbf{v} \geq 0$, we can write $\cos\theta = \mathbf{u}^{\mathrm{T}}\mathbf{v}$. It then follows that $1 - \mathbf{u}^{\mathrm{T}}\mathbf{v} \leq \frac{\epsilon^2}{1 + \mathbf{u}^{\mathrm{T}}\mathbf{v}} < \epsilon^2$. The claim follows by noting that $\|\mathbf{u} - \mathbf{v}\|_2 = \sqrt{2(1 - \mathbf{u}^{\mathrm{T}}\mathbf{v})}$. □

3.10 Conclusion

In this chapter, we proposed a new dictionary learning algorithm, termed cloud K-SVD, that facilitates collaborative learning of a dictionary that best approximates massive data distributed across geographical regions. Mathematical analysis of proposed method was also provided, which under certain assumptions shows that if we perform enough number of power method and consensus iterations then the proposed algorithm converges to the centralized K-SVD solution. Furthermore, the efficacy of the proposed algorithm was demonstrated through extensive simulations on synthetic and real data.

Chapter 4

Decentralized Methods for Through the Wall Radar Imaging

This chapter considers a distributed network of radars for through-the-wall imaging and provides a solution for accurate indoor scene reconstruction in the presence of multipath propagation. A sparsity-based method is proposed for eliminating ghost targets under imperfect knowledge of interior wall locations. Instead of aggregating and processing the observations at a central fusion station, joint scene reconstruction and estimation of interior wall locations is carried out in a decentralized manner across the network. Using alternating minimization approach, the sparse scene is reconstructed using the recently proposed MDOMP algorithm [128]. Main contribution of this chapter is proposing two decentralized methods for estimating wall locations, *i)* distributed quasi-Newton method (D-QN) which is a gradient based approach and *ii)* decentralized particle swarm optimization (DPSO) which is a greedy method for solving optimization problems. The efficacy of the proposed approaches is demonstrated using numerical simulations.

4.1 Introduction

Through-the-wall radar imaging (TWRI) technology has improved significantly over the last decade. However, effectively dealing with the uncertainty caused by high amount of multipath propagation remains a challenge [129, 130]. A number of approaches, both under conventional and sparse reconstruction frameworks, have been recently proposed in the literature to deal with this challenge [131–138]. However, these methods require prior knowledge of the exact interior layout of the building being imaged to eliminate ghost targets (accumulation of unwanted energy at incorrect target locations) and provide enhanced image quality. In practice, such information may not be perfectly available in advance, resulting in ghost targets and poor image quality.

The problem of TWRI with uncertainties in the room layout information has been addressed by Leigsnering et al. [3]. In [3], this problem is posed as a parametric dictionary learning problem, where the dictionary to be learned is parametrized by unknown wall locations $\mathbf{w}$. Similar to standard dictionary learning [4, 6], the authors in [3] pose parametric dictionary learning as an optimization problem, which is solved using alternating minimization approach involving a dictionary update step and a sparse recovery step (coefficient update). Although the overall objective function considered in traditional dictionary learning is nonconvex, it is convex for each individual step, i.e., when one considers dictionary and coefficient variables separately. Because of this, one can use tools from convex optimization to solve these individual steps. In contrast, parametric dictionary learning in TWRI results in a dictionary update step that is nonconvex and hence needs special attention. One main contribution of [3] is to empirically show that using particle swarm optimization (PSO) and a quasi-Newton method one can recover the dictionary parameter $\mathbf{w}$.

The parametric dictionary learning method proposed in [3] requires data measurements from each individual radar unit to be collected at a centralized location for processing (see Figure 4.1). In practical settings, where we have a large-scale network of radar units interrogating a scene, such as a large building, it is not feasible to accumulate data at one centralized location and hence we need to deploy decentralized methods. Furthermore, decentralized solutions are more robust to a variety of issues, such as bad communication links, node failures, etc., which can occur in adverse settings where we need to deploy radar units, e.g., military or rescue missions. As such a decentralized approach may be preferred over a centralized solution [3] in practical settings. Hence, proposing decentralized variants of PSO and quasi-Newton method for TWRI is the main focus of this chapter.

In terms of prior work on decentralized quasi-Newton method, Eisen et al. [30] have proposed a decentralized variant of quasi-Newton method for convex optimization problems. That solution is not directly applicable to the parametric dictionary learning problem here because our objective function is nonconvex and Lipschitz continuity constants of gradient can vary widely across the domain of function. Due to this using one value of step size as proposed in [30] is not a feasible option here . Because of these reasons, we need to design a decentralized variant of quasi-Newton method that takes into account the specific properties of TWRI objective function. Specifically, since the function gradient can have large Lipschitz continuity

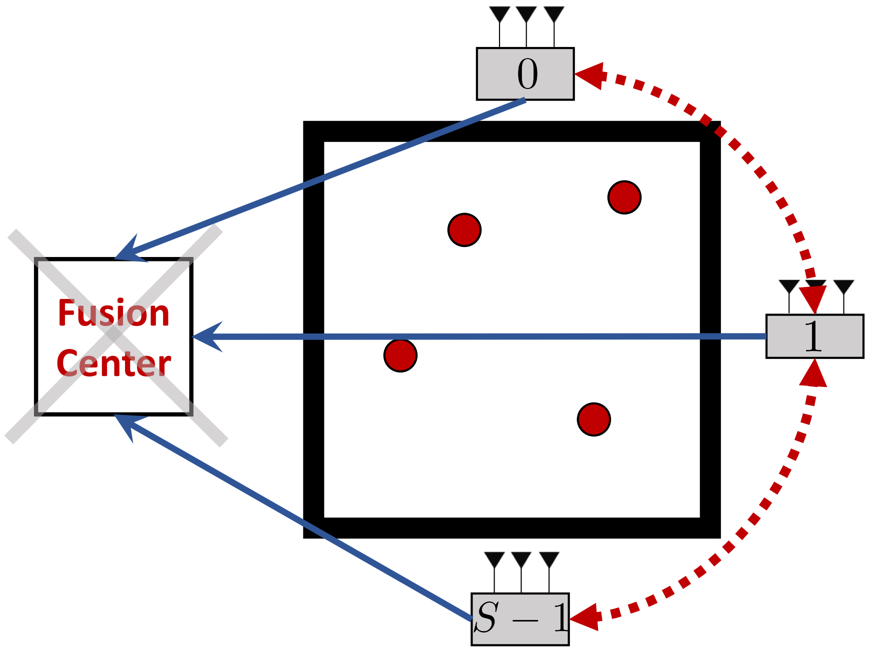


Figure 4.1: Setup in [3] requires accumulation of data at a fusion center using links shown by solid blue lines. In the absence of the fusion center we end up with a distributed TWRI setup where data communication only happens between radar units, shown by dotted red lines.

constant resulting in large values of gradients in the vicinity of stationary points. This can result in slow convergence towards the stationary point. Instead we propose a reconstruction error-based stopping rule that is provided in Section 4.4.3. This procedure cannot provide an accurate solution but it is good enough for the application at hand. Secondly, we cannot use a constant or a decreasing step size, which are common choices in decentralized optimization literature [139–143]; rather, we need to perform line search in order for our method to converge. We are assuming that side information regarding rough estimate of wall locations is available globally across the network and since the problem dimensionality is small as well this makes line search a feasible option in this case.

Our second strategy to solve the optimization problem for estimation of wall locations in distributed settings is based on applying consensus averaging [107] to develop a decentralized variant of the PSO algorithm. There have been previous attempts at developing decentralized versions of PSO [144–147]. A master-slave architecture is employed in [144]; each node is responsible for computing the objective function for a subset of particles, which are then aggregated at the master node to update the particles. A peer-to-peer algorithm for PSO is proposed in [145], where each node is capable of computing the objective function locally and is responsible for updating a subset of particles. However, all of the aforementioned PSO variants are not decentralized in the true sense, since they require centralized control to operate. Further, these algorithms assume each node to be capable of computing the objective function at any particle value. The proposed algorithm eliminates these shortcomings of the aforementioned methods.

In the following, we first formally state the problem in Section 4.2. We provide an overview of methods that will be used for solving distributed TWRI problem in Section 4.3. In Section 4.4, we describe in detail the proposed approaches, while supporting numerical results are provided in Section 4.6. Finally, conclusions are drawn in Section 4.7.

4.2 Problem Formulation

4.2.1 System Model

Consider S radar units located at known positions either along the front wall or surrounding the building being imaged. Each radar unit is equipped with M transmitters and N receivers, where both M and N are assumed to be small. An 'across-units' mode of operation is considered, wherein transmission-reception occurs across multiple radar units. While operating in this mode, each transmitted pulse is received simultaneously by all receivers from all units. We assume that radars are operating in a time division multiplexing manner which enables individual radar units to transmit and receive signals without interference from others and each radar can associate the received signal with a specific transmitter.

