Adversarially Robust Zeroth-order ADMM Consensus over Networks

Xinyang Cao and Lifeng Lai

Abstract—**Due to the grow of data size, there is a recent surge of interest in the design of distributed machine learning algorithms that run with computers connected using networks. However, Byzantine attackers can easily compromise some computers and prevent the convergence of algorithms or lead the algorithm to converge to value of attackers' choice. In this paper, we design a zeroth order adversarially robust alternating direction method of multipliers (ZOAR-ADMM) that can deal with Byzantine attackers for the zeroth-order methods in a consensus network. The main idea of the algorithm is to ask each worker store a local deviation statistics of distance between neighbor's model parameter and its own model parameter for every neighbor. These information will then be used to filter out bad model parameter from Byzantine attackers. We show that this algorithm can converge to the sample minimizer and the function can converge to the optimal value. We further provide numerical examples to illustrate the performance of the proposed algorithm.**

I. Introduction

There are a growing number of applications that produce computation and storage challenges for machine learning systems. To address these challenges and to harness the computing power of multiple machines distrbuted over networks, there is a growing interest in the design of distributed optimization algorithms [1]–[11]. Various distributed methods have been proposed in many existing works to solve distributed optimization problems, such as belief propagation [1], distributed subgradient descent algorithms [2], dual averaging methods [3] and the alternating direction method of multipliers (ADMM) [4], [11]–[14] etc. Among these, ADMM has drawn significant attentions, since it is well suited for distributed optimization and shows fast convergence. These methods mainly explore first-order methods, i.e. use gradients of the loss function for iterative updates. However, in some scenarios such as simulation-based optimization, bandit optimization and objectives without simple gradient expressions etc [15]–[18], gradients are hard to be explicitly evaluated.

In order to make distributed methods work well for these applications, zeroth-order methods, which only use function values, have been proposed [5]–[10]. Most of the existing works, both the first-order and zeroth-order methods, assume that these workers behave honestly and follow the protocol. However, in practice, there is a risk that some of the workers are compromised. These compromised workers can prevent the convergence of the optimization algorithms or lead the algorithms to converge to values chosen by the attackers by modifying or falsifying intermediate results when the server require these intermediate results for updating. For example, as shown in [19], [20], for the first-order methods, the presence of even a single Byzantine worker can prevent the convergence of distributed gradient descent algorithms.

There have been some interesting recent works on designing distributed machine learning algorithms [19]–[30] that can deal with Byzantine attacks. The main idea of several works is to compare information received from all workers and compute a quantity that is robust to attackers for algorithm update. Another idea is to employ the redundant data when dealing with Byzantine attackers. In [23], Chen et al. proposed an algorithm named DRACO that uses redundant data. Each worker computes redundant gradients, encodes them and sends the resulting vector to the server. These vectors will pass through a decoder that detects where the adversaries are through the encoded redundant gradient information. However, these algorithms require the first-order gradient information.

In this paper, we focus on problems in which the first-order gradient information is difficult to obtain. In particular, we propose a new robust zeroth-order information based distributed optimization algorithm that is robust to Byzantine attacks. We name the method as zeroth-order adversarially robust alternating direction method of multipliers (ZOAR-ADMM). In the proposed method, at each iteration, each worker will first receive model parameter from its neighbors. Then each worker will test received parameter information by computing the distance from the received parameter to the model parameter computed using local data, and then sum all such distances obtained in history to build a deviation statistic for all neighbor workers. If the deviation statistic computed for its neighbor worker is smaller than a specially designed threshold, the worker will accept the model parameter from that neighbor. If the deviation statistic is larger than the threshold, the worker will reject the model parameter and decide that worker to be an attacker. After testing, each worker will first update dual variable by using accepted model parameters, then compute temporary model parameter based on accepted parameters and deterministic gradient approximation computed from its own data. It will then update new model parameter and broadcast it to its neighbors. We show that the proposed algorithm can solve the optimization problem and the objective function can converge to the minimum value. We show this result by first investigating how the distance between model parameter and optimal value is affected by the attack vector generated by the attackers, and then carefully analyzing how

Xinyang Cao and Lifeng Lai are with Department of Electrical and Computer Engineering, University of California, Davis, CA, 95616. Email: {xycao,lflai}@ucdavis.edu. This work was supported by the National Science Foundation under grant CCF-2232907.

the proposed testing method can mitigate these effects and eventually proving that the value of objective function of the proposed algorithm will converge to the optimal value despite the presence of Byzantine attackers.

