BRIDGE: Byzantine-resilient Decentralized Gradient Descent

Cheng Fang, Zhixiong Yang, and Waheed U. Bajwa

***Abstract*—Machine learning has begun to play a central role in many applications. A multitude of these applications typically also involve datasets that are distributed across multiple computing devices/machines due to either design constraints (e.g., multi-agent and Internet-of-Things systems) or computational/privacy reasons (e.g., large-scale machine learning on smartphone data). Such applications often require the learning tasks to be carried out in a decentralized fashion, in which there is no central server that is directly connected to all nodes. In real-world decentralized settings, nodes are prone to undetected failures due to malfunctioning equipment, cyberattacks, etc., which are likely to crash non-robust learning algorithms. The focus of this paper is on robustification of decentralized learning in the presence of nodes that have undergone Byzantine failures. The Byzantine failure model allows faulty nodes to arbitrarily deviate from their intended behaviors, thereby ensuring designs of the most robust of algorithms. But the study of Byzantine resilience within decentralized learning, in contrast to distributed learning, is still in its infancy. In particular, existing Byzantine-resilient decentralized learning methods either do not scale well to large-scale machine learning models, or they lack statistical convergence guarantees that help characterize their generalization errors. In this paper, a scalable, Byzantine-resilient decentralized machine learning framework termed Byzantine-resilient decentralized gradient descent (BRIDGE) is introduced. Algorithmic and statistical convergence guarantees for one variant of BRIDGE are also provided in the paper for both strongly convex problems and a class of nonconvex problems. In addition, large-scale decentralized learning experiments are used to establish that the BRIDGE framework is scalable and it delivers competitive results for Byzantine-resilient convex and nonconvex learning.**

I. Introduction

One of the fundamental tasks of *machine learning* (ML) is to learn a model using training data that minimizes the statistical risk [1]. A typical technique that accomplishes this task is *empirical risk minimization* (ERM) of a loss function [2]–[6]. Under the ERM framework, an ML model is learned by an optimization algorithm that tries to minimize the average loss with respect to the training data that are assumed available at a single location. In many recent applications of ML, however, training data tend to be geographically distributed; examples include the multi-agent and Internet-of-Things systems, smart grids, sensor networks, etc. In several other recent applications of ML, the training data cannot be gathered at a single machine due to either the massive scale of data and/or privacy concerns; examples in this case include the social network data, smartphone data, healthcare data, etc. The applications in both such cases require that the ML model be learned using training data that are distributed over a network. When the ML/optimization algorithm in such applications requires a central coordinating server connected to all the nodes in the network, the resulting framework is often referred to as *distributed learning* [7]. Practical constraints many times also require an application to accomplish the learning tasks without a central server [8], in which case the resulting framework is referred to as *decentralized learning*.

The focus of this paper is on decentralized learning, with a particular emphasis on characterizing the *sample complexity* of the decentralized learning algorithm—i.e., the rate, as a function of the number of training data samples, at which the ERM solution approaches the *Bayes optimal solution* in a decentralized setting [5], [8]. While decentralized learning has a rich history, a significant fraction of that work has focused on the faultless setting [9]–[13]. But real-world decentralized systems are bound to undergo failures because of malfunctioning equipment, cyberattacks, and so on [14]. And when the failures happen and go undetected, the learning algorithms designed for the faultless networks break down [7], [15]. Among the different types of failures in the network, the so-called *Byzantine failure* [16] is considered the most general, as it allows the faulty/compromised nodes to arbitrarily deviate from the agreed-upon protocol [14]. Byzantine failures are the hardest to safeguard against and can easily jeopardize the ability of the network to reach consensus [17], [18]. Moreover, it has been shown in [15] that a single Byzantine node with a simple strategy can lead to the failure of a decentralized learning algorithm. The overarching goal of this paper is to develop and (algorithmically and statistically) analyze an efficient decentralized learning algorithm that is provably resilient against Byzantine failures in decentralized settings with respect to both convex and nonconvex loss functions.

A. Relationship to prior works

Although the model of Byzantine failure was brought up decades ago, it has attracted the attention of ML researchers only very recently. Motivated by applications in large-scale machine learning [8], much of that work has focused solely

C. Fang (cf446@soe.rutgers.edu) and W. U. Bajwa (waheed.bajwa@rutgers.edu) are with the Department of Electrical and Computer Engineering, Rutgers University–New Brunswick, NJ 08854. Z. Yang (zhixiong.yang@bluedanube.com) completed this work as part of his PhD dissertation at Rutgers University; he is now a Machine Learning Systems Engineer at Blue Danube Systems.

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. The material includes additonal experimental results, and proofs of main lemmas and theorems. Contact waheed.bajwa@rutgers.edu for further questions about this work.

This work is supported in part by the National Science Foundation under awards CCF-1453073, CCF-1907658, and OAC-1940074, and by the Army Research Office under award W911NF2110301.

on the distributed learning setup such as the parameter–server setting [19] and the federated learning setting [20]. A necessarily incomplete list of these works, most of which have developed and analyzed Byzantine-resilient distributed learning approaches from the perspective of stochastic gradient descent, include [21]–[42]. Nonetheless, translating the algorithmic and analytical insights from the distributed learning setups to the decentralized ones, which lack central coordinating servers, is a nontrivial endeavor. As such, despite the plethora of work on Byzantine-resilient distributed learning, the problem of Byzantine-resilient decentralized learning—with the exception of a handful of works discussed in the following—largely remains unexplored in the literature.

In terms of decentralized learning in general, there are three broad classes of iterative algorithms that can be utilized for decentralized training purposes. The first one of these classes of algorithms corresponds to first-order methods such as the *distributed gradient descent* (DGD) and its (stochastic) variants [43]–[46]. The iterative methods in this class have low (local) computational complexity, which makes them particularly well suited for large-scale problems. The second class of algorithms involves the use of augmented Lagrangian-based methods [47]–[49], which require each node in the network to locally solve an optimization subproblem. The third class of algorithms includes second-order methods [50], [51], which typically have high computational and/or communications cost. Although the decentralized learning methods within these three classes of algorithms have their own sets of strengths and weaknesses, all of these traditional works assume faultless operations within the decentralized network.

Within the context of Byzantine failures in decentralized systems, some of the first works focused on the problem of Byzantine-resilient averaging consensus [52], [53]. These works were then leveraged to develop theory and algorithms for Byzantine-resilient decentralized learning for the case of scalar-valued models [15], [54]. But neither of these works are applicable to the general vector-valued ML framework being considered in this paper. In parallel, some researchers have also developed Byzantine-resilient decentralized learning methods for some specific vector-valued problems that include the decentralized support vector machine [55] and decentralized estimation [56]–[58].

Similar to the classical ML framework, however, there is a need to develop and algorithmically/statistically analyze Byzantine-resilient decentralized learning methods for vector-valued models for general—rather than specialized—loss functions, which can be broadly divided into two classes of convex and nonconvex loss functions. The first work in the literature that tackled this problem is [59], which developed a decentralized coordinate-descent-based learning algorithm termed ByRDiE and established its resilience to Byzantine failures in the case of a loss function that is given by the sum of a convex differentiable function and a strictly convex and smooth regularizer. The analysis in [59] also provided rates for algorithmic convergence as well as statistical convergence (i.e., sample complexity) of ByRDiE. One of the limitations of [59] is its exclusive focus on convex loss functions for the purposes of analysis. More importantly, however, the coordinate-descent nature of ByRDiE makes it slow and inefficient for learning of large-scale models. Let d denote the number of parameters in the ML model being trained (e.g., the number of weights in a deep neural network). One iteration of ByRDiE then requires updating the d coordinates of the model in d network-wide collaborative steps, each one of which requires a computation of the local d-dimensional gradient at each node in the network. In the case of large-scale models such as the deep neural networks with tens or hundreds of thousands of parameters, the local computation costs as well as the network-wide coordination and communications overhead of such an approach can be prohibitive for many applications. By contrast, since the algorithmic developments in this paper are based on the gradient-descent method, the resulting computational framework is highly efficient and scalable in a decentralized setting. And while the algorithmic and statistical convergence results derived in here match those for ByRDiE in the case of convex loss functions, the proposed framework is fundamentally different from ByRDiE and therefore necessitates its own theoretical analysis.

We conclude by noting that some additional works [60]–[63] relevant to the topic of Byzantine-resilient decentralized learning have appeared during the course of revising this paper, which are being discussed here for the sake of completeness. It is worth reminding the reader, however, that the work in this paper predates these recent efforts. Equally important, none of these works provide statistical convergence rates for the proposed methods. Additionally, the work in [60] only focuses on convex loss functions and it does not provide any convergence rates. Further, the ability of the proposed algorithm to defend against a large number of Byzantine nodes severely diminishes with an increase in the problem dimension. In contrast, the authors in [61] focus on Byzantine-resilient decentralized learning in the presence of non-uniformly distributed data and time-varying networks. The focus in this work is also only on convex loss functions and the performance of the proposed algorithm is worse than that of the approach advocated in this work for static networks and uniformly distributed data. Next, an algorithm termed MOZI is proposed in [62], with the focus once again being on convex loss functions. The resilience of MOZI, however, requires an aggressive two-step 'filtering' operation, which limits the maximum number of Byzantine nodes that can be handled by the algorithm. The analysis in [62] also makes the unrealistic assumption that the faulty nodes always send messages that are 'outliers' relative to those of the regular nodes. Finally, the only paper in the literature that has investigated Byzantine-resilient decentralized learning for nonconvex loss functions is [63]. The authors in this work have introduced three methods, among which the so-called ICwTM method is effectively a variant of our approach. The ICwTM algorithm, however, has at least twice the communications overhead of our approach, since it requires the neighbors to exchange both their local models and local gradients. In addition, [63] requires the nodes to have the same initialization and it does not bring out the dependence of the network topology on the learning problem.

Remark 1. While this paper was in review, a related work [68]

Algorithm	Nonconvex	Byzantine failures	Algorithmic convergence rate	Statistical convergence rate
ByRDiE [59]	×	√	√	√
Kuwarana ncharoen et. al [60]	×	√	×	×
Peng and Ling [61]	×	√	√	×
MOZI [62]	×	√	√	×
ICwTM [63]	√	√	√	×
DGD [45]	×	×	√	×
NEXT [64]	√	×	×	×
Nonconvex DGD [65]	√	×	√	×
D-GET [66]	√	×	√	√
GT-SARAH [67]	√	×	√	√
BRIDGE (This paper)	√	√	√	√

TABLE I: Comparison of BRIDGE with different vector-valued decentralized learning/optimization methods in the literature.

appeared on a preprint server; this recent work on Byzantine-resilient decentralized learning, in contrast to our paper, studies a more general class of nonconvex loss functions and also allows the distribution of training data at the regular nodes to be heterogeneous. However, in addition to the fact that [68] significantly postdates our work, the main result in [68] relies on clairvoyant knowledge of several network-wide parameters, including the subset of Byzantine nodes within the network, and also requires the maximum 'cumulative mixing weight' associated with the Byzantine nodes to be impractically small (e.g., even in the case of a *fully connected* network, the cumulative weight must be no greater than 9.76×10^{-5}).

B. Our contributions

One of the main contributions of this paper is the introduction of an efficient and scalable algorithmic framework for Byzantine-resilient decentralized learning. The proposed framework, termed ***B**yzantine-**r**esi**l**ient **d**ecentralized **g**radient **de**scent* (BRIDGE), overcomes the computational and communications overhead associated with the one-coordinate-at-a-time update pattern of ByRDiE through its use of the gradient descent-style updates. Specifically, the network nodes locally compute the d-dimensional gradient (and exchange the local d-dimensional model) only once in each iteration of the BRIDGE framework, as opposed to the d computations of the d-dimensional gradient in each iteration of ByRDiE. The BRIDGE framework therefore has significantly less local computational cost due to fewer gradient computations, and it also has smaller network-wide coordination and communications overhead due to fewer exchange of node-to-node messages. Note that BRIDGE is being referred to as a framework since it allows for multiple variants of a single algorithm depending on the choice of the *screening* method used within the algorithm for resilience purposes; see Section III for further details.

Another main contribution of this paper is analysis of one of the variants of BRIDGE, termed BRIDGE-T, for resilience against Byzantine failures in the network. The analysis enables us to provide both algorithmic convergence rates and statistical convergence rates for BRIDGE-T for certain classes of convex and nonconvex loss functions, with the rates derived for the convex setting matching those for ByRDiE [59]. The final main contribution of this paper is reporting of large-scale numerical results on the MNIST [69] and CIFAR-10 [70] datasets for both convex and nonconvex decentralized learning problems in the presence of Byzantine failures. The reported results, which include both *independent and identically distributed* (i.i.d.) and non-i.i.d. datasets within the network, highlight the benefits of the BRIDGE framework and validate our theoretical findings.

In summary, and to the best of our knowledge, BRIDGE is the first Byzantine-resilient decentralized learning algorithm that is scalable, has results for a class of nonconvex learning problems, and provides rates for both algorithmic and statistical convergence. We also refer the reader to Table I, which compares BRIDGE with recent works in both faultless and faulty vector-valued decentralized optimization/learning settings. Additional relevant works not appearing in this table include [15], [54], since they limit themselves to scalar-valued problems, and [68], since it substantially postdates our work. Further, [15], [54] neither study nonconvex loss functions nor derive the statistical convergence rates, while the main result in [68]—despite the generality of its problem setup—is significantly restrictive, as discussed in Remark 1.

C. Notation and organization

The following notation is used throughout the rest of the paper. We denote scalars with regular-faced lowercase and uppercase letters (e.g., a and A), vectors with bold-faced lowercase letters (e.g., $\mathbf{a}$), and matrices with bold-faced uppercase letters (e.g., $\mathbf{A}$). All vectors are taken to be column vectors, while $[\mathbf{a}]_k$ and $[\mathbf{A}]_{ij}$ denote the k-th element of vector $\mathbf{a}$ and the (i,j)-th element of matrix $\mathbf{A}$, respectively. We use $\|\mathbf{a}\|$ to denote the ℓ_2-norm of $\mathbf{a}$, $\mathbf{1}$ to denote the vector of all ones, and $\mathbf{I}$ to denote the identity matrix, while $(\cdot)^T$ denotes the transpose operation. Given two matrices $\mathbf{A}$ and $\mathbf{B}$, the notation $\mathbf{A} \succeq \mathbf{B}$ signifies that $\mathbf{A} - \mathbf{B}$ is a positive semidefinite matrix. We also use $\langle \mathbf{a}_1, \mathbf{a}_2 \rangle$ to denote the inner product between two vectors. For a given vector $\mathbf{a}$ and nonnegative constant γ, we denote the ℓ_2-ball of radius γ centered around $\mathbf{a}$ as $\mathbb{B}(\mathbf{a}, \gamma) := \{\mathbf{a}' : \|\mathbf{a} - \mathbf{a}'\| \leq \gamma\}$. Finally, given a set, $|\cdot|$ denotes its cardinality, while we use the notation $\mathcal{G}(\mathcal{J}, \mathcal{E})$ to denote a graph with the set of nodes $\mathcal{J}$ and edges $\mathcal{E}$.

The rest of this paper is organized as follows. Section II provides a mathematical formulation of the Byzantine-resilient decentralized learning problem, along with a formal definition of a Byzantine node and various assumptions on the loss function. Section III introduces the BRIDGE framework and discusses different variants of the BRIDGE algorithm. Section IV provides theoretical guarantees for the BRIDGE-T algorithm for certain classes of convex and nonconvex

loss functions, which include guarantees for network-wide consensus among the nonfaulty nodes and statistical convergence. Section V reports results corresponding to numerical experiments on the MNIST and CIFAR-10 datasets for both convex and nonconvex learning problems, establishing the usefulness of the BRIDGE framework for Byzantine-resilient decentralized learning. We conclude the paper in Section VI, while the **supplementary material** contains results of additional experiments and proofs of the lemmas and theorems.

II. Problem Formulation

A. Preliminaries

Let $(\mathbf{w}, \mathbf{z}) \mapsto f(\mathbf{w}, \mathbf{z})$ be a non-negative-valued (and possibly regularized) *loss function* that maps the tuple of a *model* $\mathbf{w}$ and a *data sample* $\mathbf{z}$ to the corresponding loss $f(\mathbf{w}, \mathbf{z})$. Without loss of much generality, we assume the model $\mathbf{w}$ in this paper to be a parametric one, i.e., $\mathbf{w} \in \mathbb{R}^d$ (e.g., d could be the number of parameters in a deep neural network). The data sample $\mathbf{z}$, on the other hand, corresponds to a random variable on some probability space $(\Omega, \mathcal{F}, \mathbb{P})$, i.e., $\mathbf{z}$ is $\mathcal{F}$-measurable and has been drawn from the sample space Ω according to the probability law $\mathbb{P}$. The holy grail in machine learning (ML) is to obtain an optimal model $\mathbf{w}^*$ that minimizes the expected loss, termed the *statistical risk* [5], [6], i.e.,

$$\mathbf{w}^* \in \arg\min_{\mathbf{w} \in \mathbb{R}^d} \mathbb{E}_{\mathbb{P}}[f(\mathbf{w}, \mathbf{z})]. \tag{1}$$

A model $\mathbf{w}^*$ that satisfies (1) is termed a *statistical risk minimizer* (also known as a *Bayes optimal model*). In the real world, however, one seldom has access to the distribution of $\mathbf{z}$, which precludes the use of the statistical risk $\mathbb{E}_{\mathbb{P}}[f(\mathbf{w}, \mathbf{z})]$ in any computations. Instead, a common approach utilized in ML is to leverage a collection $\mathcal{Z} := \{\mathbf{z}_n\}_{n=1}^N$ of data samples that have been drawn according to the law $\mathbb{P}$ and solve an empirical variant of (1) as follows [5], [6]:

$$\mathbf{w}^*_{\mathsf{ERM}} \in \arg\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{w}, \mathbf{z}_n). \tag{2}$$

This approach, which is termed as the *empirical risk minimization* (ERM), typically relies on an optimization algorithm to solve for $\mathbf{w}^*_{\mathsf{ERM}}$. The resulting solution $\widehat{\mathbf{w}}$, from the perspective of an ML practitioner, must satisfy two criteria: (i) it should have fast algorithmic convergence, measured in terms of the algorithmic convergence rate, to a fixed point (often taken to be a stationary point of (2) in centralized settings); and (ii) it should have fast statistical convergence, often specified in terms of the sample complexity (number of samples), to a statistical risk minimizer. Our focus in this paper, in contrast to several related prior works (cf. Table I), is on both the algorithmic and the statistical convergence of the ERM solution. The final set of results in this case rely on a number of assumptions on the loss function $f(\mathbf{w}, \mathbf{z})$, stated below.