Let s_1 and s_2 be the indices of the transmitting and receiving radar units, respectively, where $s_1 = 0, 1, \ldots, S-1$, and $s_2 = 0, 1, \ldots, S-1$. The scene of interest is divided into P grid points, which defines the target space. Let $\sigma_p^{s_1 s_2}$ be the complex reflection coefficient associated with grid point p corresponding to the transmitting unit s_1 and receiving unit s_2, with $\sigma_p^{s_1 s_2} = 0$ representing the absence of a target. Neglecting multipath contributions, the baseband signal transmitted from m-th transmitter (here $m = 0, 1, \ldots, M-1$) of the s_1-th radar unit and recorded at receiver $n = 0, 1, \ldots, N-1$ of the s_2-th radar unit can be expressed as,

$$z_{mn}^{s_1 s_2}(t) = \sum_{p=0}^{P-1} \sigma_p^{s_1 s_2} s(t - mT_r - s_1 M T_r - \tau_{pmn}^{s_1 s_2}) \exp\left(-j2\pi f_c(mT_r + s_1 M T_r + \tau_{pmn}^{s_1 s_2})\right). \quad (4.1)$$

Here, $s(t)$ is the transmitted wideband pulse in complex baseband, f_c is the carrier frequency, T_r is the pulse repetition interval, and $\tau_{pmn}^{s_1 s_2}$ is the propagation delay from transmitter m of unit s_1 to the grid point p and back to the receiver n of unit s_2. We sample $z_{mn}^{s_1 s_2}(t)$ at or above the Nyquist rate to obtain a signal vector $\mathbf{z}_{mn}^{s_1 s_2}$ of length N_T. Stacking signal vectors corresponding to the M transmitters and N receivers, we obtain an $MNN_T \times 1$ measurement vector $\bar{\mathbf{z}}_{s_1 s_2}$, which, using (4.1), can be expressed as, $\bar{\mathbf{z}}_{s_1 s_2} = \mathbf{\Psi}_{s_1 s_2}^{(0)} \boldsymbol{\sigma}_{s_1 s_2}^{(0)}$, where

$\boldsymbol{\sigma}^{(0)}_{s_1s_2} = [\sigma_0^{s_1s_2}, \sigma_1^{s_1s_2}, \dots, \sigma_{P-1}^{s_1s_2}]^{\mathrm{T}}$ with the superscript '(0)' indicating direct path propagation, '$[.]^{\mathrm{T}}$' denotes matrix transpose, and, for $i = 0, \dots, N_T - 1$, the elements of the dictionary matrix $\boldsymbol{\Psi}^{(0)}_{s_1s_2} \in \mathbb{C}^{MNN_T \times P}$ are given by

$$\begin{aligned}\left[\boldsymbol{\Psi}^{(0)}_{s_1s_2}\right]_{i+nN_T+mN_TN,p} &= s(t_i - mT_r - s_1MT_r - \tau_{pmn}^{s_1s_2})\times \\ &\exp\left(-j2\pi f_c(t_i - (mT_r + s_1MT_r + \tau_{pmn}^{s_1s_2}))\right).\end{aligned} \tag{4.2}$$

Next, we assume that multipath for each target is generated due to secondary reflections at one or more interior walls. Parameterizing the interior wall locations as $\mathbf{w} \in \mathbb{R}^3$ and employing geometric optics to model $R-1$ additive multipath contributions in the received signal, we obtain the signal model under multipath propagation for the s_1-th transmitting unit and s_2-th receiving unit as

$$\bar{\mathbf{z}}_{s_1s_2} = \boldsymbol{\Psi}^{(0)}_{s_1s_2}\boldsymbol{\sigma}^{(0)}_{s_1s_2} + \sum_{r=1}^{R-1} \boldsymbol{\Psi}^{(r)}_{s_1s_2}(\mathbf{w})\boldsymbol{\sigma}^{(r)}_{s_1s_2}. \tag{4.3}$$

Here $\boldsymbol{\Psi}^{(r)}_{s_1s_2}$ is defined according to (4.2) with $\tau_{pmn}^{s_1s_2}$ replaced by the propagation delay $\tau_{pmn}^{s_1s_2,(r)}$ between transmitter m, grid point p, and receiver n along the r-th multipath [135]. Note that the multipath time delays $\tau_{mnp}^{s_1s_2,(r)}, r = 1, \dots, R-1$, depend on the wall locations and, therefore, the dictionary matrices $\{\boldsymbol{\Psi}^{(r)}_{s_1s_2}\}_{r=1}^{R-1}$ are all functions of $\mathbf{w}$. Defining $\widetilde{\boldsymbol{\Psi}}_{s_1s_2}(\mathbf{w}) = \left[\boldsymbol{\Psi}^{(0)}_{s_1s_2} \cdots \boldsymbol{\Psi}^{(R-1)}_{s_1s_2}(\mathbf{w})\right]$ and $\widetilde{\boldsymbol{\sigma}}_{s_1s_2} = [\boldsymbol{\sigma}^{(0)^{\mathrm{T}}}_{s_1s_2} \cdots \boldsymbol{\sigma}^{(R-1)^{\mathrm{T}}}_{s_1s_2}]^{\mathrm{T}}$, and assuming additive noise $\bar{\mathbf{n}}$, we can rewrite (4.3):

$$\bar{\mathbf{z}}_{s_1s_2} = \widetilde{\boldsymbol{\Psi}}_{s_1s_2}(\mathbf{w})\widetilde{\boldsymbol{\sigma}}_{s_1s_2} + \bar{\mathbf{n}}_{s_1s_2}. \tag{4.4}$$

4.2.2 Centralized Problem Formulation

In the case of centralized processing, the S^2 measurement vectors, $\{\bar{\mathbf{z}}_{s_1s_2}, s_1 = 0, \dots, S-1, s_2 = 0, \dots, S-1\}$, corresponding to the 'across-units' operation of the S radar units, are communicated to a fusion center where the scene reconstruction is performed [3]. Specifically, the S^2 measurements can be collectively represented as

$$\breve{\mathbf{z}} = \breve{\mathbf{A}}(\mathbf{w})\breve{\boldsymbol{\sigma}} + \breve{\mathbf{n}}, \tag{4.5}$$

where

$$\begin{aligned}&\breve{\mathbf{z}} = \left[\bar{\mathbf{z}}^{\mathrm{T}}_{0\,0}, \dots, \bar{\mathbf{z}}^{\mathrm{T}}_{S-1\,S-1}\right]^{\mathrm{T}},\ \breve{\boldsymbol{\sigma}} = \left[\widetilde{\boldsymbol{\sigma}}^{\mathrm{T}}_{0\,0}, \dots, \widetilde{\boldsymbol{\sigma}}^{\mathrm{T}}_{S-1\,S-1}\right]^{\mathrm{T}}, \\ &\breve{\mathbf{n}} = \left[\bar{\mathbf{n}}^{\mathrm{T}}_{0\,0}, \dots, \bar{\mathbf{n}}^{\mathrm{T}}_{S-1\,S-1}\right]^{\mathrm{T}},\ \text{and} \\ &\breve{\mathbf{A}}(\mathbf{w}) = \mathrm{blkdiag}\{\widetilde{\boldsymbol{\Psi}}_{0\,0}(\mathbf{w}), \dots, \widetilde{\boldsymbol{\Psi}}_{S-1\,S-1}(\mathbf{w})\},\end{aligned} \tag{4.6}$$

and blkdiag$\{.\}$ denotes a block-diagonal matrix.

Given the measurements $\breve{\mathbf{z}}$ in (4.5), the aim is to determine the wall locations $\mathbf{w}$ as well as reconstruct the scene reflectivity vector $\breve{\boldsymbol{\sigma}}$. Since the same physical scene is observed via all paths by the various radar units, the scene reflectivity vector exhibits a group sparse structure [3]. As such, for a regularization parameter λ, the scene recovery and wall location estimation can be posed as the following optimization problem:

$$\min_{\breve{\boldsymbol{\sigma}},\mathbf{w}} \|\breve{\mathbf{z}} - \breve{\mathbf{A}}(\mathbf{w})\breve{\boldsymbol{\sigma}}\|_2^2 + \lambda\|\breve{\boldsymbol{\sigma}}\|_{1,2}, \tag{4.7}$$

where $\|\breve{\boldsymbol{\sigma}}\|_{1,2} = \sum_{p=0}^{P-1} \big\|[\boldsymbol{\sigma}_{0\,0_p}^{(0)}, \ldots, \boldsymbol{\sigma}_{0\,0_p}^{(R-1)}, \ldots, \boldsymbol{\sigma}_{S-1\,S-1_p}^{(0)}, \ldots, \boldsymbol{\sigma}_{S-1\,S-1_p}^{(R-1)}]^{\mathrm{T}}\big\|_2$ and $\boldsymbol{\sigma}_{s_1\,s_{2p}}^{(0)}$ is the p-th element of vector $\boldsymbol{\sigma}_{s_1\,s_2}^{(0)}$. The optimization problem in (4.7) is nonconvex as the matrix $\breve{\mathbf{A}}(\mathbf{w})$ has a nonlinear dependence on the wall locations. An iterative approach proposed in [3] solves (4.7) by alternating between optimization over $\breve{\boldsymbol{\sigma}}$ and $\mathbf{w}$.