II. Model

In this section, we introduce our model. For an unknown distribution $\mathcal{D}$, our goal is to infer the model parameter $\theta^* \in \Theta$ of the unknown distribution. It is popular to formulate this inference problem as an optimization problem

$$\theta^* \in \arg\min_{\theta\in\Theta} F(\theta) = \mathbb{E}\{f(X,\theta)\}, \tag{1}$$

in which X is the data generated by the unknown distribution $\mathcal{D}$, $f : \mathcal{X} \times \Theta \to \mathbb{R}$ is the loss function, $\Theta \in \mathbb{R}^d$ is a closed convex set of all possible model parameters, and the expectation is over the distribution $\mathcal{D}$. $F(\theta)$ is called population risk function.

Since the expectation in (1) is over the unknown distribution $\mathcal{D}$, the population risk function $F(\theta)$ is unknown and hence we cannot solve (1) directly. Instead, one typically aims to minimize the empirical risk:

$$\min_{\theta\in\Theta} \frac{1}{N}\sum_{s=1}^{N} f(X_s,\theta), \tag{2}$$

which uses N data samples $X_s, s = 1,\dots,N$ generated by the unknown distribution $\mathcal{D}$. By solving (2), we obtain an estimate of the true model parameter θ^*. When the number of data points N is large, we can employ distributed optimization methods. In particular, we consider a network consisting of n workers bidirectionally connected with E edges. We can describe the network as a symmetric directed graph $\mathcal{G}_d = \{\mathcal{V},\mathcal{A}\}$, where $\mathcal{V}$ is the set of workers with $|\mathcal{V}| = n$ and $\mathcal{A}$ is the set of directed edges with $|\mathcal{A}| = 2E$. In a distributed setup, a connected network of workers collaboratively minimize the sum of their local loss functions over a common optimization variable. Each worker generates local updates individually and communicates with its neighbors to reach a common minimizer in a consensus network. Then we can have a distributed optimization problem with population risk,

$$\min_{\theta_i,\phi_{ij}} \sum_{i=1}^{n} F^i(\theta_i), s.t. \theta_i = \phi_{ij}, \theta_j = \phi_{ij}, \forall (i,j) \in \mathcal{A}. \tag{3}$$

where $F^i(\theta_i) = \mathbb{E}\{f(X,\theta_i)\}$, where $f(X,\theta_i)$ represents the loss function based on the data generated by the unknown distribution $\mathcal{D}$ and the model parameter θ_i. $\theta_i \in \mathbb{R}^d$ is the local optimization variable at worker i and $\phi_{ij} \in \mathbb{R}^d$ is an auxiliary variable imposing the consensus constraint on neighbor workers i and j. Again, since we do not know the distribution $\mathcal{D}$, we cannot solve (3) directly. Instead we can focus on the distributed optimization problem for empirical risk function formulated as follows

$$\min_{\theta_i,\phi_{ij}} \sum_{i=1}^{n} \overline{f}^{(i)}(\theta_i), s.t. \theta_i = \phi_{ij}, \theta_j = \phi_{ij}, \forall (i,j) \in \mathcal{A}. \tag{4}$$

where $\overline{f}^{(i)}(\theta_i) = \frac{1}{|\mathcal{S}_i|}\sum_{s\in\mathcal{S}_i} f(X_s,\theta_i)$ with $\mathcal{S}_i$ being the set of data samples at worker i.