Assumption 1 (Bounded and Lipschitz gradients)**.** The loss function $f(\mathbf{w}, \mathbf{z})$ is differentiable in the first argument $\mathbb{P}$-almost surely (a.s.) and the gradient of $f(\mathbf{w}, \mathbf{z})$ with respect to the first argument, denoted as $\nabla f(\mathbf{w}, \mathbf{z})$, is bounded and L'-Lipschitz a.s., i.e.,[1] $\forall \mathbf{w} \in \mathbb{R}^d, \|\nabla f(\mathbf{w}, \mathbf{z})\| \leq L$ a.s. and

$$\forall \mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^d,\ \|\nabla f(\mathbf{w}_1, \mathbf{z}) - \nabla f(\mathbf{w}_2, \mathbf{z})\| \leq L' \|\mathbf{w}_1 - \mathbf{w}_2\| \text{ a.s.}$$

In the literature, functions with L'-Lipschitz gradients are also referred to as L'-smooth functions. Assumption 1 implies the loss function is itself a.s. L-Lipschitz continuous [71], i.e., $\forall \mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^d,\ |f(\mathbf{w}_1, \mathbf{z}) - f(\mathbf{w}_2, \mathbf{z})| \leq L \|\mathbf{w}_1 - \mathbf{w}_2\|$ a.s.

Assumption 2 (Bounded training loss)**.** The loss function is a.s. bounded over the training samples, i.e., there exists a constant C such that $\sup_{\mathbf{w} \in \mathbb{R}^d, \mathbf{z} \in \mathcal{Z}} f(\mathbf{w}, \mathbf{z}) \leq C < \infty$ a.s.

The analysis carried out in this paper considers two different classes of loss functions, namely, convex functions and nonconvex functions. In the case of analysis for the convex loss functions, we make the following assumption.

Assumption 3 (Strong convexity)**.** The loss function $f(\mathbf{w}, \mathbf{z})$ is a.s. λ-strongly convex in the first argument, i.e.,

$$\forall \mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^d,\ f(\mathbf{w}_1, \mathbf{z}) \geq f(\mathbf{w}_2, \mathbf{z}) + \langle \nabla f(\mathbf{w}_2, \mathbf{z}), \mathbf{w}_1 - \mathbf{w}_2 \rangle + \frac{\lambda}{2} \|\mathbf{w}_1 - \mathbf{w}_2\|^2 \quad \text{a.s.}$$

Note that the Lipschitz gradients assumption can be relaxed to Lipschitz subgradients in the case of strongly convex loss functions. Some examples of loss functions that satisfy Assumptions 1 and 3 arise in ridge regression, elastic net regression, ℓ_2-regularized logistic regression, and ℓ_2-regularized training of support vector machines when the optimization variable $\mathbf{w}$ is constrained to belong to a bounded set in $\mathbb{R}^d$. Our discussion in the sequel shows that $\mathbf{w}$ indeed remains bounded for the algorithms in consideration, justifying the usage of Assumptions 1 and 3. And while Assumption 2, as currently stated, would not be satisfied for the aforementioned regularized problems, the analysis in the paper only requires boundedness of the data-dependent term(s) of the loss function over the finite set of training data. For the sake of compactness of notation, however, we refrain from expressing the loss function as the sum of two terms, with the implicit understanding that Assumption 2 is concerned only with the data-dependent component of $f(\cdot, \cdot)$. Finally, in contrast to Assumption 3, we make the following assumption in relation to analysis of the class of nonconvex loss functions.

Assumption 3′ (Local strong convexity)**.** The loss function $f(\mathbf{w}, \mathbf{z})$ is nonconvex and a.s. twice differentiable in the first argument. Next, let $\nabla^2 F(\mathbf{w})$ denote the Hessian of the statistical risk $F(\mathbf{w}) := \mathbb{E}_{\mathbb{P}}[f(\mathbf{w}, \mathbf{z})]$ and let $\mathcal{W}_s^*$ denote the set of all first-order stationary points of $F(\mathbf{w})$, i.e., $\mathcal{W}_s^* := \{\mathbf{w} \in \mathbb{R}^d : \nabla F(\mathbf{w}) = 0\}$. Then, for any $\mathbf{w}_s^* \in \mathcal{W}_s^*$, the statistical risk is *locally* λ-strongly convex in a sufficiently large neighborhood of $\mathbf{w}_s^*$, i.e., there exist positive constants λ and β such that $\forall \mathbf{w} \in \mathbb{B}(\mathbf{w}_s^*, \beta), \nabla^2 F(\mathbf{w}) \succeq \lambda \mathbf{I}$.

It is straightforward to see that the local strong convexity of the statistical risk does not imply the *global* strong convexity of either the statistical risk or the loss function. Assumptions

[1] Unless specified otherwise, all *almost sure* statements in the paper are to be understood with respect to the probability law $\mathbb{P}$.

similar to Assumption 3′ are nowadays routinely used for analysis of nonconvex optimization problems in machine learning; see, e.g., [72]–[75]. In particular, Assumption 3′ along with the proof techniques utilized in this work allow the theoretical results to be applicable to a broader class of functions in which the strong convexity is preserved only locally.

Remark 2. The convergence guarantees for nonconvex loss functions in this paper are *local* in the sense that they hold as long as the BRIDGE iterates are initialized within a sufficiently small neighborhood of a stationary point (cf. Section IV). Such local convergence guarantees are typical of many results in nonconvex optimization (see, e.g., [76]–[78]), but they do not imply that an iterative algorithm requires knowledge of the stationary point.

B. System model for decentralized learning

Consider a network of M nodes (devices, machines, etc.), expressed as a directed, static, and connected graph $\mathcal{G}(\mathcal{J},\mathcal{E})$ in which the set $\mathcal{J} := \{1,\ldots,M\}$ represents nodes in the network and the set of edges $\mathcal{E}$ represents communication links between the nodes. Specifically, $(j,i) \in \mathcal{E}$ if and only if node i can directly receive messages from node j and vice versa. We also define the neighborhood $\mathcal{N}_j$ of node j as the set of nodes that can send messages to it, i.e., $\mathcal{N}_j := \{i \in \mathcal{J} : (i,j) \in \mathcal{E}\}$. We assume each node j has access to a local training dataset $\mathcal{Z}_j := \{\mathbf{z}_{jn}\}_{n=1}^{|\mathcal{Z}_j|}$. For simplicity of exposition, we assume the cardinalities of the local training sets to be the same, as the generalization of our results to the case of $\mathcal{Z}_j$'s not being same sized is trivial. Collectively, therefore, the network has a total of MN samples that could be utilized for learning purposes.

In order to obtain an estimate of the statistical risk minimizer $\mathbf{w}^*$ (cf. (1)) in this decentralized setting, one would ideally like to solve the following ERM problem:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{MN}\sum_{j=1}^{M}\sum_{n=1}^{N} f(\mathbf{w},\mathbf{z}_{jn}) = \min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{M}\sum_{j=1}^{M} f_j(\mathbf{w}), \quad (3)$$

where we have used $f_j(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N} f(\mathbf{w},\mathbf{z}_{jn})$ to denote the *local* empirical risk associated with the j-th node. In particular, it is well known that the minimizer of (3) will statistically converge with high probability to $\mathbf{w}^*$ in the case of a strictly convex loss function [1]. The problem in (3), however, necessitates bringing together of the data at a single location; as such, it cannot be practically solved in its current form in the decentralized setting. Instead, we assume each node j maintains a local version $\mathbf{w}_j$ of the desired global model and collaborate among themselves to solve the following *decentralized* ERM problem:

$$\min_{\{\mathbf{w}_1,\ldots,\mathbf{w}_M\}} \frac{1}{M}\sum_{j=1}^{M} f_j(\mathbf{w}_j) \text{ subject to } \forall i,j,\ \mathbf{w}_i = \mathbf{w}_j. \quad (4)$$

Traditional decentralized learning algorithms proceed iteratively to solve this decentralized ERM problem [9]–[13], [47], [79]. This is typically accomplished through each node j engaging in two tasks during each iteration: update the local variable $\mathbf{w}_j$ according to some (local and data-dependent) rule $g_j(\cdot)$, and broadcast some summary of its local information to the nodes that have node j in their respective neighborhoods.

C. Byzantine-resilient decentralized learning

While decentralized learning is well understood in the case of faultless networks, the main assumption in this paper is that some of the network nodes can arbitrarily deviate from their intended behavior during the iterative process. Such deviations could be caused by malfunctioning equipment, cyberattacks, etc. We model the deviations of the faulty nodes as a *Byzantine failure*, which is formally defined as follows [14], [16].

Definition 1 (Byzantine node)**.** A node $j \in \mathcal{J}$ is said to have undergone a Byzantine failure if, during any iteration of decentralized learning, it either updates its local variable $\mathbf{w}_j$ using an update rule $\widetilde{g}_j(\cdot) \neq g_j(\cdot)$ or it broadcasts some information other than the intended summary of its local information to the nodes in its vicinity.

Throughout the remainder of this paper, we use $\mathcal{R} \subseteq \mathcal{J}$ and $\mathcal{B} \subset \mathcal{J}$ to denote the sets of nonfaulty and Byzantine nodes in the network, respectively. In addition, we use r to denote the cardinality of the set $\mathcal{R}$ and assume that the number of Byzantine nodes is upper bounded by an integer b. Thus, we have $0 \leq |\mathcal{B}| \leq b$ and $r \geq M - b$. In addition, without loss of generality, we label the nonfaulty nodes from 1 to r within our analysis, i.e., $\mathcal{R} := \{1,\ldots,r\}$.

Under this assumption of Byzantine failures in the network, it is straightforward to see that the decentralized ERM problem as stated in (4) cannot be solved. Rather, the best one could hope for is to solve an ERM problem that is restricted to the set of nonfaulty nodes, i.e.,

$$\min_{\{\mathbf{w}_j : j\in\mathcal{R}\}} \frac{1}{r}\sum_{j\in\mathcal{R}} f_j(\mathbf{w}_j) \text{ subject to } \forall i,j \in \mathcal{R},\ \mathbf{w}_i = \mathbf{w}_j, \quad (5)$$

except that the set $\mathcal{R}$ is unknown to an algorithm and therefore traditional decentralized learning algorithms cannot be utilized for this purpose. Consequently, the main goal in this paper is threefold: (i) Develop a decentralized learning algorithm that can provably solve some variant of the decentralized ERM problem (4); (ii) Establish that the resulting solution statistically converges to the statistical risk minimizer (Assumption 3) or a stationary point of the statistical risk (Assumption 3′); and (iii) Characterize the sample complexity of the solution, i.e., the statistical rate of convergence as a function of the number of samples, rN, associated with the nonfaulty nodes.

In order to accomplish the stated goal of this paper, we need to make one additional assumption concerning the topology of the network. This assumption, which is common in the literature on Byzantine resilience within decentralized networks [15], [59], requires definitions of the notions of a *source component* of a graph and a *reduced graph*, $\mathcal{G}_{\mathsf{red}}(b)$, of $\mathcal{G}$.

Definition 2 (Source component)**.** A source component of a graph is any subset of graph nodes such that each node in the subset has a directed path to every other node in the graph.

Definition 3 (Reduced graph)**.** A subgraph $\mathcal{G}_{\mathsf{red}}(b)$ of $\mathcal{G}$ is called a reduced graph with parameter b if it is generated from $\mathcal{G}$ by (i) removing all Byzantine nodes along with all their incoming and outgoing edges from $\mathcal{G}$, and (ii) additionally removing b incoming edges from each nonfaulty node.

Assumption 4 (Sufficient network connectivity)**.** The decentralized network is assumed to be sufficiently connected in the sense that all reduced graphs $\mathcal{G}_{\mathsf{red}}(b)$ of the underlying graph $\mathcal{G}(\mathcal{J}, \mathcal{E})$ contain at least one source component of cardinality greater than or equal to $(b+1)$.

We conclude by expanding further on Assumption 4, which concerns the redundancy of information flow within the network. In words, this assumption ensures that each nonfaulty node can continue to receive information from a few other nonfaulty nodes even after a certain number of edges have been removed from every nonfaulty node. And while efficient certification of this assumption remains an open problem, there is an understanding of the generation of graphs that satisfy this assumption [52]. In addition, we have empirically observed that Assumption 4 is often satisfied in Erdős–Rényi graphs as long as the degree of the least connected node is larger than $2b$. This is also the approach we take while generating graphs for our numerical experiments.

Remark 3. In the finite sample regime, in which each (nonfaulty) node has only a finite number of training data samples, the local empirical risk $f_j(\mathbf{w})$ at every node will be different due to the randomness of the data samples, regardless of whether the training data across the nodes are i.i.d. or non-i.i.d. While this makes the formulation in (5) similar to the one in [54] and [60] for scalar-valued and vector-valued Byzantine-resilient decentralized optimization, respectively, the fundamental difference between the statistical learning framework of this work and the optimization-only framework in [54], [60] is the intrinsic focus on sample complexity in statistical learning. In particular, the sample complexity results in Section IV-B also help characterize the gap between *local-only learning*, in which every regular node learns its own model using its own training data, and decentralized learning, in which the regular nodes collaborate to learn a common model.

III. Byzantine-resilient Decentralized Gradient Descent

In the faultless case, the decentralized ERM problem (4) can be solved, perhaps to one of its stationary points, using any one of the distributed/decentralized optimization methods in the literature [43]–[46], [64]–[67]. The prototypical *distributed gradient descent* (DGD) method [43] with decreasing step size, for instance, accomplishes this for (strongly) convex loss functions by letting each node in iteration $(t+1)$ update its local variable $\mathbf{w}_j(t)$ as

$$\mathbf{w}_j(t+1) = \sum_{i \in \mathcal{N}_j \cup \{j\}} a_{ji} \mathbf{w}_i(t) - \rho(t) \nabla f_j(\mathbf{w}_j(t)), \qquad (6)$$

where $0 \leq a_{ji} \leq 1$ is the weighting that node j applies to the local variable $\mathbf{w}_i(t)$ that it receives from node i, and $\{\rho(t)\}$ denotes a positive sequence of step sizes that satisfies $\rho(t+1) \leq \rho(t)$, $\rho(t) \overset{t}{\to} 0$, $\sum_{t=0}^{\infty} \rho(t) = \infty$, and $\sum_{t=0}^{\infty} \rho^2(t) < \infty$. One choice for such a sequence is $\rho(t) = \frac{1}{\lambda(t_0+t)}$ for some t_0, which ensures that a network-wide consensus is reached among all nodes, i.e., $\forall i, j, \mathbf{w}_i(t) \overset{t}{\to} \mathbf{w}_j(t)$, and all local variables converge to the decentralized (and thus the centralized) ERM solution.

Algorithm 1 The BRIDGE Framework

Input: Local datasets $\mathcal{Z}_j$, maximum number of Byzantine nodes b, step size sequence $\{\rho(t)\}_{t=0}^{\infty}$, and maximum number of iterations $t_{\max}$

1: **Initialize:** $t \leftarrow 0$ and $\forall j \in \mathcal{R}, \mathbf{w}_j(0)$
2: **for** $t = 0, 1, \ldots, t_{\max} - 1$ **do**
3: Broadcast $\mathbf{w}_j(t), \forall j \in \mathcal{R}$
4: Receive $\mathbf{w}_i(t)$ at each node $j \in \mathcal{R}$ from every $i \in \mathcal{N}_j \subset (\mathcal{R} \cup \mathcal{B})$
5: $\mathbf{y}_j(t) \leftarrow \mathsf{screen}(\{\mathbf{w}_i(t)\}_{i \in \mathcal{N}_j \cup \{j\}}), \forall j \in \mathcal{R}$
6: $\mathbf{w}_j(t+1) \leftarrow \mathbf{y}_j(t) - \rho(t) \nabla f_j(\mathbf{w}_j(t)), \forall j \in \mathcal{R}$
7: **end for**

Output: $\mathbf{w}_j(t_{\max}), \forall j \in \mathcal{R}$

Traditional distributed/decentralized optimization methods, however, fail to reach a stationary point of the decentralized ERM problem (4) (or its restricted variant (5)) in the presence of a single Byzantine failure in the network [27]–[29], [54], [80]–[82]. To overcome this shortcoming of the traditional approaches as well as improve on the limitations of existing works on Byzantine-resilient decentralized learning (cf. Sections I-A and I-B), we introduce an algorithmic framework termed **B**yzantine-**r**esi**l**ient **d**ecentralized **g**radient de**s**cent (BRIDGE).

The BRIDGE framework, which is listed in Algorithm 1, is a gradient descent-based approach whose main update step (Step 6 in Algorithm 1) is similar to the DGD update (6). The main difference between the BRIDGE framework and DGD is that each node $j \in \mathcal{R}$ in BRIDGE *screens* the incoming messages from its neighboring nodes for *potentially* malicious content (Step 5 in Algorithm 1) before updating its local variable $\mathbf{w}_j(t)$. Note, however, that BRIDGE does not permanently label any nodes as malicious, which also allows it to manage any transitory Byzantine failures in a graceful manner. While this makes BRIDGE similar to the ByRDiE algorithm [59], the fundamental advantage of BRIDGE over ByRDiE is its scalability that comes from the fact that it eschews the one-coordinate-at-a-time update of the local variables in ByRDiE in favor of one update of the entire vector $\mathbf{w}_j(t)$ in each iteration. In terms of other details, the BRIDGE framework is input with the maximum number of Byzantine nodes b that need to be tolerated, a decreasing step size sequence $\{\rho(t)\}$, and the maximum number of gradient descent iterations $t_{\max}$. Next, the local variable at each nonfaulty node j in the network is initialized at $\mathbf{w}_j(0)$. Afterward, within every iteration $(t+1)$ of the framework, each node $j \in \mathcal{R}$ broadcasts $\mathbf{w}_j(t)$ as well as receives $\mathbf{w}_i(t)$ from every node $i \in \mathcal{N}_j$. This is followed by every node $j \in \mathcal{R}$ screening the received $\mathbf{w}_i(t)$'s for any malicious information and then updating the local variable $\mathbf{w}_j(t)$.

In the following, we discuss four different variants of the *screening rule* (Step 5 in Algorithm 1), each one of which in turn gives rise to a different realization of the BRIDGE framework. The motivation for these screening rules comes from the literature on robust statistics [83], with all these rules appearing in some form within the literature on robust

Variant	Screening	Min. neighborhood size	Ave. computational complexity
BRIDGE-T	coordinate-wise	$2b+1$	$\mathcal{O}(nd)$, where $n := \max_j \lvert\mathcal{N}_j\rvert$
BRIDGE-M	coordinate-wise	1	$\mathcal{O}(nd)$
BRIDGE-K	vector	$b+3$	$\mathcal{O}(n^2 d)$
BRIDGE-B	vector + coordinate-wise	$\max(4b, 3b+2)+1$	$\mathcal{O}(n^2 d)$

TABLE II: Comparison between the four different variants of the BRIDGE framework.

averaging consensus [52], [53] and robust distributed learning [21], [27]–[29]. The challenge here of course, as discussed in Section I, is that these prior works on robust averaging consensus and distributed learning do not translate into equivalent results for Byzantine-resilient decentralized learning.