4.2.3 Distributed Problem Formulation

The focus of this work is on wall location estimation and scene reconstruction, i.e., solving (4.7), in a decentralized manner across the S radar units, with each radar unit having access to only a subset of the measurements $\breve{\mathbf{z}}$. Substituting $\breve{\mathbf{A}}(\mathbf{w})$, $\breve{\mathbf{z}}$, and $\breve{\boldsymbol{\sigma}}$ from (4.6) in (4.7), we can write the problem as

$$\min_{\breve{\boldsymbol{\sigma}},\mathbf{w}} \sum_{s_2=0}^{S-1}\sum_{s_1=0}^{S-1} \|\bar{\mathbf{z}}_{s_1\,s_2} - \tilde{\boldsymbol{\Psi}}_{s_1\,s_2}(\mathbf{w})\widetilde{\boldsymbol{\sigma}}_{s_1\,s_2}\|_2^2 + \lambda\|\breve{\boldsymbol{\sigma}}\|_{1,2}. \tag{4.8}$$

Using alternating minimization framework, we need to solve this optimization problem in a decentralized manner. For decentralized optimization over $\breve{\boldsymbol{\sigma}}$, we use the Modified Distributed orthogonal matching pursuit (MDOMP) method proposed in [128]. Then for a fixed $\breve{\boldsymbol{\sigma}}$, the objective function is just the first term in (4.8). That is,

$$\min_{\mathbf{w}} f(\mathbf{w}) := \min_{\mathbf{w}} \sum_{s_2=0}^{S-1} f_{s_2}(\mathbf{w}), \tag{4.9}$$

where,

$$f_{s_2}(\mathbf{w}) := \sum_{s_1=0}^{S-1} \|\bar{\mathbf{z}}_{s_1\,s_2} - \tilde{\boldsymbol{\Psi}}_{s_1\,s_2}(\mathbf{w})\widetilde{\boldsymbol{\sigma}}_{s_1\,s_2}\|_2^2$$

is the objective function at radar unit s_2. In this chapter, we develop decentralized variants of quasi-Newton method and particle swarm optimization (PSO) to solve this optimization problem.

4.3 Technical background

4.3.1 Consensus Averaging

For some scalar values $\{x_i\}_{i=0}^{S-1}$ that are distributed across S radar units, consensus averaging provides an iterative method for computing the average $(1/S)\sum_{i=0}^{S-1} x_i$. Let us first represent the connectivity among the distributed radar units by a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{0, 1, \ldots, S-1\}$ denotes the set of nodes (radar units in the underlying application) in a network and $\mathcal{E}$ are the edges defining the interconnection among the nodes, i.e., $(i, i) \in \mathcal{E}$ and $(i, j) \in \mathcal{E}$ when node i can communicate with node j. From graph $\mathcal{G}$, we generate a doubly stochastic matrix $\mathbf{A}$ such that its (i, j)-th entry, $\mathbf{A}_{i,j}$, satisfies the condition $\mathbf{A}_{i,j} = 0,\ \forall (i, j) \notin \mathcal{E}$. Then, starting from $\mathbf{x}^{(0)} = [x_0, \ldots, x_{S-1}]^{\mathrm{T}}$, the update at iteration t_c of consensus averaging is given by

$$\mathbf{x}^{(t_c)} = \mathbf{A}\mathbf{x}^{(t_c-1)}. \tag{4.10}$$

Previous work on consensus averaging [107, 108] shows that if $\mathbf{A}$ is doubly stochastic then as $t_c \to \infty$, each element of $\mathbf{x}^{(t_c)}$ approaches the mean of the values in $\mathbf{x}^{(0)}$.

4.3.2 MDOMP for Decentralized Sparse Scene Recovery

MDOMP has been proposed in [128] for sparse scene recovery in the case of a distributed network of TWRI units. MDOMP is a decentralized version of the orthogonal matching pursuit (OMP) algorithm [148]. At each radar unit, a communication step is performed in each iteration of MDOMP, wherein each radar unit computes a correlation vector using the local measurements only and shares it with all other radar units. Each unit then adds all correlation vectors, selects the index corresponding to the largest element in the correlation vector sum, and updates its set of active indices. The remaining part of the algorithm is similar to OMP.

4.4 Decentralized Quasi-Newton Method for TWRI

4.4.1 Quasi-Newton Method for Optimization

Gradient-based descent direction methods are a popular choice for solving optimization problems [149]. First-order methods like gradient descent are computationally efficient but can result in slow convergence for some problems. To overcome this issue we can use second-order methods such as Newton's method, which for step size γ_t, is given as follows:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \gamma_t \nabla^2 f(\mathbf{w}_t)^{-1} \nabla f(\mathbf{w}_t). \tag{4.11}$$

However, computing the Hessian matrix and its inverse for Newton's method can be computationally prohibitive in practice for high-dimensional problems. Quasi-Newton method resolves this issue by using only gradient information to approximate the Hessian matrix. For an approximation V_t of inverse of the Hessian matrix, i.e., $V_t \approx \nabla^2 f(\mathbf{w}_t)^{-1}$, we can rewrite (4.11) as:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \gamma_t V_t \nabla f(\mathbf{w}_t). \tag{4.12}$$

A number of methods exist in the literature to compute matrix V_t, with BFGS [149, Chap. 8] being one of the most widely used. Using vectors

$$\mathbf{p}_t := \mathbf{w}_{t+1} - \mathbf{w}_t \quad \text{and} \quad \mathbf{q}_t := \nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}_t),$$

BFGS method computes $\mathbf{V}_t$ as follows:

$$\mathbf{V}_{t+1} = \mathbf{V}_t + \left(1 + \frac{\mathbf{q}_t^{\mathrm{T}} \mathbf{V}_t \mathbf{q}_t}{\mathbf{p}_t^{\mathrm{T}} \mathbf{q}_t}\right) \frac{\mathbf{p}_t \mathbf{p}_t^{\mathrm{T}}}{\mathbf{p}_t^{\mathrm{T}} \mathbf{q}_t} - \frac{\mathbf{p}_t \mathbf{q}_t^{\mathrm{T}} \mathbf{V}_t + \mathbf{V}_t \mathbf{q}_t \mathbf{p}_t^{\mathrm{T}}}{\mathbf{p}_t^{\mathrm{T}} \mathbf{q}_t}.$$

For the distributed TWRI problem, the objective function is distributed across radar units; hence, we cannot use centralized quasi-Newton method. Instead, we will employ consensus averaging [107, 108] to develop a decentralized variant of the quasi-Newton method. In the following, we overview consensus averaging before presenting the proposed decentralized quasi-Newton method. Lack of access to the complete objective function in distributed settings does not permit quasi-Newton update as given in (4.12). Instead, we prospose that starting from same initial value of the minimizer $\mathbf{w}_{\text{init}}$ and Hessian approximate $\mathbf{V}_{\text{init}}$ at each radar unit, we compute the update in (4.12) locally at each radar unit as follows:

$$\mathbf{w}_{s_2,t+1} = \mathbf{w}_{s_2,t} - \tau_{s_2,t} \mathbf{V}_{s_2,t} \nabla f_{s_2}(\mathbf{w}_{s_2,t}). \tag{4.13}$$

Note that since we started from same initial values at each radar unit, we can express the global update as:

$$\begin{aligned} \mathbf{w}_{t+1} &= \sum_{s_2=0}^{S-1} \mathbf{w}_{s_2,t+1} \\ &= \sum_{s_2=0}^{S-1} \Big(\frac{1}{S} \mathbf{w}_{s_2,t} - \tau_{s_2,t} \mathbf{V}_{s_2,t} \nabla f_{s_2}(\mathbf{w}_{s_2,t})\Big). \end{aligned} \tag{4.14}$$

Now we can use consensus averaging to compute this summation. Note that we will have numerical errors in the summation due to finite number of consensus iterations. This leads to biased estimates of $\mathbf{q}_t$ and $\mathbf{p}_t$, as shown in Steps 21–22 of Algorithm 4. These biases in $\mathbf{q}_t$ and $\mathbf{p}_t$ will result in an error in the estimate of $\mathbf{V}_t$ (Step 23 of Algorithm 4). Defining $\boldsymbol{\epsilon}_{c,s_2}$ as the

Algorithm 4: Distributed Quasi-Newton Method (D-QN).

Input: Local data $\{\bar{\mathbf{z}}_{0\,0}, \ldots, \bar{\mathbf{z}}_{S-1\,S-1}\}$, $\breve{\boldsymbol{\sigma}}_i$ computed using MDOMP, constraint set $\mathcal{W}$, and a doubly stochastic matrix $\mathbf{A}$.