Define $\theta \in \mathbb{R}^{nd}$ as a vector concatenating all θ_i, $\phi \in \mathbb{R}^{2Ed}$ as a vector concatenating all ϕ_{ij}, then (4) can be written in a matrix form as

$$\begin{aligned} \min_{\theta,\phi} \quad & f(\theta) + \Gamma(\phi), \\ s.t. \quad & A\theta + B\phi = 0, \end{aligned} \tag{5}$$

where $f(\theta) = \sum_{i=1}^{n} \overline{f}^{(i)}(\theta_i)$ and $\Gamma(\phi) = 0$. Here $A = [A_1; A_2]; A_1, A_2 \in \mathbb{R}^{2Ed\times nd}$ are both composed of $2E \times n$ blocks of $d \times d$ matrices. If $(i,j) \in \mathcal{A}$ and ϕ_{ij} is the qth block of ϕ, then the (q,i)th block of A_1 and the (q,j)th block of A_2 are $d \times d$ identity matrices I_d; otherwise the corresponding blocks are $d \times d$ zero matrices 0_d. Also, we have $B = [-I_{2Ed}; -I_{2Ed}]$ with I_{2Ed} being a $2Ed \times 2Ed$ identity matrix.

In this paper, we assume that $F(\theta)$ and θ satisfy the following assumptions.

Assumption 1. *$F(\theta)$ is m_F-strongly convex and $F(\theta)$ has M_F-Lipschitz gradients on $\theta \in \Theta$ for any θ.*

Assumption 2. *The constrain set Θ is convex and compact, there exists some constant R such that $\|\theta - \theta'\| \le R$ for any $\theta, \theta' \in \Theta$.*

These assumptions are common assumptions in existing works for optimization problems [10], [31].

The iterative updates of the distributed ADMM to solve problem (4) is given in [11]. In particular, consider the augmented Lagrangian of (5), we will have

$$L(\theta,\phi,\nu) = f(\theta) + \langle \nu, A\theta + B\phi\rangle + \frac{c}{2}\|A\theta + B\phi\|^2. \tag{6}$$

By using ADMM method, the updates are

$$\begin{aligned} \nabla f(\theta^{k+1}) + A^T\nu^{k+1} + cA^TB(\phi^k - \phi^{k+1}) &= 0, \\ B^T\nu^{k+1} &= 0, \\ \nu^{k+1} - \nu^k - c(A\theta^{k+1} + B\phi^{k+1}) &= 0. \end{aligned} \tag{7}$$

By letting $\nu = [\beta;\gamma]$ with $\beta,\gamma \in \mathbb{R}^{2Ed}$ and recalling $B = [-I_{2Ed}; -I_{2Ed}]$, we will have $\gamma = -\beta$. By choosing $\phi^0 = \frac{1}{2}M_+^T\theta^0$, the ADMM form will be reduced to the following form:

$$\begin{aligned} &\theta - update : \nabla f(\theta^{k+1}) + M_-\beta^{k+1} - \frac{c}{2}M_+M_+^T\theta^k \\ &\quad + \frac{c}{2}M_+M_+^T\theta^{k+1} = 0, \\ &\beta - update : \beta^{k+1} - \beta^k - \frac{c}{2}M_-^T\theta^{k+1} = 0, \end{aligned} \tag{8}$$

where $\beta \in \mathbb{R}^{2Ed}$, the matrices $M_+ = A_1^T + A_2^T$ and $M_- = A_1^T - A_2^T$. Let $W \in \mathbb{R}^{nd\times nd}$ be a block diagonal matrix with its (i,i)th block being the degree of agent i multiplying I_d and other blocks being 0_d, $L_+ = \frac{1}{2}M_+M_+^T, L_- = \frac{1}{2}M_-M_-^T$, and $W = \frac{1}{2}(L_+ + L_-)$. By defining a new multiplier $\alpha =$

$M\_\beta \in \mathbb{R}^{nd}$, the algorithm reduces to the following form:

$$\theta - update: \nabla f(\theta^{k+1}) + \alpha^k + 2cW\theta^{k+1} = cL_+^{k+1}\theta^k,$$
$$\alpha - update: \alpha^{k+1} - \alpha^k - cL_-^{k+1}\theta^{k+1} = 0. \quad (9)$$

Note $\theta = [\theta_1, ...\theta_n]$, $\alpha = [\alpha_1, ..., \alpha_n] \in \mathbb{R}^{nd}$, and there is an optimal solution $\theta^* \in \Theta$. These matrices are related to the underlying network topology. From above, we can find that W is a block diagonal matrix with its (i,i)th being the number of neighbor of worker i. L_- is the Laplacian matrix, and L_+ is the nonnegative Laplacian matrix.