The **BRIDGE-T** variant of the BRIDGE framework uses the coordinate-wise trimmed mean as the screening rule. A similar screening principle has been utilized within distributed frameworks [28] and decentralized frameworks [15], [54], [59]. The coordinate-wise trimmed-mean screening within BRIDGE-T filters the b largest and the b smallest values in each coordinate of the local variables $\mathbf{w}_i(t)$ received from the neighborhood of node j and uses an average of the remaining values for the update of $\mathbf{w}_j(t)$. Specifically, for any iteration index t, BRIDGE-T finds the following three sets for each coordinate $k \in \{1, \ldots, d\}$ in parallel:

$$\overline{\mathcal{N}}_j^k(t) := \underset{\mathcal{X}:\mathcal{X}\subset\mathcal{N}_j, |\mathcal{X}|=b}{\arg\min} \sum_{i\in\mathcal{X}} [\mathbf{w}_i(t)]_k, \tag{7}$$

$$\underline{\mathcal{N}}_j^k(t) := \underset{\mathcal{X}:\mathcal{X}\subset\mathcal{N}_j, |\mathcal{X}|=b}{\arg\max} \sum_{i\in\mathcal{X}} [\mathbf{w}_i(t)]_k, \quad \text{and} \tag{8}$$

$$\mathcal{C}_j^k(t) := \mathcal{N}_j \setminus \left\{ \overline{\mathcal{N}}_j^k(t) \bigcup \underline{\mathcal{N}}_j^k(t) \right\}. \tag{9}$$

Afterward, the screening routine outputs a combined and filtered vector $\mathbf{y}_j(t)$ whose k-th element is given by

$$[\mathbf{y}_j(t)]_k = \frac{1}{|\mathcal{N}_j| - 2b + 1} \sum_{i\in\mathcal{C}_j^k(t) \cup\{j\}} [\mathbf{w}_i(t)]_k. \tag{10}$$

Notice that BRIDGE-T requires each node to have at least $2b+1$ neighbors. Also, note that the elements from different neighbors may survive the screening at different coordinates. Therefore, the average within BRIDGE-T is not taken over vectors; rather, the calculation of $\mathbf{y}_j(t)$ has to be carried out in a coordinate-wise manner.

The **BRIDGE-M** variant uses the coordinate-wise median as the screening rule, with a similar screening idea having been utilized within distributed frameworks [28]. Similar to BRIDGE-T, BRIDGE-M is also a coordinate-wise screening procedure in which the k-th element of the combined and filtered output $\mathbf{y}_j(t)$ takes the form

$$[\mathbf{y}_j(t)]_k = \mathsf{median}\Big(\{[\mathbf{w}_i(t)]_k\}_{i\in\mathcal{N}_j\cup\{j\}} \Big). \tag{11}$$

Notice that, unlike BRIDGE-T, the coordinate-wise median screening within BRIDGE-M neither requires an explicit knowledge of b nor does it impose an explicit constraint on the minimum number of neighbors of each node.

The **BRIDGE-K** variant uses the *Krum function* as the screening rule, which is similar to the screening principle that has been employed within distributed frameworks [21]. In terms of specifics, the Krum screening for the decentralized framework can be described as follows. Given $i, h \in \mathcal{N}_j \cup \{j\}$, write $h \sim i$ if $\mathbf{w}_h(t)$ is one of the $|\mathcal{N}_j| - b - 2$ vectors with the smallest Euclidean distance, expressed as $\|\mathbf{w}_h(t) - \mathbf{w}_i(t)\|$, from $\mathbf{w}_i(t)$. The Krum-based screening at node j then finds the neighbor index $i_j^*(t)$ as

$$i_j^*(t) = \underset{i\in\mathcal{N}_j}{\arg\min} \sum_{h\in\mathcal{N}_j\cup\{j\}: h\sim i} \|\mathbf{w}_h(t) - \mathbf{w}_i(t)\|, \tag{12}$$

and outputs the (combined and filtered) vector $\mathbf{y}_j(t)$ as $\mathbf{y}_j(t) = \mathbf{w}_{i_j^*}(t)$. Unlike BRIDGE-T and BRIDGE-M, BRIDGE-K utilizes vector-valued operations for screening, resulting in the surviving vector $\mathbf{y}_j(t)$ to be entirely from one neighbor of each node. The Krum screening rule within BRIDGE-K requires the neighborhood of every node to be larger than $b+2$. Note that since the Krum function requires the pairwise distances of all nodes within the neighborhood of every node, BRIDGE-K has high computational complexity in comparison to BRIDGE-T and BRIDGE-M.

Last but not least, the **BRIDGE-B** variant—inspired by a similar screening procedure within distributed frameworks [27]—uses a combination of Krum and coordinate-wise trimmed mean as the screening rule. Specifically, the screening within BRIDGE-B involves first selecting $|\mathcal{N}_j| - 2b$ neighbors of the j-th node by recursively finding an index $i_j^*(t) \in \mathcal{N}_j$ using (12), removing the selected node from the neighborhood, finding a new index from $\mathcal{N}_j \setminus \{i_j^*(t)\}$ using (12) again, and repeating this Krum-based process $|\mathcal{N}_j| - 2b$ times. Next, coordinate-wise trimmed mean-based screening, as described within BRIDGE-T, is applied to the received $\mathbf{w}_i(t)$'s of the $|\mathcal{N}_j| - 2b$ neighbors of node j that survive the first-stage Krum-based screening. Intuitively, the Krum-based vector screening first guarantees the surviving neighbors have the closest $\mathbf{w}_i(t)$'s in terms of the Euclidean distance, while coordinate-wise trimmed-mean screening afterward guarantees that each coordinate of the combined and filtered vector $\mathbf{y}_j(t)$ only includes the "inlier" values. The cost of this two-step screening procedure includes high computational complexity due to the use of the Krum function and the stricter requirement that the neighborhood of each node be larger than $\max(4b, 3b+2)$.

We conclude by providing a comparison in Table II between the four different variants of the BRIDGE framework. Note that both BRIDGE-T and BRIDGE-B reduce to DGD in the case of $b = 0$; this, however, is not the case for BRIDGE-M and BRIDGE-K. It is also worth noting that additional variants of BRIDGE can be obtained through further combinations of the different screening rules and/or incorporation of additional ideas from the literature on robust statistics. Nonetheless, each variant of the BRIDGE framework requires its own theoretical analysis for convergence guarantees. In this paper, we limit

ourselves to BRIDGE-T for this purpose and provide the corresponding guarantees in the next section.

IV. BRIDGE-T: Convergence Guarantees

In this section, we derive the algorithmic and statistical convergence guarantees for BRIDGE-T for both convex and nonconvex loss functions. While these results match those for ByRDiE for convex loss functions, they cannot be obtained directly from [59] since the filtered vector $\mathbf{y}_j(t)$ in BRIDGE-T corresponds to an iteration-dependent amalgamation of the iterates $\{\mathbf{w}_i(t)\}$ in the neighborhood of node j (cf. (10)). Our statistical convergence guarantees require the training data to be independent and identically distributed (i.i.d.) among all the nodes. Additionally, let $\mathbf{w}^*$ denote the unique global minimizer of the statistical risk in the case of the strongly convex loss function, while it denotes one of the first-order stationary points of the statistical risk in the case of the nonconvex loss function. We assume there exists a positive constant Γ such that for all $j \in \mathcal{R}, t \in \mathbb{R}^d$, $\|\mathbf{w}_j(t) - \mathbf{w}^*\| \leq \Gamma$. Note that this Γ can be arbitrary large and so the above assumption, which is again needed for derivation of the statistical rate of convergence, is a mild one (also, see Lemma 4 and its accompanying discussion). Broadly speaking, the analysis for our algorithmic and statistical convergence guarantees proceeds as follows.

First, we define a "consensus" vector $\mathbf{v}(t)$ and establish in Section IV-A that $\forall j \in \mathcal{R}, \mathbf{w}_j(t) \to \mathbf{v}(t)$ as $t \to \infty$ for an appropriate choice of the step-size sequence $\rho(t)$ that satisfies $\rho(t+1) \leq \rho(t)$, $\rho(t) \overset{t}{\to} 0$, $\sum_{t=0}^{\infty} \rho(t) = \infty$, and $\sum_{t=0}^{\infty} \rho^2(t) < \infty$. In particular, we work with the step size $\rho(t) = \frac{1}{\lambda(t_0+t)}, t_0 \geq \frac{L}{\lambda}$, where L is the Lipschitz constant of the loss function; see, e.g., Assumption 1. This consensus analysis relies on the condition that the $\mathbf{w}_j(0)$'s for all $j \in \mathcal{R}$ are initialized such that $\mathbf{v}(0) \in \mathbb{B}(\mathbf{w}^*, \Gamma)$. One such choice of initialization is initializing all $\mathbf{w}_j(0)$'s to be within $\mathbb{B}(\mathbf{w}^*, \Gamma)$. (Note that in terms of our analysis for the nonconvex loss function, which is provided in Section IV-B2, we will impose additional constraints on the initialization.). Afterward, we define two vectors $\mathbf{u}(t+1)$ and $\mathbf{x}(t+1)$ in Section IV-B that correspond to gradient descent updates of the consensus vector $\mathbf{v}(t)$ using, respectively, a convex combination of gradients of the local loss functions evaluated at $\mathbf{v}(t)$ and the gradient of the statistical risk evaluated at $\mathbf{v}(t)$. Finally, we define the following collection of distances: $a_1(t) := \|\mathbf{x}(t+1) - \mathbf{w}^*\|$, $a_2(t) := \|\mathbf{u}(t+1) - \mathbf{x}(t+1)\|$, $a_3(t) := \|\mathbf{v}(t+1) - \mathbf{u}(t+1)\|$, and $a_4(t+1) := \max_{j \in \mathcal{R}} \|\mathbf{w}_j(t+1) - \mathbf{v}(t+1)\|$. It is straightforward to see that $\|\mathbf{v}(t+1) - \mathbf{w}^*\| \leq a_1(t) + a_2(t) + a_3(t)$, and we establish in Section IV-B that $a_1(t) + a_2(t) + a_3(t) \overset{t}{\to} 0$ with high probability as well as $a_4(t+1) \overset{t}{\to} 0$ using Assumptions 1, 2, 3 and 4 in the convex setting and Assumptions 1, 2, 3′ and 4 in the nonconvex setting, thereby completing our proof of optimality for both convex and nonconvex loss functions.

A. Consensus guarantees

Let us pick an arbitrary index $k \in \{1, \ldots, d\}$ and define a vector $\mathbf{\Omega}(t) \in \mathbb{R}^r$ whose respective elements correspond to the k-th element of the iterate $\mathbf{w}_j(t)$ of nonfaulty nodes, i.e., $\forall j \in \mathcal{R}, [\mathbf{\Omega}(t)]_j = [\mathbf{w}_j(t)]_k$. Note that $\mathbf{\Omega}(t)$ as well as most of the variables in our discussion in this section depend on the index k; however, since k is arbitrary, we drop this explicit dependence on k in many instances for simplicity of notation.

We first show that the BRIDGE-T update at the nonfaulty nodes in the k-th coordinate can be expressed in a form that only involves the nonfaulty nodes. Specifically, we write

$$\mathbf{\Omega}(t+1) = \mathbf{Y}(t)\mathbf{\Omega}(t) - \rho(t)\mathbf{g}(t), \tag{13}$$

where the vector $\mathbf{g}(t)$ is defined as $[\mathbf{g}(t)]_j = [\nabla f_j(\mathbf{w}_j(t))]_k, j \in \mathcal{R}$. The formulation of the matrix $\mathbf{Y}(t)$ in this expression is as follows. Let $\mathcal{N}_j^r$ denote the nonfaulty nodes in the neighborhood of node j, i.e., $\mathcal{N}_j^r := \mathcal{R} \cap \mathcal{N}_j$. The set of Byzantine neighbors of node j can then be defined as $\mathcal{N}_j^b := \mathcal{N}_j \setminus \mathcal{N}_j^r$. To make the rest of the expressions clearer, we drop the iteration index t for the remainder of this discussion, even though the variables are still t-dependent. Let us now define the notation $b^* := |\mathcal{B}|$ as the actual (unknown) number of Byzantine nodes in the network, b_j^k as the number of Byzantine nodes remaining in the filtered set $\mathcal{C}_j^k$, and $q_j^k := b - b^* + b_j^k$. Since $b - b^* \geq 0$ by assumption and $b_j^k \geq 0$ by definition, notice that only one of two cases can happen during each iteration for every coordinate k: (i) $q_j^k > 0$ or (ii) $q_j^k = 0$. For case (i), we either have $b - b^* > 0$ or $b_j^k > 0$ or both. These conditions correspond to the scenario where the node j filters out more than b regular nodes from its neighborhood. Thus, we know that $\overline{\mathcal{N}}_j^k \cap \mathcal{N}_j^r \neq \emptyset$. Likewise, it follows that $\underline{\mathcal{N}}_j^k \cap \mathcal{N}_j^r \neq \emptyset$. Then $\exists m_j' \in \overline{\mathcal{N}}_j^k \cap \mathcal{N}_j^r$ and $m_j'' \in \underline{\mathcal{N}}_j^k \cap \mathcal{N}_j^r$ satisfying $[\mathbf{w}_{m_j'}]_k \leq [\mathbf{w}_i]_k \leq [\mathbf{w}_{m_j''}]_k$ for any $i \in \mathcal{C}_j^k$. Thus, for every $i \in \mathcal{C}_j^k \cap \mathcal{N}_j^b$, $\exists \theta_i \in (0,1)$ satisfying $[\mathbf{w}_i]_k = \theta_i [\mathbf{w}_{m_j'}]_k + (1-\theta_i)[\mathbf{w}_{m_j''}]_k$. Consequently, the elements of the matrix $\mathbf{Y}$ can be written as

$$[\mathbf{Y}]_{ji} = \begin{cases} \frac{1}{2(|\mathcal{N}_j| - 2b + 1)}, & i \in \mathcal{N}_j^r \cap \mathcal{C}_j^k, \\ \frac{1}{|\mathcal{N}_j| - 2b + 1}, & i = j, \\ \sum\limits_{i' \in \mathcal{N}_j^b \cap \mathcal{C}_j^k} \frac{\theta_{i'}}{q_j^k (|\mathcal{N}_j| - 2b + 1)} & \\ \quad + \sum\limits_{i' \in \mathcal{N}_j^r \cap \mathcal{C}_j^k} \frac{\theta_{i'}}{q_j^k (|\mathcal{N}_j| - 2b + 1)}, & i \in \overline{\mathcal{N}}_j^k \cap \mathcal{N}_j^r, \\ \sum\limits_{i' \in \mathcal{N}_j^b \cap \mathcal{C}_j^k} \frac{1 - \theta_{i'}}{q_j^k (|\mathcal{N}_j| - 2b + 1)} & \\ \quad + \sum\limits_{i' \in \mathcal{N}_j^r \cap \mathcal{C}_j^k} \frac{1 - \theta_{i'}}{q_j^k (|\mathcal{N}_j| - 2b + 1)}, & i \in \underline{\mathcal{N}}_j^k \cap \mathcal{N}_j^r, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

For case (ii), we must have that $b - b^* = 0$ and $b_j^k = 0$. Thus, all the filtered nodes in $\mathcal{C}_j^k$ would be regular nodes in this case. Therefore, we can describe $\mathbf{Y}$ in this case as

$$[\mathbf{Y}]_{ji} = \begin{cases} \frac{1}{|\mathcal{N}_j| - 2b + 1}, & i \in \{j\} \cup \mathcal{C}_j^k, \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

Combining the expressions of $\mathbf{Y}$ in the two cases above allows us to express the update in (13) exclusively in terms of information from the nonfaulty nodes.

Next, we define ψ to be the total number of reduced graphs that can be generated from $\mathcal{G}$, the parameter ν as $\nu := \psi r$,

and the maximum neighborhood size of the nonfaulty nodes as $\mathcal{N}_{\max} := \max_{j\in\mathcal{R}} |\mathcal{N}_j|$. Further, we define a transition matrix $\mathbf{\Phi}(t, t_0)$ from some index $t_0 \leq t$ to t, i.e.,

$$\mathbf{\Phi}(t, t_0) := \mathbf{Y}(t)\mathbf{Y}(t-1)\cdots\mathbf{Y}(t_0). \tag{16}$$

Then it follows from [84, Lemma 4] that if Assumption 4 is satisfied then

$$\lim_{t\to\infty} \mathbf{\Phi}(t, t_0) = \mathbf{1}\boldsymbol{\alpha}^T(t_0), \tag{17}$$

where the vector $\boldsymbol{\alpha}(t_0) \in \mathbb{R}^r$ satisfies $[\boldsymbol{\alpha}(t_0)]_j \geq 0$ and $\sum_{j=1}^r [\boldsymbol{\alpha}(t_0)]_j = 1$. In particular, we have [84, Theorem 3]

$$|[\mathbf{\Phi}(t, t_0)]_{ji} - [\boldsymbol{\alpha}(t_0)]_i| \leq \mu^{(\frac{t-t_0+1}{\nu})}, \tag{18}$$

where $\mu \in (0, 1)$ is defined as $\mu := 1 - \frac{1}{(2\mathcal{N}_{\max}-2b+1)^\nu}$.