Initialize: Randomly pick a starting wall position $\widehat{\mathbf{w}}_{s_2,0} \leftarrow \mathbf{w}_{\text{init}}$ and a positive-definite matrix $\mathbf{V}_{s_2,0} \leftarrow \mathbf{V}_{\text{init}}$ at each radar unit s_2, and $t \leftarrow 0$.

```
1:  while stopping rule do
2:      $\widehat{\mathbf{d}}_{s_2,t} \leftarrow -\widehat{\mathbf{V}}_{s_2,t}\nabla f(\widehat{\mathbf{w}}_{s_2,t})$
3:      $\gamma_{s_2,t} \leftarrow \arg\min_\gamma \sum_{s_2} f_{s_2}\big(\widehat{\mathbf{w}}_{s_2,t} + \gamma\widehat{\mathbf{d}}_{s_2,t}\big)$
4:      $\bar{\mathbf{w}}_{s_2,t+1} \leftarrow \widehat{\mathbf{w}}_{s_2,t} + \gamma_{s_2,t}\widehat{\mathbf{d}}_{s_2,t}$
5:      if $\bar{\mathbf{w}}_{s_2,t+1} \in \mathcal{W}$ then
6:          $\widehat{\mathbf{w}}_{s_2,t+1} \leftarrow \bar{\mathbf{w}}_{s_2,t+1}$
7:      else
8:          (Projection onto constraint set)
9:          $\mathbf{P}(\bar{\mathbf{w}}_{s_2,t+1}) = \arg\min_{\mathbf{y}\in\mathcal{W}} \|\bar{\mathbf{w}}_{s_2,t+1} - \mathbf{y}\|_2$
10:         $\widehat{\mathbf{d}}_{s_2,t} \leftarrow \mathbf{P}(\bar{\mathbf{w}}_{s_2,t+1}) - \widehat{\mathbf{w}}_{s_2,t}$
11:         $\gamma_{s_2,t} \leftarrow \arg\min_{\gamma<1} \sum_{s_2} f_{s_2}\big(\widehat{\mathbf{w}}_{s_2,t} + \gamma\widehat{\mathbf{d}}_{s_2,t}\big)$
12:         $\widehat{\mathbf{w}}_{s_2,t+1} \leftarrow \widehat{\mathbf{w}}_{s_2,t} + \gamma_{s_2,t}\widehat{\mathbf{d}}_{s_2,t}$
13:     end if
14:     (Consensus Averaging)
15:     Initialize $t_c \leftarrow 0$ and $\bar{\mathbf{w}}_{s_2,0} \leftarrow \widehat{\mathbf{w}}_{s_2,t+1}$
16:     while stopping rule do
17:         $\bar{\mathbf{w}}_{s_2,t_c+1} \leftarrow \sum_{j\in\mathcal{N}_{s_2}} \mathbf{A}_{s_2,j}\bar{\mathbf{w}}_{s_2,t_c}$
18:         $t_c \leftarrow t_c + 1$
19:     end while
20:     (Update $\widehat{\mathbf{p}}_{s_2,t}$, $\widehat{\mathbf{q}}_{s_2,t}$, and $\widehat{\mathbf{V}}_{s_2,t}$):
21:     $\widehat{\mathbf{p}}_{s_2,t} \leftarrow \widehat{\mathbf{w}}_{s_2,t+1} - \widehat{\mathbf{w}}_{s_2,t}$
22:     $\widehat{\mathbf{q}}_{s_2,t} \leftarrow \nabla\widehat{f}_{s_2}(\widehat{\mathbf{w}}_{s_2,t+1}) - \nabla\widehat{f}_{s_2}(\widehat{\mathbf{w}}_{s_2,t})$
23:     $\widehat{\mathbf{V}}_{s_2,t+1} \leftarrow \widehat{\mathbf{V}}_{s_2,t} + \left(1 + \frac{\widehat{\mathbf{q}}^T_{s_2,t}\widehat{\mathbf{V}}_{s_2,t}\widehat{\mathbf{q}}_{s_2,t}}{\widehat{\mathbf{p}}^T_{s_2,t}\widehat{\mathbf{q}}_{s_2,t}}\right)\frac{\widehat{\mathbf{p}}_{s_2,t}\widehat{\mathbf{p}}^T_{s_2,t}}{\widehat{\mathbf{p}}^T_{s_2,t}\widehat{\mathbf{q}}_{s_2,t}}$
        $-\frac{\widehat{\mathbf{p}}_{s_2,t}\widehat{\mathbf{q}}^T_{s_2,t}\widehat{\mathbf{V}}_{s_2,t}+\widehat{\mathbf{V}}_{s_2,t}\widehat{\mathbf{q}}_{s_2,t}\widehat{\mathbf{p}}^T_{s_2,t}}{\widehat{\mathbf{p}}^T_{s_2,t}\widehat{\mathbf{q}}_{s_2,t}}$
24:     $t \leftarrow t + 1$
25: end while
```

Return: $\widehat{\mathbf{w}}_{s_2,t}$

error in the estimate of $\mathbf{V}_t$ then we can re-write (4.14) as follows:

$$\begin{aligned}
\mathbf{w}_{t+1} &= \sum_{s_2=0}^{S-1}\Big(\frac{1}{S}\mathbf{w}_{s_2,t} - \tau_{s_2,t}\mathbf{V}_{s_2,t}\nabla f_{s_2}(\mathbf{w}_{s_2,t})\Big) \\
&= \sum_{s_2=0}^{S-1}\Big(\frac{1}{S}\mathbf{w}_{s_2,t} - \tau_{s_2,t}(\mathbf{V}_t + \epsilon_{c,s_2})\nabla f_{s_2}(\mathbf{w}_{s_2,t})\Big) \\
&= \sum_{s_2=0}^{S-1}\Big(\frac{1}{S}\mathbf{w}_{s_2,t} - \tau_{s_2,t}\mathbf{V}_t\nabla f_{s_2}(\mathbf{w}_{s_2,t})\Big) + \boldsymbol{\epsilon}_t. \qquad (4.15)
\end{aligned}$$

Thus, if we perform enough consensus iterations such that $\boldsymbol{\epsilon}_t$ stays sufficiently small then we can acheive similar results as the centralized quasi-Newton method.

4.4.2 Projection onto Constraint Set

After consensus averaging, we have estimate of wall locations $\bar{\mathbf{w}}_{s_2,t+1}$ at each radar unit s_2. Assuming we know initially the wall locations with an accuracy of $\pm 0.5m$, if $\mathbf{w}^*$ is the initial given wall location then we constrain our estimates to be in an interval $[\mathbf{w}^* - 0.5, \mathbf{w}^* + 0.5]$. We formally define constraint set as follows:

$$\mathcal{W} := \left\{ \mathbf{w} \in \mathbb{R}^3 : \mathbf{w}^* - 0.5 \leq \mathbf{w} \leq \mathbf{w}^* + 0.5 \right\}. \tag{4.16}$$

In order to project our estimates onto these box constraints, we use the method proposed in [150]. In Algorithm 4, each radar unit has new estimate of wall locations after Step 4. Next, we test whether the new estimate is within the costraint set defined in (4.16). If wall location estimates are outside the constraint set $\mathcal{W}$ then we use the method in [150]. The first step involves projection of $\bar{\mathbf{w}}_{s_2,t}$ onto set $\mathcal{W}$ as shown in Step 9 of Algorithm 4, while the second step finds a descent direction within the constraint set that is the difference between the projection we obtained and the previous iterate as shown in Step 10 of Algorithm 4. Finally, we compute the step size in the descent direction given as Step 11 of Algorithm 4.

4.4.3 Stopping Criterion

Empirical evidence suggests that the function gradient in our problem changes very rapidly, especially in the neighborhood of stationary points. Due to this, using gradient information as a stopping criterion is not feasible. On the other hand, from Figure 4.2, we can see that the TWRI objective function has very small value within a small neighborhood of the global minimum as compared to anywhere else within the constraint set. Using this insight, we use reconstruction error as a stopping criterion.

4.5 Decentralized Particle Swarm Optimization

4.5.1 Overview of Particle Swarm Optimization

In order to optimize an objective function $f(\mathbf{x})$, PSO is initialized with Q particles at positions $\{\mathbf{q}_j\}_{j=1}^{Q}$, which are points in the domain of $f(\mathbf{x})$. We define the variable $\mathbf{q}_{\min,j}$, which contains the value of $\mathbf{q}_j$ corresponding to the obtained minimum value of the objective function thus far for the particle j. In each iteration of PSO, we compute the value of the objective function for each particle $\{\mathbf{q}_j\}_{j=1}^{Q}$ and update the variables $\{\mathbf{q}_{\min,j}\}_{j=1}^{Q}$. Then we update the variable $\mathbf{g}_{\min}$, which is the particle value at which we have observed the minimum value of objective function