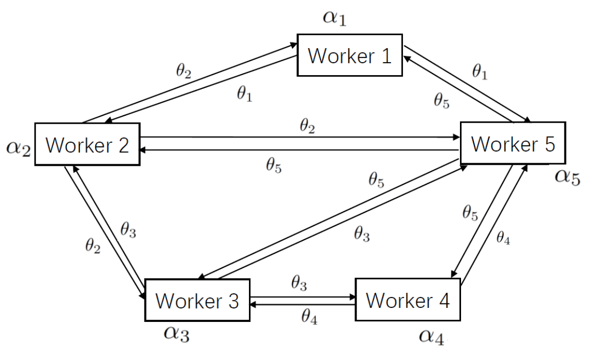

Fig. 1. Information flow of ADMM algorithm in [11].

Using the matrices defined above, the matrices form iterative updates in (9) can be distributed to each worker. For example, Figure 1 illustrates information flow of 5 workers in the network by using this algorithm. In iteration k, worker i will receive all model parameter $\theta_j^k, j \in \mathcal{N}_i$ from its neighbors, then it will first calculate α_i^k based on received information:

$$\alpha_i^k = \alpha_i^{k-1} + c|\mathcal{N}_i|\theta_i^k - c\sum_{j\in\mathcal{N}_i}\theta_j^k. \quad (10)$$

Then it will update θ_i^{k+1} by solving

$$\nabla \overline{f}^{(i)}(\theta_i^{k+1}) + \alpha_i^k + 2c|\mathcal{N}_i|\theta_i^{k+1} = c|\mathcal{N}_i|\theta_i^k + c\sum_{j\in\mathcal{N}_i}\theta_j^k, \quad (11)$$

based on received model information θ_j^k and local data. After updating θ_i^{k+1}, worker i will broadcast it to its all neighbors. Algorithm 1 (from [11]) summarizes these steps.

In this paper, we consider two problems based on Algorithm 1. First, we consider a system with Byzantine attackers, in which an unknown subset of workers might be compromised. In each iteration, compromised worker i can send arbitrary information to its neighbors. Let $\mathcal{B}$ denote the set of compromised workers. Then we can write the information sent by node i as $z_i = \theta_i + e_i$ with e_i taking the following form

$$e_i = \begin{cases} 0 & i \notin \mathcal{B} \\ \star & i \in \mathcal{B} \end{cases} \quad (12)$$

in which $\star$ denotes an arbitrary vector chosen by the attacker. Secondly, We also consider the system where gradient or subgradient information is hard to be explicitly evaluated. Instead, we will use a deterministic estimator $g_i(\theta_i)$ to estimate $\nabla \overline{f}^{(i)}(\theta_i)$, which approximates each coordinate of the gradient and then sums them up [32]:

$$g_i(\theta_i) = \frac{1}{m}\sum_{s\in\mathcal{S}_i}\sum_{l=1}^{d}\frac{f(X_s, \theta_i + uv_l) - f(X_s, \theta_i - uv_l)}{2u}v_l.$$

Here u is a scalar, whose value will be specified in the algorithm analysis, and v_l is a standard basis vector with 1 at its lth coordinate.

Then the corresponding algorithm becomes

$$g_i(\theta_i^{k+1}) + \alpha_i^k + 2c|\mathcal{N}_i|\theta_i^{k+1} = c|\mathcal{N}_i|z_i^k + c\sum_{j\in\mathcal{N}_i}z_j^k,$$
$$\alpha_i^{k+1} = \alpha_i^k + c|\mathcal{N}_i|z_i^{k+1} - c\sum_{j\in\mathcal{N}_i}z_j^{k+1}. \quad (13)$$