Next, it follows from (13) and the definition of $\mathbf{\Phi}(t, t_0)$ that

$$\begin{aligned}
\mathbf{\Omega}(t) &= \mathbf{Y}(t-1)\mathbf{\Omega}(t-1) - \rho(t-1)\mathbf{g}(t-1) \\
&= \mathbf{Y}(t-1)\mathbf{Y}(t-2)\cdots\mathbf{Y}(0)\mathbf{\Omega}(0) \\
&\quad - \sum_{\tau=0}^{t-1} \mathbf{Y}(t-1)\mathbf{Y}(t-2)\cdots\mathbf{Y}(\tau+1)\rho(\tau)\mathbf{g}(\tau) \\
&= \mathbf{\Phi}(t-1, 0)\mathbf{\Omega}(0) - \sum_{\tau=0}^{t-1} \mathbf{\Phi}(t-1, \tau+1)\rho(\tau)\mathbf{g}(\tau).
\end{aligned} \tag{19}$$

Now, similar to [84, Convergence Analysis of Algorithm 1], suppose all nodes stop computing their local gradients after iteration t so that $\mathbf{g}(\tau) = 0$ when $\tau > t$. Note that this is without loss of generality when we let t approach infinity, as we recover BRIDGE-T in that case. Further, let $T \geq 0$ be an integer and define a vector $\bar{\mathbf{v}}(t)$ as follows:

$$\begin{aligned}
\bar{\mathbf{v}}(t) &= \lim_{T\to\infty} \mathbf{\Omega}(t+T+1) \\
&= \lim_{T\to\infty} \mathbf{\Phi}(t+T, 0)\mathbf{\Omega}(0) - \lim_{T\to\infty} \sum_{\tau=0}^{t+T} \mathbf{\Phi}(t+T, \tau+1)\rho(\tau)\mathbf{g}(\tau) \\
&= \mathbf{1}\boldsymbol{\alpha}^T(0)\mathbf{\Omega}(0) - \sum_{\tau=0}^{t-1} \mathbf{1}\boldsymbol{\alpha}^T(\tau+1)\rho(\tau)\mathbf{g}(\tau).
\end{aligned} \tag{20}$$

Notice that $\bar{\mathbf{v}}(t)$ is a constant vector and we define a scalar-valued sequence $v(t)$ to be any one of its elements. We now show that $[\mathbf{w}_j(t)]_k \xrightarrow{t} v(t)$. Indeed, we have from (20) that

$$\begin{aligned}
v(t) &= \sum_{i=1}^r [\boldsymbol{\alpha}(0)]_i [\mathbf{w}_i(0)]_k \\
&\quad - \sum_{\tau=0}^{t-1} \rho(\tau) \sum_{i=1}^r [\boldsymbol{\alpha}(\tau+1)]_i [\nabla f_i(\mathbf{w}_i(\tau))]_k.
\end{aligned} \tag{21}$$

Also recall from the update of $[\mathbf{w}_j(t)]_k$ that

$$\begin{aligned}
[\mathbf{w}_j(t)]_k &= \sum_{i=1}^r [\mathbf{\Phi}(t-1, 0)]_{ji} [\mathbf{w}_i(0)]_k \\
&\quad - \sum_{\tau=0}^{t-1} \rho(\tau) \sum_{i=1}^r [\mathbf{\Phi}(t-1, \tau+1)]_{ji} [\nabla f_i(\mathbf{w}_i(\tau))]_k.
\end{aligned}$$

From Assumption 1 and the initialization of $\mathbf{w}_j(t)$'s, there exist two scalars C_w and L such that $\forall j \in \mathcal{R}$, $|[\mathbf{w}_j(0)]_k| \leq C_w$ and $|[\nabla f_j(\mathbf{w}_j)]_k| \leq L$. Therefore, we have

$$\begin{aligned}
|[\mathbf{w}_j(t)]_k - v(t)| &\leq |\sum_{i=1}^r ([\mathbf{\Phi}(t-1, 0)]_{ji} - [\boldsymbol{\alpha}(0)]_i)[\mathbf{w}_i(0)]_k| \\
&\quad + |\sum_{\tau=0}^{t-1} \rho(\tau) \sum_{i=1}^r ([\mathbf{\Phi}(t-1, \tau+1)]_{ji} - [\boldsymbol{\alpha}(\tau+1)]_i)[\nabla f_i(\mathbf{w}_i(\tau))]_k| \\
&\leq rC_w\mu^{\frac{t}{\nu}} + rL\sum_{\tau=0}^{t} \rho(\tau)\mu^{\frac{t-\tau+1}{\nu}} \xrightarrow{t} 0.
\end{aligned} \tag{22}$$

Here, the fact that the second term in the second inequality of (22) converges to zero follows from our assumptions on the decreasing step size sequence along with [84, Lemma 6].

Finally, recall that the vector-valued iterate updates in BRIDGE-T can be thought of as individual updates of the d coordinates in parallel. Therefore, since the coordinate k was arbitrarily picked, we have proven that BRIDGE-T achieves consensus among the nonfaulty nodes for both convex and nonconvex loss functions, as summarized in the following.

Theorem 1. *Define a vector $\mathbf{v}(t) \in \mathbb{R}^d$ as one whose k-th entry $[\mathbf{v}(t)]_k$ is given by the right-hand-side of (21). If Assumptions 1 and 4 are satisfied, then the gap between $\mathbf{w}_j(t), \forall j \in \mathcal{R}$, and $\mathbf{v}(t)$ goes to 0 as $t \to \infty$, i.e.,*

$$\begin{aligned}
\lim_{t\to\infty} a_4(t) &= \lim_{t\to\infty} \max_{j\in\mathcal{R}} \|\mathbf{w}_j(t) - \mathbf{v}(t)\| \\
&\leq \lim_{t\to\infty} \left[\sqrt{d}rC_w\mu^{\frac{t}{\nu}} + \sqrt{d}rL\sum_{\tau=0}^{t} \rho(\tau)\mu^{\frac{t-\tau+1}{\nu}}\right] = 0.
\end{aligned} \tag{23}$$

We conclude our discussion with a couple of remarks. First, note that Theorem 1 has been obtained without needing Assumptions 2 and 3 / 3′. Thus, BRIDGE-T guarantees consensus among the nonfaulty nodes for both convex and nonconvex loss functions under a general set of assumptions. Second, notice that the second term in (23) is a sum of t terms and it converges to 0 at a slower rate than the first term. Among all the sub-terms in the sum of the second term, the last term $\rho(t)\mu^{\frac{1}{\nu}}$ is the one that converge to zero at the slowest rate. Thus, the rate at which BRIDGE-T achieves consensus is determined by this sub-term and is given by $\mathcal{O}(\sqrt{d}\rho(t))$. In particular, if we choose $\rho(t)$ to be $\mathcal{O}(1/t)$, the rate of consensus for BRIDGE-T is $\mathcal{O}(\sqrt{d}/t)$.

B. Statistical optimality guarantees

While Theorem 1 guarantees consensus among the nonfaulty nodes by providing an upper bound on the distance $a_4(t)$, this result alone cannot be used to characterize the gap between the iterates $\{\mathbf{w}_j(t)\}_{j\in\mathcal{R}}$ and the global minimizer (resp., first-order stationary point) $\mathbf{w}^*$ of the statistical risk for the convex loss function (resp., nonconvex loss function). We accomplish the goal of establishing the statistical optimality by providing bounds on the remaining three distances $a_1(t)$, $a_2(t)$, and $a_3(t)$ described in the beginning of the section.

We start with a bound on $a_3(t)$. To this end, let $\mathbf{v}(t)$ be as defined in Theorem 1 and notice from (20) that

$$\mathbf{v}(t+1) = \mathbf{v}(t) - \rho(t)\mathbf{g}_1(t), \tag{24}$$

where $\mathbf{g}_1(t)$ has the k-th entry defined in terms of gradients of the local loss functions evaluated at the local iterates as $[\mathbf{g}_1(t)]_k = \sum_{j=1}^r [\boldsymbol{\alpha}_k(t)]_j [\nabla f_j(\mathbf{w}_j(t))]_k$. Here, $\boldsymbol{\alpha}_k(t)$ is an element of the probability simplex associated with the k-th coordinate of the consensus vector $\mathbf{v}(t)$, as described earlier. Next, define another vector $\mathbf{g}_2(t)$ whose k-th entry is defined in terms of gradients of the local loss functions evaluated at the consensus vector as $[\mathbf{g}_2(t)]_k = \sum_{j=1}^r [\boldsymbol{\alpha}_k(t)]_j [\nabla f_j(\mathbf{v}(t))]_k$. Further, define a new sequence $\mathbf{u}(t+1)$ as

$$\mathbf{u}(t+1) = \mathbf{v}(t) - \rho(t)\mathbf{g}_2(t). \tag{25}$$

It then follows from (22), Assumption 1, and some algebraic manipulations that

$$\begin{aligned} a_3(t) &= \|\mathbf{v}(t+1) - \mathbf{u}(t+1)\| = \rho(t)\|\mathbf{g}_2(t) - \mathbf{g}_1(t)\| \\ &\leq \rho(t)L' \max_{j\in\mathcal{R}} \|\mathbf{v}(t) - \mathbf{w}_j(t)\| = \rho(t)L' a_4(t). \end{aligned} \tag{26}$$

We next turn our attention to a bound on $a_2(t)$, which is necessarily going to be probabilistic in nature because of its dependence on the training samples, and define a new sequence $\mathbf{x}(t)$ as

$$\mathbf{x}(t+1) = \mathbf{v}(t) - \rho(t)\nabla \mathbb{E}_{\mathbb{P}}[f(\mathbf{v}(t), \mathbf{z})]. \tag{27}$$

Trivially, we have

$$\begin{aligned} a_2(t) &= \|\mathbf{u}(t+1) - \mathbf{x}(t+1)\| \\ &= \rho(t)\|\mathbf{g}_2(t) - \mathbb{E}_{\mathbb{P}}[f(\mathbf{v}(t), \mathbf{z})]\|. \end{aligned} \tag{28}$$

The following lemma, whose proof is provided in Section S.II in supplementary material, now bounds $a_2(t)$ by establishing that $\mathbf{g}_2(t)$ converges in probability to $\mathbb{E}_{\mathbb{P}}[f(\mathbf{v}(t), \mathbf{z})]$.

Lemma 1. *Suppose Assumptions 1 and 2 are satisfied and the training data are i.i.d. Then, fixing any $\delta \in (0,1)$, we have with probability at least $1-\delta$ that*

$$\begin{aligned} a_2(t) &\leq \rho(t) \sup_t \|\mathbf{g}_2(t) - \mathbb{E}_{\mathbb{P}}[f(\mathbf{v}(t), \mathbf{z})]\| \\ &= \mathcal{O}\left(\sqrt{\frac{d\|\boldsymbol{\alpha}_m\|^2 \log\frac{2}{\delta}}{N}}\right)\rho(t), \end{aligned} \tag{29}$$

where the vector $\boldsymbol{\alpha}_m \in \mathbb{R}^r$ is a problem-dependent (unknown) vector defined in Section S.II in supplementary material and satisfies $[\boldsymbol{\alpha}_m]_j \geq 0$ and $\sum_{j=1}^r [\boldsymbol{\alpha}_m]_j = 1$.

Notice that while the bounds for $a_2(t)$, $a_3(t)$, and $a_4(t)$ have been obtained for both the convex and nonconvex loss functions in the same manner, the bound for $a_1(t) = \|\mathbf{x}(t+1) - \mathbf{w}^*\|$ in the nonconvex setting does require the aid of Assumption 3′, as opposed to Assumption 3 in the convex setting. This leads to separate proofs of the final statistical optimality results under Assumption 3 and Assumption 3′.

1) Statistical optimality for the convex case: Notice that $\mathbf{x}(t+1)$ is obtained from $\mathbf{v}(t)$ by taking a regular gradient descent step, with step size $\rho(t)$, with respect to the gradient $\mathbb{E}_{\mathbb{P}}[f(\mathbf{v}(t), \mathbf{z})]$ of the statistical risk. Under Assumptions 1 and 3, therefore, it follows from our understanding of the behavior of gradient descent iterations that [85, Chapter 2.1.5]

$$\begin{aligned} a_1(t) = \|\mathbf{x}(t+1) - \mathbf{w}^*\| &\leq (1 - L'\rho(t))\|\mathbf{v}(t) - \mathbf{w}^*\| \\ &\leq (1 - \lambda\rho(t))\|\mathbf{v}(t) - \mathbf{w}^*\|, \end{aligned} \tag{30}$$

where the last inequality holds because of the fact that $\lambda \leq L'$. We then have the following bound:

$$\|\mathbf{v}(t+1) - \mathbf{w}^*\| \leq (1 - \lambda\rho(t))\|\mathbf{v}(t) - \mathbf{w}^*\| + a_2(t) + a_3(t). \tag{31}$$

Notice that (31) only provides the relationship between steps t and $t+1$. In order to bound the distance $\|\mathbf{v}(t+1) - \mathbf{w}^*\|$ in terms of the initial distance $\|\mathbf{v}(0) - \mathbf{w}^*\|$, we can recursively make use of (31) to arrive at the following lemma.

Lemma 2. *Suppose Assumptions 1, 2, 3, and 4 are satisfied and the training data are i.i.d. Then fixing any $\delta \in (0,1)$, an upper bound on $\|\mathbf{v}(t+1) - \mathbf{w}^*\|$ can be derived with probability at least $1-\delta$ as*

$$\begin{aligned} \|\mathbf{v}(t+1) - \mathbf{w}^*\| &\leq \frac{t_0}{t+t_0}C_1 + \frac{C_2(N)}{\lambda} + \frac{C_3}{t+t_0} \\ &+ C_4\frac{1}{t+t_0}\left(\frac{1}{t_0} + \frac{1}{1+t_0} + \frac{1}{2+t_0} + \cdots + \frac{1}{t+t_0}\right), \end{aligned} \tag{32}$$

where $C_1 = \|\mathbf{v}(0) - \mathbf{w}^*\|$, $C_2(N) = \mathcal{O}\left(\sqrt{\frac{d\|\boldsymbol{\alpha}_m\|^2 \log\frac{2}{\delta}}{N}}\right)$, $C_3 = \frac{\sqrt{d}}{L}' r C_w \lambda\left(1 - \mu^{\frac{1}{\nu}}\right)$ and $C_4 = \frac{\sqrt{d}LL' r \mu^{\frac{1}{\nu}}}{t_0\lambda^2\left(1-\mu^{\frac{1}{\nu}}\right)}$.

Proof of Lemma 2 is provided in Section S.III in supplementary material. Lemma 2 establishes that $\|\mathbf{v}(t+1) - \mathbf{w}^*\|$ can be upper bounded by a sum of terms that can be made arbitrarily small for sufficiently large t and N. Since $\max_{j\in\mathcal{R}} \|\mathbf{w}_j(t) - \mathbf{v}(t)\|$ can also be made arbitrarily small when t is sufficiently large, we can therefore bound $\|\mathbf{w}_j(t+1) - \mathbf{w}^*\|, j \in \mathcal{R}$, using the bounds on $\|\mathbf{v}(t+1) - \mathbf{w}^*\|$ and $\max_{j\in\mathcal{R}} \|\mathbf{w}_j(t+1) - \mathbf{v}(t+1)\|$ to arrive at the following lemma.

Lemma 3. *Suppose Assumptions 1, 2, 3, and 4 are satisfied and the training data are i.i.d. Then fixing any $\delta \in (0,1)$ and any $\epsilon > \frac{C_2(N)}{\lambda} > 0$, we can always find a t_1 such that for all $t \geq t_1$ and $j \in \mathcal{R}$, with probability at least $1-\delta$, $\|\mathbf{w}_j(t+1) - \mathbf{w}^*\| \leq \epsilon$.*

Proof of Lemma 3 is provided in Section S.IV in supplementary material. Notice that given a sufficiently large N, $C_2(N)$ is arbitrarily small, which means that the iterates of non-Byzantine nodes can be made arbitrarily close to $\mathbf{w}^*$. We are now ready to state the main result concerning the statistical convergence of BRIDGE-T at the nonfaulty nodes to the global statistical risk minimizer $\mathbf{w}^*$.

Theorem 2. *Suppose Assumptions 1, 2, 3, and 4 are satisfied and the training data are i.i.d. Then the iterates of BRIDGE-T converge sublinearly in t to the minimum of the global statistical risk at each nonfaulty node. In particular, given any $\epsilon > \frac{\epsilon''}{\lambda} > 0$, $\forall j \in \mathcal{R}$, with probability at least $1-\delta$ and for large enough t,*

$$\|\mathbf{w}_j(t+1) - \mathbf{w}^*\| \leq \epsilon, \tag{33}$$

where $\delta = 2\exp\left(-\frac{4rN\epsilon''^2}{16L^2 rd\|\boldsymbol{\alpha}_m\|^2 + \epsilon''^2} + r\log\left(\frac{12L\sqrt{rd}}{\epsilon''}\right) + d\log\left(\frac{12L'\beta\sqrt{d}}{\epsilon''}\right)\right)$ and $\epsilon'' = C_2(N) = \mathcal{O}\left(\sqrt{\frac{d\|\boldsymbol{\alpha}_m\|^2 \log\frac{2}{\delta}}{N}}\right)$.

Proof of Theorem 2 is provided in Section S.V in supplementary material. Note that when $N \to \infty$ and when $\rho(t)$ is chosen as a $\mathcal{O}(1/t)$ sequence, (33) leads to a sublinear convergence rate, as shown in Section S.V in supplementary material. Thus, both the algorithmic and statistical convergence rates derived for BRIDGE-T match the existing Byzantine-resilient rates in the decentralized setting [59]. In terms of the statistical guarantees, recall that when there is no failure in the network, the non-resilient gradient descent algorithms such as DGD usually have the statistical learning rate as $\mathcal{O}\big(\sqrt{1/MN}\big)$, while if each node runs centralized algorithm with the given N samples, the learning rate is given as $\mathcal{O}\big(\sqrt{1/N}\big)$. BRIDGE-T achieves the statistical learning rate of $\mathcal{O}\big(\sqrt{\|\boldsymbol{\alpha}_m\|^2/N}\big)$, which lies between the rate of centralized learning and DGD. In particular, compared to local learning, BRIDGE-T reduces the sample complexity by a factor of $\|\boldsymbol{\alpha}_m\|^2$ for each node by cooperating over a network, but it cannot approach the fault-free rate. This shows the trade-off between sample complexity and robustness.

2) Statistical optimality for the nonconvex case: A general challenge for optimization methods in the nonconvex setting is the presence of multiple stationary points within the landscape of the loss function. Distributed frameworks in general and potential Byzantine failures within the network in particular make this an even more challenging problem. We overcome this challenge by making use of Assumption 3′ (local strong convexity) and aiming for local convergence guarantees.

In terms of specifics, recall the positive constant β from Assumption 3′ that describes the region of local strong convexity around a stationary point $\mathbf{w}^*$, let $\beta_1 \leq \beta$ be another positive constant that will be defined shortly, and pick any $\beta_0 \in (0, \beta - \beta_1]$. Then our local convergence guarantees are based on the assumption that $\forall j \in \mathcal{R}, \mathbf{w}_j(0)$'s are initialized such that $\mathbf{v}(0) \in \mathbb{B}(\mathbf{w}^*, \beta_0)$, with one such choice of initialization being that the $\mathbf{w}_j(0)$'s at the nonfaulty nodes are initialized within $\mathbb{B}(\mathbf{w}^*, \beta_0)$. In particular, assuming $\beta \geq \Gamma$ for Γ defined in the beginning of Section IV, we obtain the following lemma that establishes the boundedness of the iterates $\mathbf{w}_j(t)$ for any $j \in \mathcal{R}$ and $t \in \mathbb{R}$, characterizes the relationship between β, β_0, and β_1, and helps us understand how large β need be as a function of the different parameters.