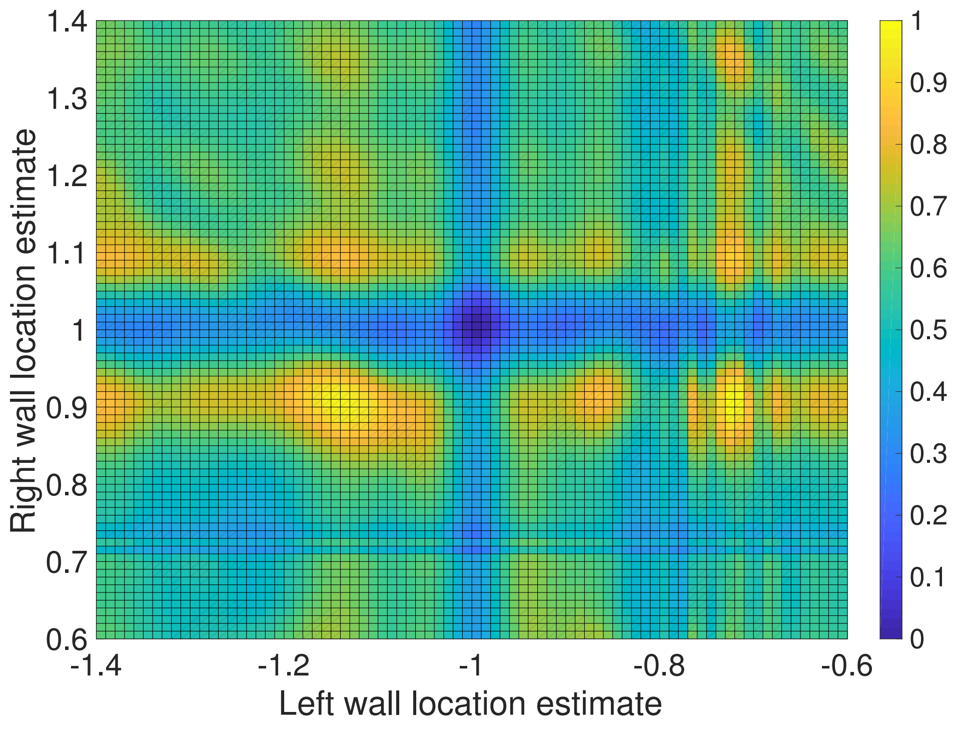


Figure 4.2: Normalized objective function (4.9) when varying wall locations over an interval of ± 0.4 in each dimension around true wall locations.

so far. That is,

$$\mathbf{g}_{\min} = \arg \min_{\mathbf{q} \in \{q_j\}_{j=1}^{Q}} f(\mathbf{q}).$$

Finally, we update the particle velocity $\mathbf{v}_j$ and particle value $\mathbf{q}_j$, where the velocity $\mathbf{v}_j$ provides the displacement of the particle $\mathbf{q}_j$ in the domain of objective function $f(\mathbf{x})$. The velocity update is based on the particle's current value $\mathbf{q}_j$, minimum value at the particle $\mathbf{q}_{\min,j}$, and the minimum value across all particles $\mathbf{g}_{\min}$. The velocity update equation for centralized PSO is the same as for decentralized PSO, which is given in Step 20 of Algorithm 5.

4.5.2 Decentralized approach

We develop a decentralized variant of the PSO algorithm (D-PSO), whose pseudocode is provided in Algorithm 5. Similar to the conventional PSO, we start by initializing with Q particles $\{\mathbf{q}_{s_2,j}^{(0)}\}_{j=1}^{Q}$ at the s_2th node/radar unit. We assume that each radar unit has the same initial values for the particles. One way of achieving this is by initializing particles randomly with the same seed at each node. The first step towards updating the value of a particle consists of computing the objective function (4.8) at the current value of the particle. In the decentralized case, each radar unit can only compute a part of the objective function, as specified in Step 3

Algorithm 5: Decentralized Particle Swarm Optimization algorithm (D-PSO).

```
Input: Local data $\{\bar{\mathbf{z}}_{0\,0},\ldots,\bar{\mathbf{z}}_{S-1\,S-1}\}$, $\breve{\sigma}_i$ computed using MDOMP, and
  doubly-stochastic matrix $\mathbf{A}$.
Initialize: Generate particles with positions $\{\mathbf{q}^{(0)}_{s_2,j}\}_{j=1}^{Q}$ and velocities $\{\mathbf{v}^{(0)}_{s_2,j}\}_{j=1}^{Q}$
  randomly at each site $s_2$. $h_{min,j,s_2} \leftarrow \infty$, $b_{min,s_2} \leftarrow \infty$. $t \leftarrow 0$.

1: while stopping rule do
2:   for $j = 1,\ldots,Q$ do
3:     Calculate at each radar unit $s_2$: $h_{s_2} \leftarrow \sum_{s_1=0}^{S-1} \|\bar{\mathbf{z}}_{s_1\,s_2} - \tilde{\mathbf{\Psi}}_{s_1\,s_2}(\mathbf{q}^{(t)}_{s_2,j})\widetilde{\boldsymbol{\sigma}}_{s_1\,s_2}\|_2^2$
4:     (Inititalize Consensus Averaging) Set $t_c \leftarrow 0$ and $\widehat{h}^{(0)} \leftarrow \left[h_0 \ldots h_{S-1}\right]^{\mathrm{T}}$
5:     while stopping rule do
6:       $\widehat{h}^{(t_c)}_{s_2} \leftarrow \sum_{j\in\mathcal{N}_{s_2}} \mathbf{A}_{s_2,j}\widehat{h}^{(t_c-1)}_{j}$
7:       $t_c \leftarrow t_c + 1$
8:     end while
9:     if $\widehat{h}_{s_2} < h_{min,j,s_2}$ then
10:      $h_{min,j,s_2} \leftarrow \widehat{h}_{s_2}$
11:      $\mathbf{q}_{min,j,s_2} \leftarrow \mathbf{q}^{(t)}_{s_2,j}$
12:    end if
13:  end for
14:  $k_{s_2} \leftarrow \arg\min_j\{h_{min,j,s_2}\}$
15:  if $h_{min,k_{s_2},s_2} < b_{min,s_2}$ then
16:    $b_{min,s_2} \leftarrow h_{min,k_{s_2},s_2}$
17:    $\mathbf{g}_{min,s_2} \leftarrow \{\mathbf{q}_{min,k_{s_2},s_2}\}$
18:  end if
19:  for $j = 1,\ldots,Q$ do
20:    Update velocity: $\mathbf{v}^{(t+1)}_{s_2,j} \leftarrow \mathbf{v}^{(t)}_{s_2,j} + c_1\mathcal{U}(0,1)(\mathbf{q}_{\min,j,s_2} - \mathbf{q}^{(t)}_{i,j}) + c_2\mathcal{U}(0,1)(\mathbf{g}_{\min} - \mathbf{q}^{(t)}_{s_2,j})$
21:    Update positions: $\mathbf{q}^{(t)}_{s_2,j} \leftarrow \mathbf{q}^{(t)}_{s_2,j} + \mathbf{v}^{(t)}_{s_2,j}$.
22:  end for
23:  $t \leftarrow t + 1$
24: end while
Return: $\{\mathbf{q}^{(t)}_{s_2,j}\}_{j=1}^{Q}$.
```

of Algorithm 5. Using consensus averaging, we compute the complete value of the objective function, as delineated in Steps 4–8 of Algorithm 5. Next, in Steps 9–12, we update the minimum objective value achieved by the particle $\mathbf{q}_{s_2,j}$. We repeat Steps 3–12 for all Q particles at the radar unit s_2. The velocity $\mathbf{v}_{s_2,j}$ and particle values $\mathbf{q}_{s_2,j}$ are updated as detailed in Steps 20–21 of Algorithm 5. Note that in Step 20, $\mathcal{U}(0,1)$ is a random number chosen from a uniform distribution over the interval $[0,1]$.

4.6 Simulation Results

We consider a square room with four walls, each of length 2 m. We deploy $S = 5$ radar units, uniformly distributed over an extent of 2 m in crossrange, at a standoff distance of 1.5 m from the front wall. Each radar unit is equipped with $M = 1$ transmitter and $N = 3$ receivers.

Table 4.1: Performance comparison of decentralized gradient descent (D-GD), decentralized quasi-Newton method (D-QN), and decentralized particle swarm optimization (D-PSO) methods.

Optimization Method	Estimates within ± 0.1 accuracy	Estimates within ± 0.01 accuracy
D-GD	94.8%	86.8%
D-QN	96.8%	92.6%
D-PSO	83%	50%

We assume that at any given time instant, only one radar unit transmits and all units receive the reflections. For each transmission, we use a Gaussian pulse with 50% relative bandwidth, modulating a sinusoid of carrier frequency $f_c = 2$ GHz. The received signal at each radar unit is sampled at the Nyquist rate and $N_T = 150$ samples are collected over the interval of interest. In addition to the direct signal, we assume two multipath contributions arising from the side walls, i.e., $R = 3$. Multipath returns are assumed to be attenuated by 6 dB as compared to the direct path signal. Further, the received signals are assumed to be corrupted by complex circular Gaussian noise, resulting in a signal-to-noise ratio (SNR) of 20 dB.