For a clearer presentation, we will use following equivalent form of the updates in analysis when there are Byzantine attackers:

$$\theta - update: g(\theta^{k+1}) + \alpha^k + 2cW^{k+1}\theta^{k+1} = cL_+^{k+1}z^k,$$
$$\alpha - update: \alpha^{k+1} - \alpha^k - cL_-^{k+1}z^{k+1} = 0, \quad (14)$$

where $g(\theta) = \sum_{i=1}^n g_i(\theta_i)$. Compared with (9), θ^k is replaced by z^k and θ^{k+1} is replaced by z^{k+1}. The goal of our paper is to design robust zeroth-order algorithms, by designing proper tests for each worker that can tolerate Byzantine attacks. For $g(\theta)$ generated by deterministic estimator, we will use $g(\theta)$ to estimate $\nabla f(\theta)$. For $\nabla f(\theta)$, we have following assumption, which are similar to those used in [19], [22], [24],

Assumption 3. *There exist positive constants σ_1 and α_1 such that for any unit vector $v \in B$, $\langle \nabla f(X, \theta^*), v\rangle$ is sub-exponential with σ_1 and α_1, that is,*

$$\sup_{v\in B}\mathbb{E}[\exp(\lambda\langle\nabla f(X,\theta^*), v\rangle)] \leq e^{\sigma_1^2\lambda^2/2}, \forall|\lambda| \leq 1/\alpha_1,$$

where B denotes the unit sphere $\{v : \|v\|_2 = 1\}$.

We now define gradient difference $w(X,\theta) = \nabla f(X,\theta) - \nabla f(X,\theta^*)$ and assume that for every θ, $w(X,\theta)$ normalized by $\parallel \theta - \theta^* \parallel$ is also sub-exponential.

Assumption 4. *There exist positive constants σ_2 and α_2 such that for any $\theta \in \Theta$ with $\theta \neq \theta^*$ and any unit vector $v \in B$, $\langle w(X,\theta) - \mathbb{E}[w(X,\theta)], v\rangle / \parallel \theta - \theta^* \parallel$ is sub-exponential with σ_2 and α_2, that is,*

$$\sup_{\theta\in\Theta, v\in B}\mathbb{E}\left[\exp\left(\frac{\lambda\langle w(X,\theta) - \mathbb{E}[w(X,\theta)], v\rangle}{\|\theta - \theta^*\|}\right)\right]$$
$$\leq e^{\sigma_2^2\lambda^2/2}, \ \forall|\lambda| \leq \frac{1}{\alpha_2}. \quad (15)$$

This allows us to show that $\frac{1}{m}\sum_{s\in\mathcal{S}_s} w(X_s,\theta)$ concentrates on $\mathbb{E}[w(X,\theta)]$ for every fixed θ.

Assumptions 3 and 4 ensure that random gradient $\nabla f(\theta)$ has good concentration properties, i.e., an average of m $i.i.d$ random gradients $\frac{1}{m}\sum_{s\in\mathcal{S}_s}\nabla f(X_s,\theta)$ sharply concentrates on $\nabla F(\theta)$ for every fixed θ, which is an assumption on the

Algorithm 1: ADMM [11]

```
  Initialize $\theta^1 = 0, c, \alpha^0 = 0, T$.
  for $k = 1$ to $T$ do
    For the worker $i$:
    1: Receives the model parameter $\theta_j^k$ from its neighbor;
    2: Computes $\alpha_i^k = \alpha_i^{k-1} + c|\mathcal{N}_i|\theta_i^k - c\sum_{j\in\mathcal{N}_i}\theta_j^k$
    3: Solves $\nabla f_i(\theta_i^{k+1}) + \alpha_i^k + 2c|\mathcal{N}_i|\theta_i^{k+1}$
    $= c|\mathcal{N}_i|\theta_i^k + c\sum_{j\in\mathcal{N}_i}\theta_j^k$
    to gets updated $\theta_i^{k+1}$ and communicates it with its
    neighbors;
  end for
output $\theta^T$.
```