Lemma 4. *Suppose Assumptions 1, 2, 3′, and 4 are satisfied and the training data are i.i.d. Then with the initialization described above for $\beta \geq \max\{\Gamma, \beta_1\}$ and β_1 defined as $\beta_1 := \frac{C_2(N)}{\lambda} + \frac{C_3}{t_0} + \frac{C_4}{t_0^2} + C_5$, the $\mathbf{w}_j(t)$'s will never escape from $\mathbb{B}(\mathbf{w}^*, \beta)$ for all $j \in \mathcal{R}, t \in \mathbb{R}$. Here, the constants C_2, C_3, and C_4 are as defined in Lemma 2, while the constant $C_5 := \sqrt{d} r C_w \mu^{\frac{1}{\nu}} + \sqrt{d} r L \frac{1}{1-\mu^{\frac{1}{\nu}}} \left[\frac{1}{\lambda t_0}\mu^{\frac{1}{\nu}} + \frac{1}{\lambda(t_0+1)}\right]$.*

Lemma 4 is proved in Section S.VI in supplementary material. In terms of the implications of this lemma, it first and foremost helps justify the assumption of bounded iterates of the nonfaulty nodes stated in the beginning of Section IV. More importantly, however, notice that the constraint $\beta \geq \Gamma$ and the assumption that $\forall j \in \mathcal{R}, t \in \mathbb{R}^d$, $\|\mathbf{w}_j(t) - \mathbf{w}^*\| \leq \Gamma$ have the potential to make Assumption 3′ meaningless for large-enough Γ. But Lemma 4 effectively helps us characterize the extent of Γ, i.e., choosing Γ to be $C_1 + \frac{C_2(N)}{\lambda} + \frac{C_3}{t_0} + \frac{C_4}{t_0^2} + C_5$ is sufficient to guarantee that all iterates of the nonfaulty nodes remain within $\mathbb{B}(\mathbf{w}^*, \Gamma) \subseteq \mathbb{B}(\mathbf{w}^*, \beta)$.

We conclude with our main result for the nonconvex case, which mirrors that for the convex loss functions.

Theorem 3. *Suppose Assumptions 1, 2, 3′ and 4 are satisfied and the training data are i.i.d. Then with the earlier described initialization within $\mathbb{B}(\mathbf{w}^*, \beta_0)$, the iterates of BRIDGE-T converge sublinearly in t to the stationary point $\mathbf{w}^*$ of the statistical risk at each nonfaulty node. In particular, given any $\epsilon > \frac{\epsilon''}{\lambda} > 0$, $\forall j \in \mathcal{R}$, with probability at least $1 - \delta$ and for large enough t,*

$$\|\mathbf{w}_j(t+1) - \mathbf{w}^*\| \leq \epsilon, \tag{34}$$

where $\delta = 2\exp\Big(-\frac{4rN\epsilon''^2}{16L^2rd\|\boldsymbol{\alpha}_m\|^2+\epsilon''^2} + r\log\Big(\frac{12L\sqrt{rd}}{\epsilon''}\Big) + d\log\Big(\frac{12L'\beta\sqrt{d}}{\epsilon''}\Big)\Big)$ and $\epsilon'' = C_2(N) = \mathcal{O}\Big(\sqrt{\frac{d\|\boldsymbol{\alpha}_m\|^2 \log\frac{2}{\delta}}{N}}\Big)$.

Proof of Theorem 3 is provided in Section S.VI-A in supplementary material. It can be seen from Section S.VI-A that the proof in the convex setting maps to the nonconvex one without much additional work. The reason for this is the new proof technique being used in the convex case, instead of the one utilized in [86], which guarantees BRIDGE-T converges to a local stationary point for the nonconvex functions described in Assumption 3′.

Remark 4. The convergence rates derived for BRIDGE-T are a function of the dimension d. Such dimension dependence is typical of many results in (centralized) statistical learning theory [87], [88]. While there is an ongoing effort to obtain dimension-independent rates in statistical learning [88], [89], we leave an investigation of this within the context of Byzantine-resilient decentralized learning for future work.

V. Numerical Results

The numerical experiments are separated into three parts. In the first part, we run experiments on the MNIST and CIFAR-10 datasets using a linear classifier with squared hinge loss, which is a case that fully satisfies all our assumptions for the theoretical guarantees in the convex setting. In the second part, we run experiments on the MNIST and CIFAR-10 datasets using a convolutional neural network. By showing that BRIDGE works in this general nonconvex loss function setting, we establish that BRIDGE indeed works for loss functions that satisfy Assumption 3′. In the third part, we run experiments on the MNIST dataset with non-i.i.d. distributions of data across the agents. The purpose of all the experiments is to provide numerical validation of our theoretical results and address the usefulness of our Byzantine-resilient technique on a broad scope (convex, nonconvex, non-i.i.d. data) of machine learning problems.

A. Linear classifier on MNIST and CIFAR-10

The first set of experiments is performed to demonstrate two facts: BRIDGE can maintain good performance under Byzantine attacks while classic decentralized learning methods

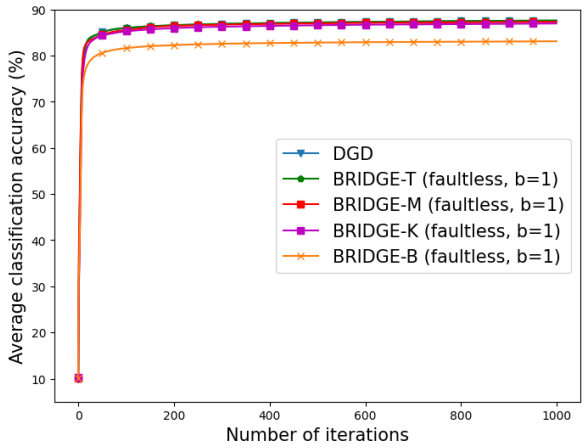

Fig. 1: Comparison between DGD and BRIDGE-T, -M, -K, -B in the faultless setting for a convex loss function and the MNIST dataset, where b is set to be $b = 1$ for BRIDGE.

fail; and compared to an existing Byzantine-resilient method, ByRDiE [59], BRIDGE is more efficient in terms of communications cost. We choose one of the most well-understood machine learning tools, the linear classifier with squared hinge loss, to learn the model for this purpose. Note that by showing BRIDGE works in this strictly convex and Lispchitz loss function setting means BRIDGE also works for strongly convex loss functions with bounded Lipschitz gradients.

The MNIST dataset is a set of 60,000 training images and 10,000 test images of handwritten digits from '0' to '9'. Each image is converted to a 784-dimensional vector and we distribute 60,000 images equally among 50 nodes. The CIFAR-10 dataset is a set of 50,000 training images and 10,000 test images of 10 different classes. Each image is converted to a 3072-dimensional vector and we distribute 50,000 images equally among 50 nodes. Then, unless stated otherwise, we connect each pair of nodes with probability $p = 0.5$. Some of the nodes are randomly picked to be Byzantine nodes, which broadcast random vectors to all their neighbors during each iteration. The parameter b for BRIDGE is set to be $b = 1$ in the faultless setting, while it is set to be equal to $|\mathcal{B}|$ in the faulty setting. Once a random network is generated and the Byzantine nodes are randomly placed, we check for each variant of BRIDGE whether the minimum neighborhood-size condition listed in Table II for its execution is satisfied before running that variant. The classifiers are trained using the "one vs all" strategy. We run five sets of experiments, with the first four on the MNIST dataset and the last one on the CIFAR-10 dataset: (i) classic distributed gradient descent (DGD) and BRIDGE-T, -M, -K, -B with no Byzantine nodes; (ii) classic DGD and BRIDGE-T, -M, -K, -B with 2 and 4 Byzantine nodes; (iii) BRIDGE-T, -M, and -K with 6, 12, 18, and 24 Byzantine nodes and varying probabilities of connection ($p = 0.5, 0.75$, and 1); (iv) ByRDiE and BRIDGE-T with 2 Byzantine nodes; and (v) BRIDGE-T, -M, and -K with 0, 2, 4, and 6 Byzantine nodes. The performance is evaluated by two metrics: classification accuracy on the 10,000 test images and whether consensus is achieved. When comparing ByRDiE and BRIDGE, we compare the accuracy with respect to the number of communication iterations, which is defined as the number of scalar-valued information exchanged among the neighboring nodes.

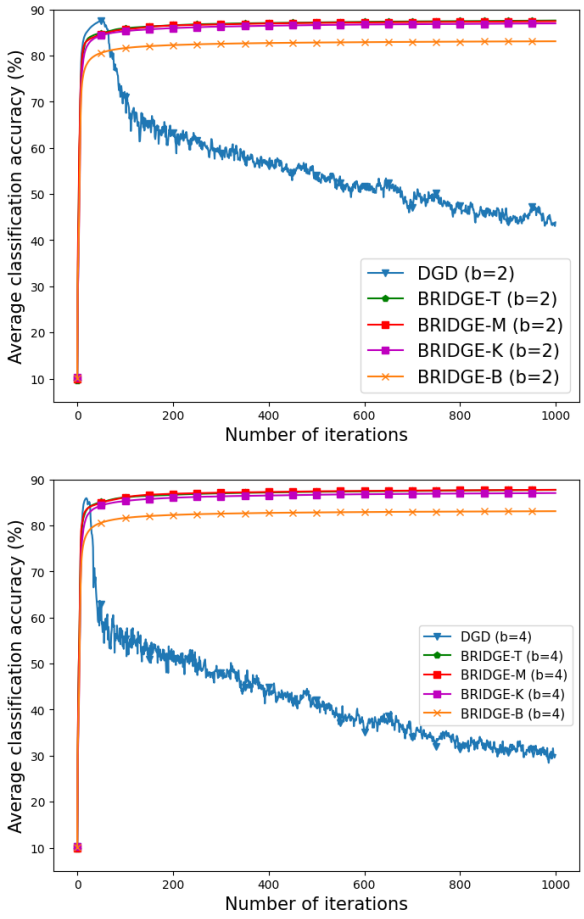

Fig. 2: Comparison between DGD and BRIDGE-T, M, K, B with two and four Byzantine nodes for a convex loss function with the MNIST dataset.

As we can see from Figure 1, despite using $b = 1$, the performances of all BRIDGE methods except BRIDGE-B are as good as DGD in the faultless setting with $\sim 88\%$ average accuracy, while BRIDGE-B performs slightly worse at 83% average accuracy (*final accuracy:* DGD $= 87.8\%$, BRIDGE-T $= 87.6\%$, -M $= 87.3\%$; -K $= 86.9\%$, and -B $= 83.1\%$). Note that these accuracy figures match the state-of-the-art results for the MNIST dataset using a linear classifier [69]. We also attribute the superiority of BRIDGE-T over -M, -K, and -B to its ability to retain information from a wider set of neighbors after the screening in each iteration. Next, we conclude from Figure 2 that DGD fails in the faulty setting when $|\mathcal{B}| = 2$ and produces an even worse accuracy in the faulty setting when $|\mathcal{B}| = 4$. However, BRIDGE-T, -M, -K and -B with $b = |\mathcal{B}|$ are able to learn relatively good models in these faulty settings.

The next set of results in Figure 3 highlights the robustness of the BRIDGE framework to a larger number of Byzantine nodes for varying levels of network connectivity that range from $p = 0.5$ to $p = 1$. We exclude BRIDGE-B in this figure since it fails to run for a majority of the randomly generated networks because of the stringent minimum neighborhood size

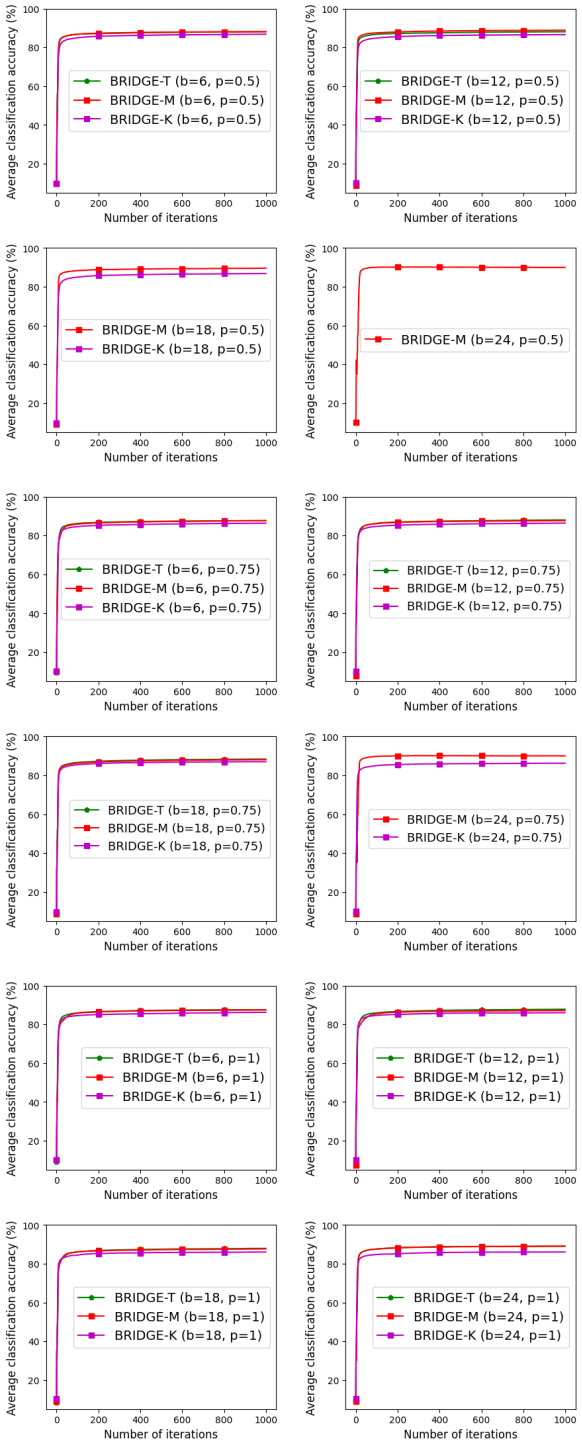

Fig. 3: Comparison between BRIDGE-T, -M, and -K for different numbers of Byzantine nodes and varying levels of network connectivity (convex loss and MNIST dataset).

condition (cf. Table II). The results in this figure reaffirm our findings from Figure 2 that the BRIDGE framework is extremely resilient to Byzantine attacks in the network. In particular, we see that BRIDGE-T (when it satisfies the minimum neighborhood size condition) and -M perform very similarly in the face of a large number of Byzantine nodes in the network, while BRIDGE-K is a close third in performance. Another observation from this figure is the robustness of

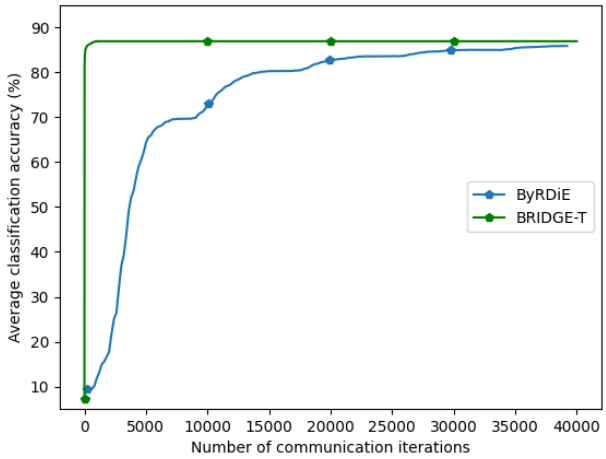

Fig. 4: Comparing BRIDGE-T with ByRDiE in the presence of two Byzantine nodes (convex loss and MNIST dataset).

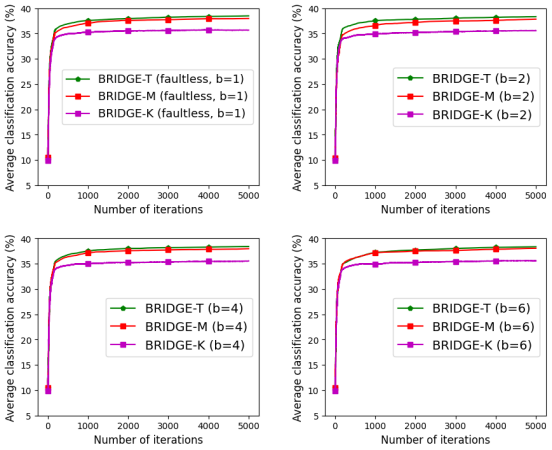

Fig. 5: Performance of BRIDGE-T, -M, and -K with zero, two, four, and six Byzantine nodes for a convex loss function with the CIFAR-10 dataset.

BRIDGE-M for loosely connected Erdős–Rényi networks and a large number of Byzantine attacks; indeed, BRIDGE-M is the only variant that can be run in the case of $b = 24$ and $p = 0.5$ since it always satisfies the minimum neighborhood size condition even when close to 50% of the nodes in the network are Byzantine. In contrast, BRIDGE-T could not be run when $b \geq 18$ (resp., $b = 24$) and $p = 0.5$ (resp., $p = 0.75$), while BRIDGE-K could not be run when $b = 24$ and $p = 0.5$.

We next compare BRIDGE-T and ByRDiE in Figure 4. Both BRIDGE-T and ByRDiE are resilient to two Byzantine nodes but since ByRDiE is based on a coordinate-wise screening method, the time it takes to reach the final optimal solution is thousands-fold more than BRIDGE-T. Indeed, since nodes within the BRIDGE framework compute the local gradients and communicate with their neighbors only once per iteration, this leads to massive savings in computation and communications costs in comparison with ByRDiE. This difference in terms of computation and communications costs is even more pronounced in higher-dimensional tasks; thus we will not compare with ByRDiE for the rest of our experiments.