The region of interest covers the room interior and is divided into 32×32 pixels in crossrange and downrange. Four point targets are assumed to be located within the room at distinct locations. In the simulation, starting from a random point in interval $\mathbf{w}^* \pm 0.5$, our goal is to estimate the correct wall positions and reconstruct the scene. For comparison with other gradient methods, we also implemented a decentralized variant of gradient descent method (D-GD) tailored for the TWRI problem. Instead of using the descent direction given in Step 2 of Algorithm 4, we employ gradient as a descent direction for D-GD. Rest of the algorithm works same as Algorithm 4. The results comparing D-PSO, D-QN, and D-GD are provided in Table 4.1. In the simulation, we are only assuming unknown side wall locations and we set the true wall locations to be $[-1 \quad 1]^{\mathrm{T}}$. First and second columns of Table 4.1 provide the percentage of times a method estimates the wall location within ± 0.1 and ± 0.01 of the true value, respectively. We can see that D-QN method estimates wall locations with higher accuracy as compared to the D-GD and D-PSO methods.

4.7 Conclusion

In this chapter we have proposed decentralized variants of quasi-Newton method and particle swarm optimization for wall position estimation in TWRI problem. Due to the properties of the objective function, we propose using a line search approach for step-size computation and we provide a stopping criterion specifically for the TWRI problem for decentralized quasi-Newton method. Finally, simulation results are provided to show the effectiveness of the proposed solution.

Bibliography

[1] Y. LeCun, "The MNIST database of handwritten digits," 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[2] P. Baldi, P. Sadowski, and D. Whiteson, "Searching for exotic particles in high-energy physics with deep learning," *Nature communications*, vol. 5, p. 4308, 2014.

[3] M. Leigsnering, F. Ahmad, M. G. Amin, and A. M. Zoubir, "Parametric dictionary learning for sparsity-based twri in multipath environments," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 52, no. 2, pp. 532–547, 2016.

[4] 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.

[5] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, 2012.

[6] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, no. Jan, pp. 19–60, 2010.

[7] R. Arora, A. Cotter, and N. Srebro, "Stochastic optimization of PCA with capped MSG," in *Proc. Advances in Neural Inform. Process. Syst.*, 2013, pp. 1815–1823.

[8] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, 2013, pp. 2411–2418.

[9] M. Taj and A. Cavallaro, "Distributed and decentralized multicamera tracking," *IEEE Signal Process. Mag.*, vol. 28, no. 3, pp. 46–58, 2011.

[10] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2015, pp. 3061–3070.

[11] S. Krishna and H. Gupta, "Overcoming challenges in connected autonomous vehicles development: Open source vehicular data analytics platform," SAE Technical Paper, Tech. Rep., 2019.

[12] Y. Wu, C. Wu, B. Li, L. Zhang, Z. Li, and F. Lau, "Scaling social media applications into geo-distributed clouds," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 689–702, 2015.

[13] A. Zarezade, U. Upadhyay, H. R. Rabiee, and M. Gomez-Rodriguez, "RedQueen: An online algorithm for smart broadcasting in social networks," in *Proc. 10th ACM Int. Conf. Web Search and Data Mining*, 2017, pp. 51–60.

[14] D. Pohl, A. Bouchachia, and H. Hellwagner, "Online indexing and clustering of social media data for emergency management," *Neurocomputing*, vol. 172, pp. 168–179, 2016.

[15] V. Setty, "Distributed and dynamic clustering for news events," in *Proc. 12th ACM Int. Conf. Distrib. Event-based Syst.*, 2018, pp. 254–257.

[16] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," *arXiv preprint arXiv:1712.09405*, 2017.

[17] S. R. Fitriyani and H. Murfi, "The K-means with mini batch algorithm for topics detection on online news," in *Proc. 4th IEEE Int. Conf. Inf. Commun. Technol.*, 2016, pp. 1–5.

[18] J. Kim, B. Tabibian, A. Oh, B. Schölkopf, and M. Gomez-Rodriguez, "Leveraging the crowd to detect and reduce the spread of fake news and misinformation," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, 2018, pp. 324–332.

[19] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," in *2nd IEEE Int. Conf. Open and Big Data (OBD)*, 2016, pp. 25–30.

[20] J.-J. Yang, J.-Q. Li, and Y. Niu, "A hybrid solution for privacy preserving medical data sharing in the cloud environment," *Future Generation Comput. Syst.*, vol. 43, pp. 74–86, 2015.

[21] T. O. of the Nat. Coordinator for Health Information Technology, "Report on health information blocking," U.S. Department of HHS, Tech. Rep., 2015.

[22] L. J. Kish and E. J. Topol, "Unpatients–why patients should own their medical data," *Nature biotechnology*, vol. 33, no. 9, p. 921, 2015.

[23] A. I. Anton, J. B. Earp, Q. He, W. Stufflebeam, D. Bolchini, and C. Jensen, "Financial privacy policies and the need for standardization," *IEEE Security Privacy*, vol. 2, no. 2, pp. 36–45, 2004.

[24] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction using mini-batches," *J. Mach. Learning Res.*, vol. 13, no. Jan, pp. 165–202, 2012.

[25] R. Johnson and T. Zhang, “Accelerating stochastic gradient descent using predictive variance reduction,” in *Proc. Advances Neural Inform. Process. Syst.*, 2013, pp. 315–323.

[26] M. Li, T. Zhang, Y. Chen, and A. J. Smola, “Efficient mini-batch training for stochastic optimization,” in *Proc. 20th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2014, pp. 661–670.

[27] A. Defazio, F. Bach, and S. Lacoste-Julien, “SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives,” in *Proc. Advances Neural Inform. Process. Syst.*, 2014, pp. 1646–1654.

[28] M. Leigsnering, F. Ahmad, M. G. Amin, and A. M. Zoubir, “Parametric dictionary learning for sparsity-based twri in multipath environments,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 52, no. 2, pp. 532–547, 2016.

[29] S. J. Wright, R. D. Nowak, and M. A. Figueiredo, “Sparse reconstruction by separable approximation,” *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2479–2493, 2009.

[30] M. Eisen, A. Mokhtari, and A. Ribeiro, “Decentralized quasi-newton methods,” *IEEE Trans. Signal Process.*, vol. 65, no. 10, pp. 2613–2628, 2017.

[31] G. H. Golub and C. F. Van Loan, *Matrix computations*, 3rd ed. Baltimore, MD: Johns Hopkins University Press, 2012.

[32] 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.

[33] T. Krasulina, “The method of stochastic approximation for the determination of the least eigenvalue of a symmetrical matrix,” *USSR Comput. Mathematics Math. Physics*, vol. 9, no. 6, pp. 189–195, 1969.

[34] M. K. Warmuth and D. Kuzmin, “Randomized PCA algorithms with regret bounds that are logarithmic in the dimension,” in *Proc. Advances Neural Inform. Process. Syst.*, vol. 19, 2007, pp. 1481–1488. [Online]. Available: http://books.nips.cc/papers/files/nips19/NIPS2006{_}0521.pdf

[35] D. Zhang and L. Balzano, “Global Convergence of a Grassmannian Gradient Descent Algorithm for Subspace Estimation,” in *Proc. 19th Inter. Conf. Artificial Intell. Stat. (AISTATS)*, 2016. [Online]. Available: http://arxiv.org/abs/1506.07405#c#

[36] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *Proc. 19th Intl. Conf. Comput. Stat. (COMPSTAT’10)*, 2010, pp. 177–186.

[37] B. Recht, C. Re, S. Wright, and F. Niu, “Hogwild: A lock-free approach to parallelizing stochastic gradient descent,” in *Proc. Advances Neural Inform. Process. Syst.*, 2011, pp. 693–701.

[38] Z. Allen-Zhu and Y. Li, "First efficient convergence for streaming k-PCA: a global, gap-free, and near-optimal rate," in *Proc. IEEE 58th Annu. Symp. Found. Comput. Sci (FOCS)*, 2017, pp. 487–492.

[39] 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 *Conf. Learn. Theory*, 2016, pp. 1147–1164.

[40] C. Chatterjee, "Adaptive algorithms for first principal eigenvector computation," *Neural Netw.*, vol. 18, pp. 145–159, 2005.

[41] Bin Yang, "Projection approximation subspace tracking," *IEEE Trans. Signal Process.*, vol. 43, no. 1, pp. 95–107, 1995. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=365290

[42] X. G. Doukopoulos and G. V. Moustakides, "Fast and stable subspace tracking," *IEEE Trans. Signal Process.*, vol. 56, no. 4, pp. 1452–1465, 2008.

[43] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Stat.*, pp. 400–407, 1951.

[44] A. Shapiro and T. Homem-de Mello, "On the rate of convergence of optimal solutions of monte carlo approximations of stochastic programs," *SIAM J. Optimization*, vol. 11, no. 1, pp. 70–86, 2000.

[45] J. Linderoth, A. Shapiro, and S. Wright, "The empirical behavior of sampling methods for stochastic programming," *Ann. Operations Res.*, vol. 142, no. 1, pp. 215–241, 2006.

[46] D. Garber and E. Hazan, "Fast and Simple PCA via Convex Optimization," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015.

[47] J. Nie, W. Kotlowski, and M. K. Warmuth, "Online PCA with optimal regret," *J. Mach. Learn. Res.*, vol. 17, no. 173, pp. 1–49, 2016.