Algorithm 2: ZOAR-ADMM

```
  Initialize $\theta^1 = 0, c, \alpha^0 = 0, T, U$.
  for $k = 1$ to $T$ do
    For the worker $i$:
    1: Receives the model parameter $\theta_j^k$ from its neighbor;
    if $\sum_{t=1}^k \|\theta_i^t - \theta_j^t\| > U$ then
      2: worker $i$ detects that worker $j$ is an attacker,
      rejects $\theta_j^k$ and removes worker $j$ from $\mathcal{N}_i^k$;
    else
      2: worker $i$ accepts $\theta_j^k$;
    end if
    3: Computes $\alpha_i^k = \alpha_i^{k-1} + c|\mathcal{N}_i^k|\theta_i^k - c\sum_{j\in\mathcal{N}_i^k}\theta_j^k$
    4: Solves $g_i(\theta_i^{k+1}) + \alpha_i^k + 2c|\mathcal{N}_i^k|\theta_i^{k+1}$
    $= c|\mathcal{N}_i^k|\theta_i^k + c\sum_{j\in\mathcal{N}_i^k}\theta_j^k$
    to gets updated $\theta_i^{k+1}$ and communicates it with its
    neighbors;
  end for
output $\theta^T$.
```

upper bound of the variance of the gradient.

We also assume data in each worker has following assumption.

Assumption 5. *For any $\delta \in (0, 1/m)$, there exists an $M_f = M_f(\delta)$ and $m_f = m_f(\delta)$ such that*

$$\mathbf{Pr}\left\{\forall \theta, \theta' \in \Theta, m_f \le \frac{\|\nabla f(X,\theta) - \nabla f(X,\theta')\|}{\|\theta - \theta'\|} \le M_f\right\} \ge 1 - \frac{\delta}{3}. \quad (16)$$

Assumption 5 ensures that $\nabla f(X,\theta)$ in each worker is M_f-Lipschitz and $f(X,\theta)$ is m_f strongly convex with high probability.

III. Algorithm

In this section, we describe our algorithm in distributed network that can tolerate Byzantine attacks in ADMM updates.

If there is no network, each worker will compute model parameter by itself, then in each iteration, different workers will have different model parameter. But in a network, workers will communicate with its neighbor, then each worker can know the model parameter deviation between itself and its neighbor. The main idea of our algorithm is to use this model parameter deviation to detect Byzantine attackers. As we will show in Lemma 4, for the case where all the workers are honest, the deviation statistic $\sum_{t=1}^k \sum_{(i,j)\in\mathcal{A}} \|\theta_i^t - \theta_j^t\|$ will be bounded by a quantity value U no matter what the value k is. As the result, this bound can serve as the standard threshold for each worker to decide whether its neighbor is honest or not. Inspired by this bound, in our algorithm, each worker maintains the local deviation statistic $\sum_{t=1}^k \|\theta_i^t - \theta_j^t\|$ for every neighboring worker j, and compares it with U to test if neighboring worker j provides a reasonable value or not. The local deviation statistic from an honest worker will always smaller than U, no matter how many iterations have passed.

In particular, in iteration k, worker i tests all the model information θ_j^k from its neighbor $j, j \in \mathcal{N}_i$. If the local deviation statistic $\sum_{t=1}^k \|\theta_i^t - \theta_j^t\|$ from neighbor j is larger than U, neighbor j will be considered as a Byzantine attacker. The model parameter sent by a Byzantine attacker will be rejected forever and worker i will not send information to worker j. Worker j will be removed from set $\mathcal{N}_i$ and worker i will be removed from set $\mathcal{N}_j$. Then worker i and worker j will have new neighbor set $\mathcal{N}_i^k$ and $\mathcal{N}_j^k$. After testing all neighbors, worker i updates α_i^k first:

$$\alpha_i^k = \alpha_i^{k-1} + c|\mathcal{N}_i^k|\theta_i^k - c\sum_{j\in\mathcal{N}_i^k}\theta_j^k. \quad (17)$$

Then worker i will update θ_i by solving

$$g_i(\theta_i) + \alpha_i^k + 2c|\mathcal{N}_i^k|\theta_i = c|\mathcal{N}_i^k|\theta_i^k + c\sum_{j\in\mathcal{N}_i^k}\theta_j^k, \quad (18)$$

where we use deterministic gradient estimator $g_i(\theta_i)$ using its own local m data samples:

$$g_i(\theta_i) = \frac{1}{m}\sum_{s\in\mathcal{S}_i}\sum_{l=1}^d \frac{f(X_s, \theta_i + u_k v_l) - f(X_s, \theta_i - u_k v_l)}{2u_k} v_l.$$

In our algorithm, at iteration k, we will choose $u_k = \frac{1}{dk^2}$. After worker i update θ_i, it will communicate its value with its neighbors. Main steps of the algorithm are list in Algorithm 2.