Last but not the least, Figure 5 highlights the robustness of

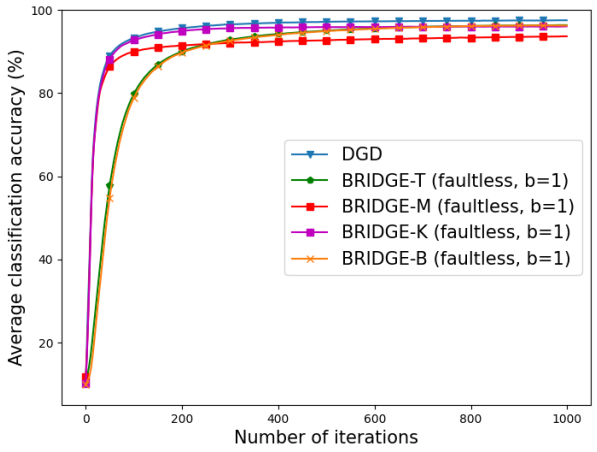

Fig. 6: Comparison between DGD and BRIDGE-T, -M, -K, and -B in the faultless setting for a nonconvex loss function with the MNIST dataset.

the BRIDGE framework on the higher-dimensional CIFAR-10 dataset for the case of a linear classifier. It can be seen from this figure that the earlier conclusions concerning the different variants of BRIDGE still hold, with BRIDGE-T approaching the state-of-the-art accuracy of $\sim 39\%$ [90] for a linear classifier. We conclude by noting that BRIDGE-B is excluded here and in the following for the experiments on CIFAR-10 dataset because of its relatively high computational complexity on larger-dimensional data.

B. Convolutional neural network on MNIST and CIFAR-10

In the theoretical analysis, we gave local convergence guarantees of BRIDGE-T in the nonconvex setting. In this set of experiments, we numerically show that BRIDGE indeed performs well in the nonconvex case. We train a convolutional neural network (CNN) on MNIST and CIFAR-10 datasets for this purpose, with the model including two convolutional layers followed by two fully connected layers. Each convolutional layer is followed by a max pooling and ReLU activation while the output layer uses softmax activation. We construct a network with 50 nodes and each pair of nodes has probability of 0.5 to be directly connected. Each node has access to 1200 samples randomly picked from the training set. We randomly choose two or four of the nodes to be Byzantine nodes, which broadcast random vectors to all their neighbors during each iteration for all screening methods. We again use the averaged classification accuracy on the MNIST and CIFAR-10 test sets over all nonfaulty nodes as the metric for performance. In this experiment, we cannot compare with ByRDiE due to the fact that the learning model is of high dimension, which makes ByRDiE unfeasible in this setting.

As we can see from Figure 6, the performances of all BRIDGE methods are as good as DGD in the faultless setting with 92% to 95% percent average accuracy. In Figure 7, we see that DGD fails in the faulty setting when $b = 2$ and $b = 4$. But BRIDGE-T, -M, -K, and -B are able to learn a relatively good model in these cases. The final set of results for the CIFAR-10 dataset obtained using BRIDGE-T, -M, and -K for the case of zero, two, and four Byzantine nodes is presented in Figure 8. The top-left quadrant in this figure is reserved for the accuracy of the centralized solution for the chosen CNN architecture.[2] It can be seen that both BRIDGE-T and -M remain resilient to Byzantine attacks and achieve accuracy similar to the centralized solution. However, BRIDGE-K gets stuck in a suboptimal critical point of the loss landscape for the case of two and four Byzantine nodes. ***Non-i.i.d data distribution on MNIST:*** In the interest of space, additional experimental results using non-i.i.d data distribution on MNIST are provided in Section S.I of the supplementary material.

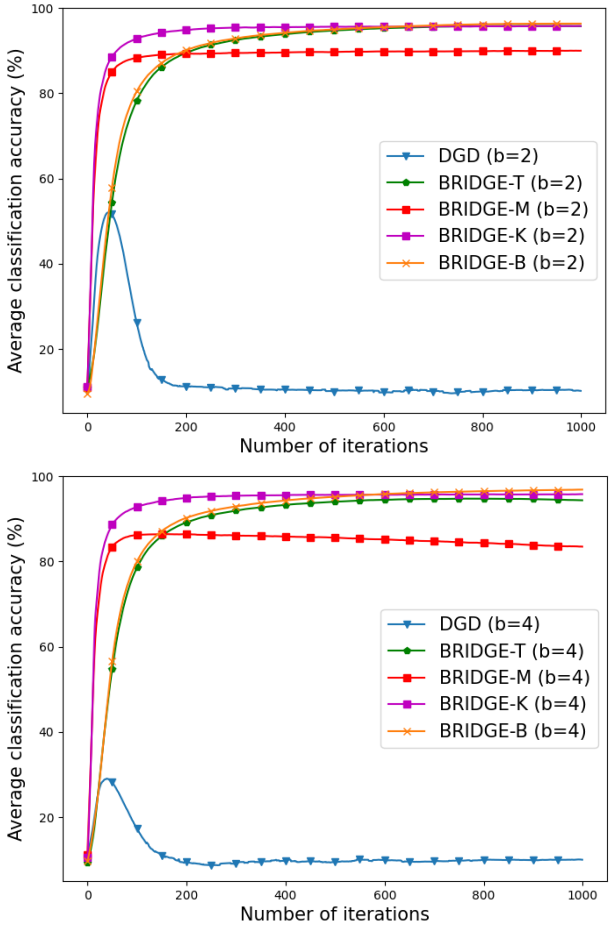

Fig. 7: Comparison between DGD and BRIDGE-T, -M, -K, and -B with two and four Byzantine nodes for nonconvex loss with the MNIST dataset.

VI. Conclusion

This paper introduced a new decentralized machine learning framework called Byzantine resilient decentralized gradient descent (BRIDGE). This framework was designed to solve machine learning problems when the training set is distributed over a decentralized network in the presence of Byzantine failures. Both theoretical results and experimental results were used to show that the framework could perform well while tolerating Byzantine attacks. One variant of the framework

[2]Note that a better CIFAR-10 accuracy can be obtained through a fine tuning of the CNN architecture and the step size sequence.

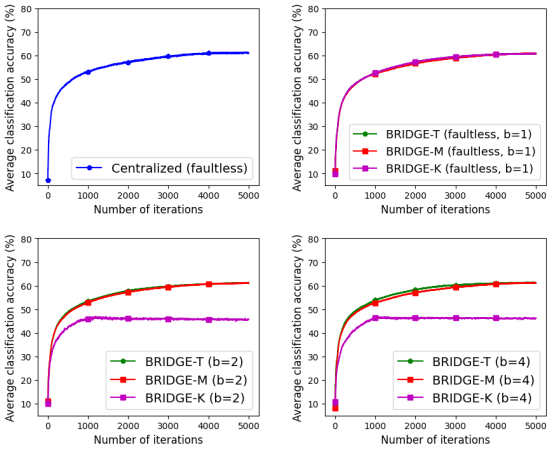

Fig. 8: Performance comparison of BRIDGE-T, -M, and -K with zero, two, and four Byzantine nodes for nonconvex loss with the CIFAR-10 dataset.

was shown to converge sublinearly to the global minimum in the convex setting and first-order stationary point in the convex setting. In addition, statistical convergence rates were also provided for this variant.

Future work aims to improve the framework to tolerate more Byzantine agents within the network with faster convergence rates and to deal with more general nonconvex objective functions with either i.i.d. or non-i.i.d. distribution of the dataset. In addition, an exploration of the number of Byzantine nodes that this framework can tolerate under different kinds of non-i.i.d. settings will also be undertaken in future work.

REFERENCES

[1] V. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed. New York, NY: Springer-Verlag, 1999.

[2] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1–47, 2002.

[3] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging Artificial Intell. Applicat. Comput. Eng.*, vol. 160, pp. 3–24, 2007.

[4] Y. Bengio, "Learning deep architectures for AI," *Found. and Trends Mach. Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[5] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*, 2nd ed. Cambridge, MA: MIT Press, 2018.

[6] R. M. Golden, *Statistical Machine Learning: A Unified Framework*. Boca Raton, FL: Chapman and Hall/CRC, 2020.

[7] Z. Yang, A. Gang, and W. U. Bajwa, "Adversary-resilient distributed and decentralized statistical inference and machine learning: An overview of recent advances under the Byzantine threat model," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 146–159, May 2020.

[8] M. Nokleby, H. Raja, and W. U. Bajwa, "Scaling-up distributed processing of data streams for machine learning," *Proceedings of the IEEE*, vol. 108, no. 11, pp. 1984–2012, 2020.

[9] J. B. Predd, S. B. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 56–69, 2006.

[10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. and Trends Mach. Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[11] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Found. and Trends Mach. Learning*, vol. 7, no. 4-5, pp. 311–801, 2014.

[12] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

[13] T. Sun, D. Li, and B. Wang, "Decentralized federated averaging," *arXiv preprint arXiv:2104.11375*, 2021.

[14] K. Driscoll, B. Hall, H. Sivencrona, and P. Zumsteq, "Byzantine fault tolerance, from theory to reality," in *Proc. Int. Conf. Computer Safety, Reliability, and Security (SAFECOMP'03)*, 2003, pp. 235–248.

[15] L. Su and N. H. Vaidya, "Fault-tolerant multi-agent optimization: Optimal iterative distributed algorithms," in *Proc. ACM Symp. Principles of Distributed Computing*, 2016, pp. 425–434.

[16] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Programming Languages and Syst.*, vol. 4, no. 3, pp. 382–401, 1982.

[17] P. Dutta, R. Guerraoui, and M. Vukolic, "Best-case complexity of asynchronous Byzantine consensus," EPFL/IC/200499, Tech. Rep., 2005.

[18] J. Sousa and A. Bessani, "From Byzantine consensus to BFT state machine replication: A latency-optimal transformation," in *Proc. 9th Euro. Dependable Computing Conf.(EDCC'12)*, 2012, pp. 37–48.

[19] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *Proc. 11th USENIX Symp. Operating Systems Design and Implementation (OSDI'14)*, Broomfield, CO, Oct. 2014, pp. 583–598.

[20] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Proc. NeurIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[21] P. Blanchard, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Advances in Neural Inf. Process. Syst.*, 2017, pp. 118–128.

[22] L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos, "DRACO: Byzantine-resilient distributed training via redundant gradients," in *Proc. 35th Intl. Conf. Machine Learning (ICML)*, 2018, pp. 903–912.

[23] X. Cao and L. Lai, "Robust distributed gradient descent with arbitrary number of Byzantine attackers," in *Proc. IEEE Int. Conf. Acoust. Speech and Signal Process. (ICASSP'19)*, 2018, pp. 6373–6377.

[24] L. Su and J. Xu, "Securing distributed machine learning in high dimensions," *arXiv preprint arXiv:1804.10140*, 2018.

[25] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Defending against saddle point attack in Byzantine-robust distributed learning," in *Proc. 36th Intl. Conf. Machine Learning*, Jun. 2019, pp. 7074–7084.

[26] G. Damaskinos, E. E. Mhamdi, R. Guerraoui, R. Patra, and M. Taziki, "Asynchronous Byzantine machine learning (the case of SGD)," in *Proc. 35th Int. Conf. Machine Learning*, 2018, pp. 1145–1154.

[27] E. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in Byzantium," in *Proc. 35th Int. Conf. Machine Learning*, 2018, pp. 3521–3530.

[28] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. 35th Intl. Conf. Machine Learning*, Jul. 2018, pp. 5650–5659.

[29] D. Alistarh, Z. Allen-Zhu, and J. Li, "Byzantine stochastic gradient descent," in *Proc. Advances in Neural Information Processing Systems*, 2018, pp. 4618–4628.

[30] C. Xie, O. Koyejo, and I. Gupta, "Zeno: Byzantine-suspicious stochastic gradient descent," *arXiv preprint arXiv:1805.10032*, 2018.

[31] ——, "Generalized Byzantine-tolerant SGD," *arXiv preprint arXiv:1802.10116*, 2018.

[32] ——, "Phocas: Dimensional Byzantine-resilient stochastic gradient descent," *arXiv preprint arXiv:1805.09682*, 2018.

[33] X. Chen, T. Chen, H. Sun, S. Wu, and M. Hong, "Distributed training with heterogeneous data: Bridging median- and mean-based algorithms," in *Proc. Advances in Neural Information Processing Systems*, 2020, pp. 21 616–21 626.

[34] S. Rajput, H. Wang, Z. Charles, and D. Papailiopoulos, "DETOX: A redundancy-based framework for faster and more robust gradient aggregation," in *Proc. Advances in Neural Information Processing Systems*, 2019.

[35] L. Li, W. Xu, T. Chen, G. Giannakis, and Q. Ling, "RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 1544–1551.

[36] R. Jin, X. He, and H. Dai, "Distributed Byzantine tolerant stochastic gradient descent in the era of big data," in *Proc. IEEE Intl. Conf. Communications (ICC)*, 2019, pp. 1–6.

[37] F. Lin, Q. Ling, and Z. Xiong, "Byzantine-resilient distributed large-scale matrix completion," in *Proc. IEEE Int. Conf. Acoust. Speech and Signal Process. (ICASSP'19)*, 2019, pp. 8167–8171.

[38] A. Ghosh, J. Hong, D. Yin, and K. Ramchandran, "Robust federated learning in a heterogeneous environment," *arXiv preprint arXiv:1906.06629*, 2019.

[39] D. Data, L. Song, and S. N. Diggavi, "Data encoding for Byzantine-resilient distributed optimization," *IEEE Transactions on Information Theory*, vol. 67, no. 2, pp. 1117–1140, 2021.

[40] E. M. E. Mhamdi, R. Guerraoui, A. Guirguis, and S. Rouault, "SGD: Decentralized Byzantine resilience," *arXiv preprint arXiv:1905.03853*, 2019.

[41] C. Xie, S. Koyejo, and I. Gupta, "Zeno++: Robust fully asynchronous SGD," in *Proc. 37th Intl. Conf. Machine Learning*, Jul. 2020, pp. 10 495–10 503.

[42] E.-M. El-Mhamdi, R. Guerraoui, and S. Rouault, "Fast and robust distributed learning in high dimension," *arXiv preprint arXiv:1905.04374*, 2019.

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

[44] S. S. Ram, A. Nedić, and V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *J. Optim. Theory and Appl.*, vol. 147, no. 3, pp. 516–545, 2010.

[45] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Trans. Autom. Control*, vol. 60, no. 3, pp. 601–615, 2015.

[46] S. Pu and A. Nedić, "Distributed stochastic gradient tracking methods," *Mathematical Programming*, vol. 187, pp. 409–457, 2021.

[47] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *J. Mach. Learning Research*, vol. 11, pp. 1663–1707, 2010.

[48] J. F. Mota, J. M. Xavier, P. M. Aquiar, and M. Puschel, "D-ADMM: A communication-efficient distributed algorithm for separable optimization," *IEEE Trans. Signal Process.*, vol. 61, no. 10, pp. 2718–2723, 2013.

[49] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Trans. Signal Process.*, vol. 62, no. 7, pp. 1750–1761, 2014.

[50] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "A decentralized second-order method with exact linear convergence rate for consensus optimization," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 4, pp. 507–522, 2016.

[51] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network Newton distributed optimization methods," *IEEE Trans. Signal Process.*, vol. 65, no. 1, pp. 146–161, 2017.

[52] H. J. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram, "Resilient asymptotic consensus in robust networks," *IEEE J. Sel. Areas in Commun.*, vol. 31, no. 4, pp. 766–781, 2013.

[53] N. H. Vaidya, L. Tseng, and G. Liang, "Iterative Byzantine vector consensus in incomplete graphs," in *Proc. 15th Int. Conf. Distributed Computing and Networking*, 2014, pp. 14–28.

[54] S. Sundaram and B. Gharesifard, "Distributed optimization under adversarial nodes," *IEEE Trans. Autom. Control*, vol. 64, no. 3, pp. 1063–1076, 2019.

[55] Z. Yang and W. U. Bajwa, "RD-SVM: A resilient distributed support vector machine," in *Proc. IEEE Int. Conf. Acoust. Speech and Signal Process. (ICASSP'16)*, 2016, pp. 2444–2448.

[56] W. Xu, Z. Li, and Q. Ling, "Robust decentralized dynamic optimization at presence of malfunctioning agents," *Signal Process.*, vol. 153, pp. 24–33, 2018.

[57] A. Mitra, J. Richards, S. Bagchi, and S. Sundaram, "Resilient distributed state estimation with mobile agents: Overcoming Byzantine adversaries, communication losses, and intermittent measurements," *Autonomous Robots*, vol. 43, no. 3, pp. 743–768, 2019.

[58] L. Su and S. Shahrampour, "Finite-time guarantees for Byzantine-resilient distributed state estimation with noisy measurements," *IEEE Transactions on Automatic Control*, vol. 65, no. 9, pp. 3758–3771, 2020.

[59] Z. Yang and W. U. Bajwa, "ByRDiE: Byzantine-resilient distributed coordinate descent for decentralized learning," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 4, pp. 611–627, Dec. 2019.

[60] K. Kuwarancharoen, L. Xin, and S. Sundaram, "Byzantine-resilient distributed optimization of multi-dimensional functions," in *Proc. American Control Conference (ACC)*, 2020, pp. 4399–4404.

[61] J. Peng, W. Li, and Q. Ling, "Byzantine-robust decentralized stochastic optimization over static and time-varying networks," *Signal Processing*, vol. 183, p. 108020, 2021.

[62] S. Guo, T. Zhang, X. Xie, L. Ma, T. Xiang, and Y. Liu, "Towards Byzantine-resilient learning in decentralized systems," *arXiv preprint arXiv:2002.08569*, 2020.

[63] E.-M. El-Mhamdi, R. Guerraoui, A. Guirguis, L. Hoang, and S. Rouault, "Collaborative learning as an agreement problem," *arXiv preprint arXiv:2008.00742v3*, 2020.

[64] P. D. Lorenzo and G. Scutari, "Next: In-network nonconvex optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, pp. 120–136, 2016.

[65] J. Zeng and W. Yin, "On nonconvex decentralized gradient descent," *IEEE Transactions on Signal Processing*, vol. 66, pp. 2834–2848, 2018.

[66] H. Sun, S. Lu, and M. Hong, "Improving the sample and communication complexity for decentralized non-convex optimization: Joint gradient estimation and tracking," in *Proc. 37th Intl. Conf. Machine Learning*, Jul. 2020, pp. 9217–9228.

[67] R. Xin, U. A. Khan, and S. Kar, "Fast decentralized non-convex finite-sum optimization with recursive variance reduction," *arXiv preprint arXiv:2008.07428*, 2020.

[68] L. He, S. P. Karimireddy, and M. Jaggi, "Byzantine-robust decentralized learning via self-centered clipping," *arXiv preprint arXiv:2202.01545*, 2022.

[69] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[70] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[71] H. H. Sohrab, *Basic Real Analysis*, 2nd ed. New York, NY: Springer, 2003.

[72] J. Sun, Q. Qu, and J. Wright, "When are nonconvex problems not scary?" *arXiv preprint arXiv:1510.06096*, 2015.

[73] P. Jain and P. Kar, "Non-convex optimization for machine learning," *Foundations and Trends in Machine Learning*, vol. 10, no. 3-4, p. 142–336, 2017.

[74] B. Yonel and B. Yazici, "A deterministic theory for exact non-convex phase retrieval," *IEEE Transactions on Signal Processing*, vol. 68, pp. 4612–4626, 2020.