[48] C. De Sa, K. Olukotun, and C. Ré, "Global convergence of stochastic gradient descent for some non-convex matrix problems," in *Proc. 32nd Int. Conf. Mach. Learn.*, Jul. 2015, pp. 2332–2341. [Online]. Available: http://arxiv.org/abs/1411.1134

[49] M. Hardt and E. Price, "The noisy power method: A meta algorithm with applications," in *Proc. 27th Int. Conf. Neural Inform. Proces. Syst.*, 2014, pp. 2861–2869. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969033.2969146

[50] O. Shamir, "A Stochastic PCA Algorithm with an Exponential Convergence Rate," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015.

[51] ——, "Fast Stochastic Algorithms for SVD and PCA: Convergence Properties and Convexity," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016.

[52] C.-L. Li, H.-T. Lin, and C.-J. Lu, "Rivalry of Two Families of Algorithms for Memory-Restricted Streaming PCA," in *Proc. 19th Int. Conf. Artificial Intell. Stat.*, 2016, pp. 473–481.

[53] M. F. Balcan, Y. Liang, L. Song, D. Woodruff, and B. Xie, "Communication efficient distributed kernel principal component analysis," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, 2016, pp. 725–734.

[54] D. Garber, O. Shamir, and N. Srebro, "Communication-efficient algorithms for distributed stochastic principal component analysis," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017.

[55] 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.

[56] P. Xu, B. He, C. De Sa, I. Mitliagkas, and C. Re, "Accelerated stochastic power iteration," in *Int. Conf. Artificial Intell. and Statis.*, 2018, pp. 58–67.

[57] E. Hazan, S. Kale, and S. Shalev-Shwartz, "Near-optimal algorithms for online matrix prediction," *SIAM J. Computing*, vol. 46, no. 2, pp. 744–773, 2017.

[58] E. Hazan, K. Y. Levy, and S. Shalev-Shwartz, "On graduated optimization for stochastic non-convex problems," in *Int. Conf. Mach. Learn.*, 2016, pp. 1833–1841.

[59] R. Ge, F. Huang, C. Jin, and Y. Yuan, "Escaping from saddle pointsŮonline stochastic gradient for tensor decomposition," in *In Proc. Conf. on Learning Theory*, 2015, pp. 797–842.

[60] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola, "Stochastic variance reduction for non-convex optimization," in *Int. Conf. Mach. learning*, 2016, pp. 314–323.

[61] S. J. Reddi, S. Sra, B. Póczos, and A. Smola, "Fast stochastic methods for nonsmooth nonconvex optimization," in *Proc. 30th Annu. Conf. Neural Inform. Process. Syst.*, 2016.

[62] Z. Allen-Zhu, "Natasha 2: Faster non-convex optimization than SGD," in *Advances Neural Inf. Process. Syst.*, 2018, pp. 2680–2691.

[63] ——, "How to make the gradients small stochastically: Even faster convex and nonconvex SGD," in *Proc. Advances Neural Inf. Process. Syst.*, 2018, pp. 1157–1167.

[64] Z. Allen-Zhu and E. Hazan, "Variance reduction for faster non-convex optimization," in *Intl. Conf. Mach. Learn.*, 2016, pp. 699–707.

[65] A. Balsubramani, S. Dasgupta, and Y. Freund, "The Fast Convergence of Incremental PCA," in *Proc. Advances Neural Inform. Process. Syst.*, 2013, pp. 3174–3182.

[66] R. Durrett, *Probability: Theory and Examples.* Cambridge University press, 2010.

[67] S. Boucheron, G. Lugosi, and P. Massart, *Concentration inequalities: A nonasymptotic theory of independence.* Oxford University press, 2013.

[68] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *J. Edu. Psych.*, vol. 6, no. 24, pp. 417–441, Sep. 1933.

[69] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (GPCA)," *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 27, no. 12, pp. 1945–1959, Dec. 2005.

[70] T. Zhang, A. Szlam, Y. Wang, and G. Lerman, "Hybrid linear modeling via local best-fit flats," *Intl. J. Comput. Vision*, vol. 100, no. 3, pp. 217–240, Dec. 2012.

[71] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T.-W. Lee, and T. J. Sejnowski, "Dictionary learning algorithms for sparse representation," *Neural Computation*, vol. 15, no. 2, pp. 349–396, Feb. 2003.

[72] I. Tosic and P. Frossard, "Dictionary learning," *IEEE Signal Process. Mag.*, vol. 28, no. 2, pp. 27–38, Mar. 2011.

[73] M. Gastpar, P.-L. Dragotti, and M. Vetterli, "The distributed Karhunen–Loève transform," *IEEE Trans. Inform. Theory*, vol. 52, no. 12, pp. 5177–5196, Dec. 2006.

[74] M. Nokleby and W. U. Bajwa, "Resource tradeoffs in distributed subspace tracking over the wireless medium," in *Proc. 1st IEEE Global Conf. Signal and Inform. Process., GlobalSIP, Symp. on Netw. Theory*, Austin, TX, Dec. 2013, pp. 823–826.

[75] 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, 2011.

[76] A. Scaglione, R. Pagliari, and H. Krim, "The decentralized estimation of the sample covariance," in *Proc. IEEE 42nd Asilomar Conf. on Signals, Syst. Comput.*, 2008, pp. 1722–1726.

[77] S. V. Macua, P. Belanovic, and S. Zazo, "Consensus-based distributed principal component analysis in wireless sensor networks," in *Proc. IEEE 11th Int. Workshop Signal Process. Advances Wireless Commun.*, 2010, pp. 1–5.

[78] R. Tron and R. Vidal, "Distributed computer vision algorithms through distributed averaging," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2011, pp. 57–63.

[79] S. Yoon and V. Pavlovic, "Distributed Probabilistic Learning for Camera Networks with Missing Data," in *Proc. Advances Neural Inform. Process. Syst.*, 2012.

[80] 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.

[81] P. Chainais and C. Richard, “Learning a common dictionary over a sensor network,” in *Proc. IEEE 5th Int. Workshop Comput. Advances Multi-Sensor Adaptive Process.*, 2013, pp. 133–136.

[82] J. Liang, M. Zhang, X. Zeng, and G. Yu, “Distributed dictionary learning for sparse representation in sensor networks,” *IEEE Trans. Image Process.*, vol. 23, no. 6, pp. 2528–2541, June 2014.

[83] J. Chen, C. Richard, A. O. Hero, and A. H. Sayed, “Diffusion LMS for multitask problems with overlapping hypothesis subspaces,” in *IEEE Int. Workshop Mach. Learn. Signal Process.*, 2014, pp. 1–6.

[84] J. Chen, C. Richard, and A. Sayed, “Multitask diffusion adaptation over networks,” *IEEE Trans. Signal Process.*, vol. 62, no. 16, pp. 4129–4144, Aug 2014.

[85] G. Amato, P. Bolettieri, V. Monteiro de Lira, C. I. Muntean, R. Perego, and C. Renso, “Social media image recognition for food trend analysis,” in *Proc. Int. ACM SIGIR Conf. Res. Development Inform. Retrieval*, 2017, pp. 1333–1336.

[86] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Meghini, and C. Vairo, “Deep learning for decentralized parking lot occupancy detection,” *Expert Syst. Appl.*, vol. 72, pp. 327–334, 2017.

[87] Y. Lu and M. Do, “A theory for sampling signals from a union of subspaces,” *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2334–2345, Jun. 2008.

[88] D. L. Swets and J. Weng, “Using discriminant eigenfeatures for image retrieval,” *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 18, no. 8, pp. 831–836, Aug. 1996.

[89] M. Elad and M. Aharon, “Image denoising via sparse and redundant representations over learned dictionaries,” vol. 15, no. 12, pp. 3736–3745, 2006.

[90] J. Mairal, F. Bach, and J. Ponce, “Task-driven dictionary learning,” *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 34, no. 4, pp. 791–804, Apr. 2012.

[91] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, “Online dictionary learning for sparse coding,” in *Proc. ACM 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 689–696.

[92] R. Olfati-Saber, J. A. Fax, and R. M. Murray, “Consensus and cooperation in networked multi-agent systems,” *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[93] J. Speyer, “Computation and transmission requirements for a decentralized linear-quadratic-Gaussian control problem,” *IEEE Trans. Automat. Control*, vol. 24, no. 2, pp. 266–269, Apr. 1979.

[94] J. Tsitsiklis and M. Athans, "Convergence and asymptotic agreement in distributed decision problems," *IEEE Trans. Autom. Control*, vol. 29, no. 1, pp. 42–50, Jan. 1984.

[95] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed linear support vector machines," in *Proc. 9th ACM/IEEE Int. Conf. Inform. Process. Sensor Netw.*, 2010, pp. 35–46.

[96] E. Kokiopoulou and P. Frossard, "Distributed classification of multiple observation sets by consensus," *IEEE Trans. Signal Process.*, vol. 59, no. 1, pp. 104–114, Jan 2011.