[75] Y. Zhou, H. Zhang, and Y. Liang, "Geometrical properties and accelerated gradient solvers of non-convex phase retrieval," in *Proc. 54th Annu. Allerton Conf. Communication, Control, and Computing*, 2016, pp. 331–335.

[76] T. Ypma, "Local convergence of inexact Newton methods," *SIAM Journal on Numerical Analysis*, vol. 21, no. 3, pp. 583–590, 1984.

[77] P. Ochs, "Local convergence of the heavy-ball method and iPiano for non-convex optimization," *Journal of Optimization Theory and Applications*, vol. 177, no. 1, pp. 153–180, 2018.

[78] S. Bock and M. Weiß, "A proof of local convergence for the Adam optimizer," in *Proc. International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[79] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Trans. Autom. control*, vol. 57, no. 3, pp. 592–606, 2012.

[80] L. Su and N. Vaidya, "Multi-agent optimization in the presence of Byzantine adversaries: Fundamental limits," in *Proc. American Control Conference (ACC)*, 2016, pp. 7183–7188.

[81] H. Yang, X. zhong Zhang, M. Fang, and J. Liu, "Byzantine-resilient stochastic gradient descent for distributed learning: A Lipschitz-inspired coordinate-wise median approach," *Proc. IEEE Conference on Decision and Control (CDC)*, pp. 5832–5837, 2019.

[82] D. Data and S. Diggavi, "Byzantine-resilient high-dimensional SGD with local iterations on heterogeneous data," in *Proc. 38th Intl. Conf. Machine Learning*, Jul. 2021, pp. 2478–2488.

[83] P. J. Huber, *Robust Statistics*. Berlin, Heidelberg: Springer, 2011.

[84] L. Su and N. H. Vaidya, "Fault-tolerant distributed optimization (part IV): Constrained optimization with arbitrary directed networks," *arXiv preprint arXiv:1511.01821*, 2015.

[85] Y. Nesterov, *Introductory Lectures on Convex Optimization*, ser. Applied optimization; v. 87. Springer US, 2004.

[86] Z. Yang and W. U. Bajwa, "BRIDGE: Byzantine-resilient decentralized gradient descent," *arXiv preprint arXiv:1908.08098v1*, Aug. 2019. [Online]. Available: https://arxiv.org/abs/1908.08098v1

[87] S. Mei, Y. Bai, and A. Montanari, "The landscape of empirical risk for nonconvex losses," *The Annals of Statistics*, vol. 46, no. 6A, pp. 2747 – 2774, 2018.

[88] D. Davis and D. Drusvyatskiy, "Graphical convergence of subgradients in nonconvex optimization and learning," *Mathematics of Operations Research*, vol. 47, no. 1, pp. 209–231, 2021.

[89] D. J. Foster, A. Sekhari, and K. Sridharan, "Uniform convergence of gradients for non-convex learning and optimization," in *Proc. Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018.

[90] T. V. J. Le and N. Gopee, "Classifying CIFAR-10 images using unsupervised feature & ensemble learning," Dec 2016. [Online]. Available: https://trucvietle.me/files/601-report.pdf

Supplementary Material

BRIDGE: Byzantine-resilient Decentralized Gradient Descent

Cheng Fang, Zhixiong Yang, and Waheed U. Bajwa

S.I. Non-i.i.d. data distribution on MNIST

In the theoretical analysis section, we gave convergence gurantees for BRIDGE-T in both convex and nonconvex settings. However, the main results are based on identical and independent distribution (i.i.d.) of the dataset. In this section, we compare our method to the one proposed in [1], which we term "Byzantine-robust decentralized stochastic optimization" (BRDSO) in the following discussion based on the terminology used in [1]. We compare BRIDGE-T with BRDSO [1] in the following non-i.i.d. settings.

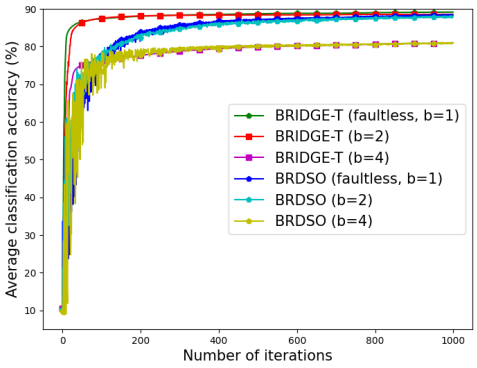

Fig. S.1: Comparing BRIDGE-T with BRDSO [1] in the extreme non-i.i.d. setting (convex loss and MNIST dataset).

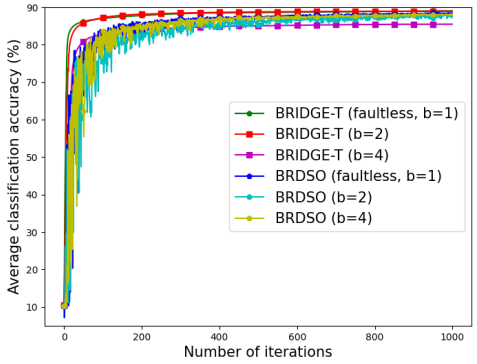

Fig. S.2: Comparing BRIDGE-T with BRDSO [1] in the moderate non-i.i.d. setting (convex loss and MNIST dataset)

Extreme non-i.i.d. setting: We group the dataset corresponding to labels and distribute all the samples labelled "0" to 5 agents, distribute all the samples labelled "1" to another 5 agents, and so on. We can see from Figure S.1 that when the number of Byzantine nodes is equal to 0 or 2, the accuracies of both the algorithms are as good as in the i.i.d. case, while in the case when the number of Byzantine nodes is equal to 4, there is about 9 percent of accuracy drop due to the non-i.i.d. distribution of data for both algorithms. In this extreme non-i.i.d. setting when four of the agents are chosen to be Byzantine, the worst-case scenario happens when all the Byzantine nodes are assigned all samples with the same label. This means 80 percent of the samples of one label is not being used towards the training process, which causes both algorithms to underperform compared to the i.i.d. setting.

Moderate non-i.i.d. setting: We group the dataset corresponding to labels and distribute the samples associated with each label evenly to 10 agents. Every agent receives two sets of differently labelled data evenly. As we can see from Figure S.2, both algorithms perform as well as in the i.i.d. setting in the presence of two or four Byzantine nodes. We conclude that, with distribution closer to i.i.d., the impact of Byzantine nodes in the non-i.i.d. setting will be less.

S.II. Proof of Lemma 1

First, we drop $\mathbb{P}$ and $\mathbf{z}$ in notation of the statistical risk for convenience and observe that for any dimension k we have

$$\mathbb{E}[\mathbf{g}_2(t)]_k = \mathbb{E}\sum_{j=1}^{r}[\boldsymbol{\alpha}_k(t)]_j[\nabla f_j(\mathbf{v}(t))]_k = \mathbb{E}[\nabla f(\mathbf{v}(t))]_k.$$

Since k is arbitrary, it then follows that

$$\mathbb{E}[\mathbf{g}_2(t)] = \mathbb{E}[\nabla f(\mathbf{v}(t))]. \tag{S.1}$$

In the definition of $\mathbf{g}_2(t)$, notice that $\mathbf{v}(t)$ depends on t and $\boldsymbol{\alpha}_k(t)$ depends on both t and k. We therefore need to show that the convergence of $\mathbf{g}_2(t)$ to $\mathbb{E}[\nabla f(\mathbf{v}(t))]$ is uniformly over all $\mathbf{v}(t)$ and $\boldsymbol{\alpha}_k(t)$. We fix an arbitrary coordinate k and drop the index k for the rest of this section for simplicity. We next define a vector $\mathbf{h}(t)$ as $\mathbf{h}(t) := [[\nabla f_j(\mathbf{v}(t))] : j \in \mathcal{R}]$ and note that $\mathbf{g}_2(t) = \langle \boldsymbol{\alpha}(t), \mathbf{h}(t)\rangle$. Since the training data are i.i.d., $\mathbf{h}(t)$ has identically distributed elements. We therefore have from Hoeffding's inequality [2] that for any $\epsilon_0 \in (0,1)$:

$$\mathbb{P}\big(|\langle \boldsymbol{\alpha}(t), \mathbf{h}(t)\rangle - \mathbb{E}[\nabla f(\mathbf{v}(t))]| \geq \epsilon_0\big) \leq 2\exp\Big(-\frac{2N\epsilon_0^2}{L^2\|\boldsymbol{\alpha}(t)\|^2}\Big). \tag{S.2}$$

Further, since the r-dimensional vector $\boldsymbol{\alpha}(t)$ is an arbitrary element of the standard simplex, defined as

$$\Delta := \{\mathbf{q} \in \mathbb{R}^r : \sum_{j=1}^{r}[\mathbf{q}]_j = 1 \text{ and } \forall j, [\mathbf{q}]_j \geq 0\}, \tag{S.3}$$

the probability bound in (S.2) also holds for any $\mathbf{q} \in \Delta$, i.e.,

$$\mathbb{P}\Big(|\langle \mathbf{q}, \mathbf{h}(t)\rangle - \mathbb{E}[\nabla f(\mathbf{v}(t))]| \geq \epsilon_0\Big) \leq 2\exp\Big(-\frac{2N{\epsilon_0}^2}{L^2\|\mathbf{q}\|^2}\Big). \tag{S.4}$$

We now define the set $\mathcal{S}_{\boldsymbol{\alpha}} := \{\boldsymbol{\alpha}_k(t)\}_{t,k=1}^{\infty,d}$. Our next goal is to leverage (S.4) and derive a probability bound similar to (S.2) that *uniformly* holds for *all* $\mathbf{q} \in \mathcal{S}_{\boldsymbol{\alpha}}$. To this end, let

$$\mathbb{C}_\xi := \{\mathbf{c}_1, \ldots, \mathbf{c}_{d_\xi}\} \subset \Delta \quad \text{s.t.} \quad \mathcal{S}_{\boldsymbol{\alpha}} \subseteq \bigcup_{q=1}^{d_\xi} \mathbb{B}(\mathbf{c}_q, \xi) \tag{S.5}$$

denote a ξ-covering of $\mathcal{S}_{\boldsymbol{\alpha}}$ in terms of the ℓ_2 norm and define $\bar{\mathbf{c}} := \arg\max_{\mathbf{c}\in\mathbb{C}_\xi} \|\mathbf{c}\|$. Then from (S.4) and the union bound

$$\mathbb{P}\bigg(\sup_{\mathbf{c}\in\mathbb{C}_\xi} |\langle \mathbf{c}, \mathbf{h}(t)\rangle - \mathbb{E}[\nabla f(\mathbf{v}(t))]| \geq \epsilon_0 \bigg) \leq 2d_\xi \exp\bigg(-\frac{2N{\epsilon_0}^2}{L^2\|\bar{\mathbf{c}}\|^2}\bigg). \tag{S.6}$$

In addition, we have

$$\begin{aligned}\sup_{\mathbf{q}\in\mathcal{S}_{\boldsymbol{\alpha}}} |\langle \mathbf{q}, \mathbf{h}(t)\rangle - \mathbb{E}[\nabla f(\mathbf{v}(t))]| &\overset{(a)}{\leq} \sup_{\mathbf{q}\in\mathcal{S}_{\boldsymbol{\alpha}}, \mathbf{c}\in\mathbb{C}_\xi} \|\mathbf{q}-\mathbf{c}\|\|\mathbf{h}(t)\| \\ &+ \sup_{\mathbf{c}\in\mathbf{c}_\xi} |\langle \mathbf{c}, \mathbf{h}(t)\rangle - \mathbb{E}[\nabla f(\mathbf{v}(t))]|, \end{aligned} \tag{S.7}$$

where (a) is due to the triangle and Cauchy–Schwarz inequalities. Trivially, $\sup_{\mathbf{q}\in\mathcal{S}_{\boldsymbol{\alpha}}, \mathbf{c}\in\mathbb{C}_\xi} \|\mathbf{q}-\mathbf{c}\| \leq \xi$ from the definition of $\mathbb{C}_\xi$, while $\|\mathbf{h}(t)\| \leq \sqrt{r}L$ from the definition of $\mathbf{h}(t)$ and Assumption 1. Combining (S.6) and (S.7), we get

$$\mathbb{P}\bigg(\sup_{\mathbf{q}\in\mathcal{S}_{\boldsymbol{\alpha}}} |\langle \mathbf{q}, \mathbf{h}(t)\rangle - \mathbb{E}[\nabla f(\mathbf{v}(t))]| \geq \epsilon_0 + \sqrt{r}\xi L \bigg) \leq 2d_\xi \exp\bigg(-\frac{2N{\epsilon_0}^2}{L^2\|\bar{\mathbf{c}}\|^2}\bigg). \tag{S.8}$$

We now define $\boldsymbol{\alpha}_m := \arg\max_{\mathbf{q}\in\mathcal{S}_{\boldsymbol{\alpha}}} \|\mathbf{q}\|$. It can then be shown from the definitions of $\mathbb{C}_\xi$ and $\bar{\mathbf{c}}$ that

$$\|\bar{\mathbf{c}}\|^2 \leq 2(\|\boldsymbol{\alpha}_m\|^2 + \xi^2). \tag{S.9}$$

Therefore, fixing $\epsilon_0 \in (0,1)$, and defining $\epsilon' := 2\epsilon_0$ and $\xi := \epsilon'/(2L\sqrt{r})$, we have from (S.8) and (S.9) that

$$\mathbb{P}\bigg(\sup_{\mathbf{q}\in\mathcal{S}_{\boldsymbol{\alpha}}} |\langle \mathbf{q}, \mathbf{h}(t)\rangle - \mathbb{E}[\nabla f(\mathbf{v}(t))]| \geq \epsilon' \bigg) \leq 2d_\xi \exp\bigg(-\frac{4rN{\epsilon'}^2}{4L^2r\|\boldsymbol{\alpha}_m\|^2 + {\epsilon'}^2}\bigg). \tag{S.10}$$

Note that (S.10) is derived for a fixed but arbitrary k. Extending this to the entire vector gives us for any $\mathbf{v}(t)$

$$\mathbb{P}\bigg(\|\mathbf{g}_2(t) - \mathbb{E}[\nabla f(\mathbf{v}(t))]\| \geq \sqrt{d}\epsilon' \bigg) \leq 2d_\xi \exp\bigg(-\frac{4rN{\epsilon'}^2}{4L^2r\|\boldsymbol{\alpha}_m\|^2 + {\epsilon'}^2}\bigg). \tag{S.11}$$

To obtain the desired uniform bound, we next need to remove the dependence on $\mathbf{v}(t)$ in (S.11). Here we drop t from $\mathbf{v}(t)$ for simplicity of notation and write $\mathbf{g}_2(t)$ as $\mathbf{g}_2(\mathbf{v})$ to show the dependence of $\mathbf{g}_2$ on $\mathbf{v}$. Notice from our discussion in the beginning of Section IV and the analysis in Section IV-A that $\mathbf{v}(t) \in \mathbb{V} := \{\mathbf{v} : \|\mathbf{v}\| \leq \Gamma_0\}$ for some Γ_0 and all t. We then define $\mathbf{E}_\zeta := \{\mathbf{e}_1, \ldots, \mathbf{e}_{m_\zeta}\} \subset \mathbb{V}$ to be a ζ-covering of $\mathbb{V}$ in terms of the ℓ_2 norm. It then follows from (S.11) that

$$\mathbb{P}\bigg(\sup_{\mathbf{e}\in\mathbf{E}_\zeta} \|\mathbf{g}_2(\mathbf{e}) - \mathbb{E}[\nabla f(\mathbf{e})]\| \geq \sqrt{d}\epsilon' \bigg) \leq 2d_\xi m_\zeta \exp\bigg(-\frac{4rN{\epsilon'}^2}{4L^2r\|\boldsymbol{\alpha}_m\|^2 + {\epsilon'}^2}\bigg). \tag{S.12}$$

Similar to (S.7), we can also write

$$\begin{aligned}\sup_{\mathbf{v}\in\mathbb{V}} \|\mathbf{g}_2(\mathbf{v}) - \mathbb{E}[\nabla f(\mathbf{v})]\| &\leq \sup_{\mathbf{e}\in\mathbf{E}_\zeta} \|\mathbf{g}_2(\mathbf{e}) - \mathbb{E}[\nabla f(\mathbf{e})]\| \\ &+ \sup_{\mathbf{e}\in\mathbf{E}_\zeta, \mathbf{v}\in\mathbb{V}} \Big[\|\mathbf{g}_2(\mathbf{v}) - \mathbf{g}_2(\mathbf{e})\| + \|\mathbb{E}[\nabla f(\mathbf{e})] - \mathbb{E}[\nabla f(\mathbf{v})]\| \Big]. \end{aligned} \tag{S.13}$$

Further, Assumption 1 and definition of the set $\mathbf{E}_\zeta$ imply

$$\sup_{\mathbf{e}\in\mathbf{E}_\zeta, \mathbf{v}\in\mathbb{V}} \|\mathbf{g}_2(\mathbf{v}) - \mathbf{g}_2(\mathbf{e})\| \leq L'\zeta. \tag{S.14}$$

We now define $\epsilon'' := 2\epsilon'\sqrt{d}$ and $\zeta := \epsilon''/4L'$. We then obtain the following from (S.11)–(S.14):

$$\mathbb{P}\bigg(\sup_{\mathbf{v}\in\mathbb{V}} \|\mathbf{g}_2(\mathbf{v}) - \mathbb{E}[\nabla f(\mathbf{v})]\| \geq \epsilon'' \bigg) \leq 2d_\xi m_\zeta \exp\bigg(-\frac{4rN{\epsilon''}^2}{16L^2rd\|\boldsymbol{\alpha}_m\|^2 + {\epsilon''}^2}\bigg). \tag{S.15}$$

Since $\mathbf{v}(t) \in \mathbb{V}$ for all t, we then have

$$\mathbb{P}\bigg(\sup_{t} \|\mathbf{g}_2(\mathbf{v}(t)) - \mathbb{E}[\nabla f(\mathbf{v}(t))]\| \geq \epsilon'' \bigg) \leq 2d_\xi m_\zeta \exp\bigg(-\frac{4rN{\epsilon''}^2}{16L^2rd\|\boldsymbol{\alpha}_m\|^2 + {\epsilon''}^2}\bigg). \tag{S.16}$$

The proof now follows from (S.16) and the following facts about the covering numbers of the sets $\mathcal{S}_{\boldsymbol{\alpha}}$ and $\mathbb{V}$: (1) Since $\mathcal{S}_{\boldsymbol{\alpha}}$ is a subset of Δ, which can be circumscribed by a sphere in $\mathbb{R}^{r-1}$ of radius $\sqrt{r-1/r} < 1$, we can upper bound d_ξ by $\left(\frac{12L\sqrt{rd}}{\epsilon''}\right)^r$ [3]; and (2) Since $\mathbb{V} \subset \mathbb{R}^d$ can be circumscribed by a sphere in $\mathbb{R}^d$ of radius $\Gamma_0\sqrt{d}$, we can upper bound m_ζ by $\left(\frac{12L'\Gamma_0\sqrt{d}}{\epsilon''}\right)^d$. Then for any $\epsilon'' \in (0,1)$, we have

$$\sup_{t} \|\mathbf{g}_2(\mathbf{v}(t)) - \mathbb{E}[\nabla f(\mathbf{v}(t))]\| < \epsilon'' \tag{S.17}$$

with probability exceeding

$$1 - 2\exp\bigg(-\frac{4rN{\epsilon''}^2}{16L^2rd\|\boldsymbol{\alpha}_m\|^2 + {\epsilon''}^2} + r\log\left(\frac{12L\sqrt{rd}}{\epsilon''}\right) + d\log\left(\frac{12L'\Gamma_0\sqrt{d}}{\epsilon''}\right)\bigg). \tag{S.18}$$

Equivalently, we have with probability at least $1-\delta$ that

$$\sup_{t} \|\mathbf{g}_2(\mathbf{v}(t)) - \mathbb{E}[\nabla f(\mathbf{v}(t))]\| < \mathcal{O}\left(\sqrt{\frac{4L^2d\|\boldsymbol{\alpha}_m\|^2\log\frac{2}{\delta}}{N}}\right),$$

where $\delta = 2\exp\Big(-\frac{4rN{\epsilon''}^2}{16L^2rd\|\boldsymbol{\alpha}_m\|^2+{\epsilon''}^2} + r\log\left(\frac{12L\sqrt{rd}}{\epsilon''}\right) + d\log\left(\frac{12L'\Gamma_0\sqrt{d}}{\epsilon''}\right)\Big)$. ■

S.III. Proof of Lemma 2

All statements in this proof are probabilistic, with probability at least $1-\delta$ and δ being given in Appendix S.II. In particular, the discussion should be assumed to have been implicitly conditioned on this high probability event. From (31), we iteratively add up from $\mathbf{v}(0)$ to $\mathbf{v}(t+1)$ and get

$$\begin{aligned}
\|\mathbf{v}(t+1)-\mathbf{w}^*\| &\le \left(1-\frac{1}{t+t_0}\right)\|\mathbf{v}(t)-\mathbf{w}^*\| \\
&\quad + a_2(t)+a_3(t) \\
&\le \left(1-\frac{1}{t+t_0}\right)\Bigg[\left(1-\frac{1}{t-1+t_0}\right)\|\mathbf{v}(t-1)-\mathbf{w}^*\| \\
&\quad + a_2(t-1,N)+a_3(t-1)\Bigg] + a_2(t)+a_3(t) \\
&\le \prod_{\tau=0}^{t}\left(1-\frac{1}{\tau+t_0}\right)\|\mathbf{v}(0)-\mathbf{w}^*\| + a_2(t)+a_3(t) \\
&\quad + \sum_{\tau=0}^{t-1}[a_2(\tau)+a_3(\tau)]\prod_{z=\tau+1}^{t}\left(1-\frac{1}{z+t_0}\right). \qquad \text{(S.19)}
\end{aligned}$$

The first term in (S.19) can be simplified as

$$\begin{aligned}
&\prod_{\tau=0}^{t}\left(1-\frac{1}{\tau+t_0}\right)\|\mathbf{v}(0)-\mathbf{w}^*\| \\
&= \left(1-\frac{1}{0+t_0}\right)\left(1-\frac{1}{1+t_0}\right)\cdots\left(1-\frac{1}{t+t_0}\right)\|\mathbf{v}(0)-\mathbf{w}^*\| \\
&= \left(\frac{t_0-1}{t+t_0}\right)C_1, \qquad \text{(S.20)}
\end{aligned}$$

where $C_1=\|\mathbf{v}(0)-\mathbf{w}^*\|$.

The remaining terms in (S.19) can be simplified as

$$\begin{aligned}
&a_2(t)+a_3(t)+\sum_{\tau=0}^{t-1}[a_2(\tau)+a_3(\tau)]\prod_{z=\tau+1}^{t}\left(1-\frac{1}{z+t_0}\right) \\
&\le \left(1-\frac{1}{t+t_0}\right)a_2(t-1) \\
&\quad + \left(1-\frac{1}{t+t_0}\right)\left(1-\frac{1}{t-1+t_0}\right)a_2(t-2) \\
&\quad + a_2(t)+\cdots+\left(1-\frac{1}{t+t_0}\right)\cdots\left(1-\frac{1}{1+t_0}\right)a_2(0) \\
&\quad + \frac{L'}{\lambda(t+t_0)}a_4(t)+\left(1-\frac{1}{t+t_0}\right)\frac{L'}{\lambda(t-1+t_0)}a_4(t-1) \\
&\quad + \cdots+\left(1-\frac{1}{t+t_0}\right)\cdots\left(1-\frac{1}{1+t_0}\right)\frac{L'}{\lambda(t_0)}a_4(0) \\
&= C_2(N)\rho(t)+C_2(N)\rho(t-1)\frac{t+t_0-1}{t+t_0} \\
&\quad + \cdots+C_2(N)\rho(0)\frac{t_0}{t+t_0}+\frac{L'}{\lambda(t+t_0)}a_4(t) \\
&\quad + \frac{t+t_0-1}{\lambda(t+t_0)}\frac{1}{t-1+t_0}L'a_4(t-1) \\
&\quad + \cdots+\frac{t_0}{\lambda(t+t_0)}\frac{1}{t_0}L'a_4(0) \\
&= \frac{t}{\lambda(t+t_0)}C_2(N)+\frac{L'}{\lambda}\frac{1}{t_0+t}[a_4(0)+\cdots+a_4(t)], \qquad \text{(S.21)}
\end{aligned}$$

where $C_2(N)=\mathcal{O}\left(\sqrt{\frac{d\|\boldsymbol{\alpha}_m\|^2\log\frac{2}{\delta}}{N}}\right)$.

The term $a_4(0)+\cdots+a_4(t)$ in (S.21) can be further simplified using (22) as:

$$\begin{aligned}
a_4(0)+\cdots+a_4(t) &\le \sqrt{d}\left(rC_w+rL\frac{1}{\lambda t_0}\mu^{\frac{1}{\nu}}\right) \\
&\quad + \sqrt{d}\left[rC_w\mu^{\frac{1}{\nu}}+rL\frac{1}{\lambda t_0}\mu^{\frac{2}{\nu}}+rL\frac{1}{\lambda(1+t_0)}\mu^{\frac{1}{\nu}}\right]+\cdots \\
&\quad + \sqrt{d}\Bigg[rC_w\mu^{\frac{t}{\nu}}+rL\frac{1}{\lambda t_0}\mu^{\frac{t+1}{\nu}}+rL\frac{1}{\lambda(1+t_0)}\mu^{\frac{t}{\nu}} \\
&\quad + rL\frac{1}{\lambda(2+t_0)}\mu^{\frac{t-1}{\nu}}+\cdots+rL\frac{1}{\lambda(t+t_0)}\mu^{\frac{1}{\nu}}\Bigg] \\
&\le \sqrt{d}\left(rC_w+rC_w\mu^{\frac{1}{\nu}}+rC_w\mu^{\frac{2}{\nu}}+\cdots+rC_w\mu^{\frac{t}{\nu}}\right) \\
&\quad + \frac{\sqrt{d}rL}{\lambda}\mu^{\frac{1}{\nu}}\left(\frac{1}{t_0}+\frac{1}{t_0+1}+\cdots\frac{1}{t_0+t}\right) \\
&\quad + \frac{\sqrt{d}rL}{\lambda}\mu^{\frac{2}{\nu}}\left(\frac{1}{t_0}+\frac{1}{t_0+1}+\cdots+\frac{1}{t_0+t}\right)+\cdots+ \\
&\quad + \frac{\sqrt{d}rL}{\lambda}\mu^{\frac{t}{\nu}}\left(\frac{1}{t_0}+\frac{1}{t_0+1}+\cdots+\frac{1}{t_0+t}\right) \\
&= \sqrt{d}rC_w\frac{(1-\mu^{\frac{t}{\nu}})}{1-\mu^{\frac{1}{\nu}}} \\
&\quad + \frac{\sqrt{d}rL}{\lambda}\frac{\mu^{\frac{1}{\nu}}(1-\mu^{\frac{t+1}{\nu}})}{1-\mu^{\frac{1}{\nu}}}\left(\frac{1}{t_0}+\frac{1}{t_0+1}+\cdots+\frac{1}{t_0+t}\right). \qquad \text{(S.22)}
\end{aligned}$$

Plugging (S.22) into (S.21) we finish the simplification of the remaining terms as:

$$\begin{aligned}
&a_2(t)+a_3(t)+\sum_{\tau=0}^{t-1}[a_2(\tau)+a_3(\tau)]\prod_{z=\tau+1}^{t}\left(1-\frac{1}{z+t_0}\right) \\
&\le \frac{1}{\lambda}C_2(N)+\frac{L'}{\lambda(t_0+t)}\left(\frac{\sqrt{d}rC_w}{1-\mu^{\frac{1}{\nu}}}\right) \\
&\quad + \frac{\sqrt{d}LL'r\mu^{\frac{1}{\nu}}}{\lambda^2\left(1-\mu^{\frac{1}{\nu}}\right)}\frac{1}{t+t_0}\left(\frac{1}{t_0}+\frac{1}{1+t_0}+\cdots+\frac{1}{t+t_0}\right). \qquad \text{(S.23)}
\end{aligned}$$

Finally we express $\|\mathbf{v}(t+1)-\mathbf{w}^*\|$ using (S.20) and (S.23) as:

$$\begin{aligned}
\|\mathbf{v}(t+1)-\mathbf{w}^*\| &\le \frac{t_0}{t+t_0}C_1+\frac{C_2(N)}{\lambda}+\frac{C_3}{t+t_0} \\
&\quad + \frac{C_4}{t+t_0}\left(\frac{1}{t_0}+\frac{1}{1+t_0}+\cdots+\frac{1}{t+t_0}\right), \qquad \text{(S.24)}
\end{aligned}$$

where $C_3=\frac{\sqrt{d}rC_wL'}{\lambda\left(1-\mu^{\frac{1}{\nu}}\right)}$ and $C_4=\frac{\sqrt{d}LL'r\mu^{\frac{1}{\nu}}}{\lambda^2\left(1-\mu^{\frac{1}{\nu}}\right)}$. ■

S.IV. Proof of Lemma 3

Similar to the proof of Lemma 2, our statements in this appendix are also conditioned on the high probability event

described in Appendix S.II. From Theorem 1 and (S.24) in the proof of Lemma 2, we conclude for all $j \in \mathcal{R}$ that

$$\|\mathbf{w}_j(t+1) - \mathbf{w}^*\| \leq \frac{t_0}{t+t_0}C_1 + \frac{C_2(N)}{\lambda} + \frac{C_3}{t+t_0} + \frac{C_4}{t+t_0}\left(\frac{1}{t_0} + \frac{1}{1+t_0} + \cdots + \frac{1}{t+t_0}\right) + a_4(t+1). \tag{S.25}$$

Next, we get an upper bound on $a_4(t+1)$ as following when we choose t as an even number:

$$\begin{aligned} a_4(t+1) &\leq \sqrt{d}rC_w\mu^{\frac{t+1}{\nu}} + \sqrt{d}rL\Big[\rho(0)\mu^{\frac{t+1}{\nu}} + \rho(1)\mu^{\frac{t}{\nu}} + \cdots \\ &\quad + \rho(t-1)\mu^{\frac{2}{\nu}} + \rho(t)\mu^{\frac{1}{\nu}} + \rho(t+1)\Big] \\ &\leq \sqrt{d}rC_w\mu^{\frac{t+1}{\nu}} \\ &\quad + \sqrt{d}rL\left[\rho(0)\mu^{\frac{t+1}{\nu}} + \cdots + \rho(0)\mu^{\frac{\frac{t}{2}+1}{\nu}}\right] \\ &\quad + \sqrt{d}rL\left[\rho\left(\frac{t}{2}+1\right)\mu^{\frac{\frac{t}{2}}{\nu}} + \cdots + \rho\left(\frac{t}{2}+1\right)\mu^{\frac{1}{\nu}} + \rho\left(\frac{t}{2}+1\right)\right] \\ &= \sqrt{d}rC_w\mu^{\frac{t+1}{\nu}} + \sqrt{d}rL\rho(0)\mu^{\frac{t+2}{2\nu}}\frac{1-\mu^{\frac{t}{2\nu}}}{1-\mu^{\frac{1}{\nu}}} \\ &\quad + \sqrt{d}rL\rho\left(\frac{t}{2}+1\right)\frac{1-\mu^{\frac{t}{2\nu}}}{1-\mu^{\frac{1}{\nu}}} \\ &\leq \sqrt{d}rC_w\mu^{\frac{t+1}{\nu}} \\ &\quad + \sqrt{d}rL\frac{1}{1-\mu^{\frac{1}{\nu}}}\left[\rho(0)\mu^{\frac{t+2}{2\nu}} + \rho\left(\frac{t}{2}+1\right)\right]. \end{aligned} \tag{S.26}$$

When t is an odd number, we have:

$$\begin{aligned} &\sqrt{d}rL\Big[\rho(0)\mu^{\frac{t+1}{\nu}} + \rho(1)\mu^{\frac{t}{\nu}} \\ &\quad + \cdots + \rho(t-1)\mu^{\frac{2}{\nu}} + \rho(t)\mu^{\frac{1}{\nu}} + \rho(t+1)\Big] \\ &\leq \sqrt{d}rL\left[\rho(0)\mu^{\frac{t+1}{\nu}} + \cdots + \rho(0)\mu^{\frac{\frac{t}{2}+\frac{1}{2}}{\nu}}\right] \\ &\quad + \sqrt{d}rL\left[\rho\left(\frac{t}{2}+\frac{1}{2}\right)\mu^{\frac{\frac{t}{2}-\frac{1}{2}}{\nu}} + \cdots + \rho\left(\frac{t}{2}+\frac{1}{2}\right)\mu^{\frac{1}{\nu}} \right. \\ &\quad \left. + \rho\left(\frac{t}{2}+\frac{1}{2}\right)\right] \end{aligned} \tag{S.27}$$

and the remaining steps are similar to (S.26), so we omit them.

Plugging either (S.26) or (S.27) into (S.25), we have for all $j \in \mathcal{R}$

$$\begin{aligned} \|\mathbf{w}_j(t+1) - \mathbf{w}^*\| &\leq \frac{t_0}{t+t_0}C_1 + \frac{C_2(N)}{\lambda} + \frac{C_3}{t+t_0} \\ &\quad + \frac{C_4}{t+t_0}\left(\frac{1}{t_0} + \frac{1}{1+t_0} + \cdots + \frac{1}{t+t_0}\right) \\ &\quad + \sqrt{d}rC_w\mu^{\frac{t+1}{\nu}} + \sqrt{d}rL\frac{1}{1-\mu^{\frac{1}{\nu}}}\left[\rho(0)\mu^{\frac{t+2}{2\nu}} + \rho\left(\frac{t}{2}+1\right)\right]. \end{aligned} \tag{S.28}$$

From (S.28) we see that all the terms except $\frac{C_2(N)}{\lambda}$ are monotonically decreasing with increasing t. Thus, given any $\epsilon > \frac{C_2(N)}{\lambda} > 0$, we can find a t_1 such that for all $t \geq t_1$, with probability at least $1-\delta$, $\|\mathbf{w}_j(t+1) - \mathbf{w}^*\| \leq \epsilon$. ■

S.V. Proof of Theorem 2

Lemma 3 establishes the convergence of $\mathbf{w}_j(t+1)$ to $\mathbf{w}^*$ for all $j \in \mathcal{R}$ with probability at least $1-\delta$. Next, we derive the rate of convergence. From (S.25) since $a_4(t)$ has a convergence rate of $\mathcal{O}(1/t)$, as mentioned in Theorem 1, the term that converges the slowest to 0 among all the terms in $\|\mathbf{w}_j(t+1) - \mathbf{w}^*\|$ is $\frac{C_4}{t+t_0}\left(\frac{1}{t_0} + \frac{1}{1+t_0} + \cdots + \frac{1}{t+t_0}\right)$. Therefore, using the harmonic series approximation, we conclude that the convergence rate is $\mathcal{O}\left(\frac{\log t}{t}\right)$. The above statement shows BRIDGE-T converges to the minimum of the global statistical risk at a sublinear rate, thus completing the proof of Theorem 2. ■

S.VI. Proof of Lemma 4

Under the stated assumptions of the lemma as well as the initialization for the nonconvex loss function, it is straightforward to see that (S.28) also holds for the nonconvex setting. In particular, the upper bound in (S.28) on $\|\mathbf{w}_j(t+1) - \mathbf{w}^*\|$ for all $j \in \mathcal{R}$ monotonically decreases with t. Thus, $\|\mathbf{w}_j(t+1) - \mathbf{w}^*\|$ for all $j \in \mathcal{R}, t \in \mathbb{R}$ can be upper bounded by the bound on $\|\mathbf{w}_j(1) - \mathbf{w}^*\|$, which is $C_1 + \frac{C_2(N)}{\lambda} + \frac{C_3}{t_0} + \frac{C_4}{t_0^2} + C_5$. The proof now follows from the fact that $C_1 := \|\mathbf{v}(0) - \mathbf{w}^*\| \leq \beta_0$ by virtue of the initialization. ■

A. Proof of Theorem 3

The local strong convexity of the loss function implies that $f(\mathbf{w}, \mathbf{z})$ can be treated as λ-strongly convex when restricted to the ball $\mathbb{B}(\mathbf{w}^*, \beta)$. Therefore, Assumption 3′, the Γ-boundedness of the iterates stated in the beginning of Section IV and Lemma 4, and the constraint $\beta \geq \Gamma$ imply that the proof of this theorem is a straightforward replication of the proof of Theorem 2 for the convex case. ■

References

[1] J. Peng, W. Li, and Q. Ling, "Byzantine-robust decentralized stochastic optimization over static and time-varying networks," *Signal Processing*, vol. 183, p. 108020, 2021.

[2] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *J. American Stat. Assoc.*, vol. 58, no. 301, pp. 13–30, 1963.

[3] J. Verger-Gaugry, "Covering a ball with smaller equal balls in R^n," *Discrete & Computational Geometry*, vol. 33, no. 1, pp. 143–155, 2005.