[97] S. Lee and A. Nedić, "Distributed random projection algorithm for convex optimization," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 2, pp. 221–229, Apr. 2013.

[98] U. A. Khan, S. Kar, and J. M. F. Moura, "Distributed sensor localization in random environments using minimal number of anchor nodes," *IEEE Trans. Signal Process.*, vol. 57, no. 5, pp. 2000–2016, May 2009.

[99] ——, "DILAND: An algorithm for distributed sensor localization with noisy distance measurements," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1940–1947, Mar. 2010.

[100] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5262–5276, 2010.

[101] S.-Y. Tu and A. H. Sayed, "Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks," *IEEE Trans. Signal Process.*, vol. 60, no. 12, pp. 6217–6234, Dec. 2012.

[102] M. E. Yildiz, F. Ciaramello, and A. Scaglione, "Distributed distance estimation for manifold learning and dimensionality reduction," in *Proc. IEEE Int. Conf. Acous., Speech Signal Process.*, 2009, pp. 3353–3356.

[103] D. Kempe, F. McSherry, K. David, and F. McSherry, "A decentralized algorithm for spectral analysis," *J. Comput. Syst. Sci.*, vol. 74, no. 1, pp. 70–83, 2008.

[104] K. Engan, S. O. Aase, and J. H. Husøy, "Multi-frame compression: Theory and design," *Signal Process.*, vol. 80, no. 10, pp. 2121–2140, 2000.

[105] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998.

[106] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," vol. 53, no. 12, pp. 4655–4666, 2007.

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

[108] A. Olshevsky and J. N. Tsitsiklis, “Convergence speed in distributed consensus and averaging,” *SIAM J. Controls Optimization*, vol. 48, no. 1, pp. 33–55, 2009.

[109] R. Olfati-Saber and R. M. Murray, “Consensus problems in networks of agents with switching topology and time-delays,” vol. 49, no. 9, pp. 1520–1533, 2004.

[110] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *J. Royal Stat. Soc. Series B (Methodological)*, pp. 267–288, 1996.

[111] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, “Least angle regression,” *Ann. Stat.*, vol. 32, no. 2, pp. 407–451, 2004.

[112] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright, “Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems,” *IEEE J. Sel. Topics Signal Process.*, vol. 1, no. 4, pp. 586–597, Dec. 2007.

[113] J. Ding, L. Chen, and Y. Gu, “Perturbation analysis of orthogonal matching pursuit,” *IEEE Trans. Signal Process.*, vol. 61, no. 2, pp. 398–410, 2013.

[114] J.-J. Fuchs, “On sparse representations in arbitrary redundant bases,” *IEEE Trans. Inform. Theory*, vol. 50, no. 6, pp. 1341–1344, Jun. 2004.

[115] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov chains and mixing times.* American Math. Soc., 2009.

[116] S. P. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Mixing times for random walks on geometric random graphs.” in *Proc. ALENEX/ANALCO.* SIAM, 2005, pp. 240–249.

[117] R. Rubinstein, M. Zibulevsky, and M. Elad, “Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit,” *CS Technion*, vol. 40, no. 8, pp. 1–15, 2008.

[118] T. Wu, A. D. Sarwate, and W. U. Bajwa, “Active dictionary learning for image representation,” in *Proc. SPIE Unmanned Syst. Technology XVII*, vol. 9468, Baltimore, MD, Apr. 2015.

[119] Z. Jiang, Z. Lin, and L. S. Davis, “Label consistent K-SVD: Learning a discriminative dictionary for recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2651–2664, 2013.

[120] M. Yang, L. Zhang, X. Feng, and D. Zhang, “Sparse representation based fisher discrimination dictionary learning for image classification,” *Int. J. Comput. Vision*, vol. 109, no. 3, pp. 209–232, 2014.

[121] S. Bahrampour, N. M. Nasrabadi, A. Ray, and W. K. Jenkins, “Multimodal task-driven dictionary learning for image classification,” *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 24–38, 2016.

[122] A. Koppel, G. Warnell, E. Stump, and A. Ribeiro, “D4L: Decentralized dynamic discriminative dictionary learning,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 2966–2973.

[123] Z. Shakeri, H. Raja, and W. U. Bajwa, “Dictionary learning based nonlinear classifier training from distributed data,” in *Proc. 2nd IEEE Global Conf. Signal Inform. Process., Symp. Netw. Theory*, Atlanta, GA, 2014, pp. 759–763.

[124] Y. LeCun and C. Cortes, “The MNIST database of handwritten digits,” http://yann.lecun.com/exdb/mnist/, 1998.

[125] Y. Chen, N. M. Nasrabadi, and T. D. Tran, “Hyperspectral image classification using dictionary-based sparse representation,” *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 10, pp. 3973–3985, 2011.

[126] G. W. Stewart, “Perturbation theory for the singular value decomposition,” *Technical Report.*, 1998.

[127] N. Mehta and A. G. Gray, “Sparsity-based generalization bounds for predictive sparse coding,” in *Proc. 30th Intl. Conf. Mach. Learn.*, Atlanta, GA, Jun. 2013, pp. 36–44.

[128] M. Stiefel, M. Leigsnering, A. Zoubir, F. Ahmad, and M. Amin, “Distributed greedy sparse recovery for through-the-wall radar imaging,” in *Proc. 31st IEEE Int. Review Progress Applied Comput. Electromagnetics*, 2015, pp. 1–2.

[129] M. G. Amin, Ed., *Through-the-Wall Radar Imaging.* Boca Raton, FL: CRC press, 2011.

[130] M. Amin and F. Ahmad, “Compressive sensing for through-the-wall radar imaging,” *J. Electronic Imag.*, vol. 22, no. 3, pp. 030 901–1–030 901–21,, 2013.

[131] S. Kidera, T. Sakamoto, and T. Sato, “Extended imaging algorithm based on aperture synthesis with double-scattered waves for UWB radars,” *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 12, pp. 5128–5139, Dec. 2011.

[132] P. Setlur, M. Amin, and F. Ahmad, “Multipath model and exploitation in through-the-wall and urban radar sensing,” *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 10, pp. 4021–4034, Oct. 2011.

[133] M. Amin, Ed., *Compressive Sensing for Urban Radar.* Boca Raton, FL: CRC Press, 2015.

[134] M. Leigsnering, F. Ahmad, M. G. Amin, and A. M. Zoubir, “Compressive sensing based specular multipath exploitation for through-the-wall radar imaging,” in *Proc. IEEE Int. Conference on Acoustics, Speech and Signal Process. (ICASSP)*, 2013, pp. 6004–6008.

[135] M. Leigsnering, F. Ahmad, M. Amin, and A. Zoubir, "Multipath exploitation in through-the-wall radar imaging using sparse reconstruction," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 50, no. 2, pp. 920–939, 2014.

[136] G. Gennarelli, I. Catapano, and F. Soldovieri, "RF/microwave imaging of sparse targets in urban areas," *IEEE Antennas Wireless Propag. Lett.*, vol. 12, pp. 643–646, 2013.

[137] M. Leigsnering, F. Ahmad, M. Amin, and A. Zoubir, "Compressive sensing-based multipath exploitation for stationary and moving indoor target localization," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 8, pp. 1469–1483, 2015.

[138] A. AlBeladi and A. Muqaibel, "Compressive sensing based joint wall position detection and multipath exploitation in through-the-wall radar imaging," in *Proc. 9th Int. Symp. Image Signal Process. and Analysis*, 2015, pp. 260–264.

[139] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Automat. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[140] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM J. Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.

[141] M. Ghassemi and A. D. Sarwate, "Distributed proportional stochastic coordinate descent with social sampling," in *53rd Annu. Allerton Conf. Commun., Control, and Comput.*, Sep. 2015, pp. 17–24.

[142] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Trans. Signal Process.*, vol. 60, no. 8, pp. 4289–4305, 2012.

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

[144] F. Sahin and A. Devasia, *Distributed particle swarm optimization for structural Bayesian network learning.* INTECH Open Access Publisher, 2007.

[145] S. Bouamama, "A new distributed particle swarm optimization algorithm for constraint reasoning," in *Knowledge-Based Intell. Inf. Eng. Syst.* Springer, 2010, pp. 312–321.

[146] J. Pugh and A. Martinoli, "Distributed scalable multi-robot learning using particle swarm optimization," *Swarm Intell.*, vol. 3, no. 3, pp. 203–222, 2009.

[147] A. Erdeljan, D. Capko, S. Vukmirovic, D. Bojanic, and V. Congradac, "Distributed PSO algorithm for data model partitioning in power distribution systems," *J. Appl. Res. Technol.*, vol. 12, no. 5, pp. 947–957, 2014.

[148] J. A. Tropp and A. C. Gilbert, “Signal recovery from random measurements via orthogonal matching pursuit,” *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, 2007.

[149] J. Nocedal and S. J. Wright, *Sequential quadratic programming.* Springer, 2006.

[150] W. W. Hager and H. Zhang, “A new active set algorithm for box constrained optimization,” *SIAM J. Opt.*, vol. 17, no. 2, pp. 526–557, 2006.