IV. Convergence Analysis

Before presenting detailed analysis, here we introduce some notations for the network and describe the high level ideas. On iteration k, when we describe the network, we let $Q^k = LD^{\frac{1}{2}}L^T$, where $LDL^T = \frac{L_-^k}{2}$ is the singular value decomposition of the positive semidefinite matrix $\frac{L_-^k}{2}$, and L_-^k represents the Laplacian matrix of the network at iteration k. We will define a new auxiliary sequence $r^k = \sum_{s=0}^k Q^s(\theta^s + e^s)$ to represent the accumulation of the network constraint in optimization problem over iterations. In addition, we define matrix p and matrix G as

$$p^k = \begin{bmatrix} r^k \\ \theta^k \end{bmatrix}, G^{k+1} = \begin{bmatrix} cI & 0 \\ 0 & cL_+^{k+1}/2 \end{bmatrix}. \quad (19)$$

We also define two constants that will be used in the analysis:

$$\Delta_1 = \sqrt{2}\sigma_1\sqrt{(d\log 6 + \log(3/\delta))/m}, \quad (20)$$
$$\Delta_2 = \sqrt{2}\sigma_2\sqrt{(\tau_1+\tau_2)/m} \quad (21)$$

with $\tau_1 = d\log 18 + d\log(M_F \vee M_f/\sigma_2)$, $\tau_2 = 0.5d\log(m/d) + \log(6/\delta) + \log(\frac{2r\sigma_2^2\sqrt{m}}{\alpha_2\sigma_1})$.

In our analysis, we will first study the properties of the zeroth-order gradient estimation at an honest worker. We will then analyze the impacts of attacks on each iteration of ADMM. Finally, we will show that our proposed algorithm can reduce the error caused by Byzantine attackers and the function value will converge to the function value based on the optimal parameter.

A. Bound of zeroth-order gradient estimation

In this section, we will derive an upper bound on the gradient estimate at an honest worker. This bound will be used in the subsequent analysis.

Recall that we have $f(\theta) = \sum_{i=1}^{n} \overline{f}^{(i)}(\theta_i)$. To consider the difference between zeroth-order gradient estimation and the true unknown gradient of $f(\theta)$, we denote $h(\theta) = \nabla f(\theta) - g(\theta)$. For $h(\theta)$, we have

Lemma 1. *([32]) Under Assumptions 1, 2, 5, in iteration k, for any $\delta \in (0,1)$, with probability at least $1-\delta/3$, the deterministic estimator $g(\theta^k)$ satisfies*

$$\|g(\theta^k) - \nabla f(\theta^k)\|^2 \le nM_f^2 d^2 u_k^2/(4m). \quad (22)$$

Lemma 1 illustrates that there is a bound for the distance between zeroth-order estimate and the true gradient. From this lemma and assumptions mentioned above, we have the following upper bound on $\|g_i(\theta)\|$.

Lemma 2. *Under Assumptions 1-5, in iteration k, for any $\delta \in (0,1)$, with probability at least $(1-\delta)$, the deterministic estimator $g_i(\theta_i^k)$ satisfies*

$$\|g_i(\theta_i^k)\| \le V_k + M_f\|\theta_i^k - \theta^*\|, \quad (23)$$

where $V_k = \frac{M_f^2 d^2 u_k^2}{m} + \Delta_1$.

B. Impact of Byzantine attackers in ADMM

In this section, we analyze the impact of Byzantine attacks on the iterations of ADMM. To facilitate the analysis of the algorithm, we show that the algorithm has the following equivalent form.

Lemma 3. *The algorithm satisfies*

$$g(\theta^{k+1}) = 2cW^{k+1}e^{k+1} - cL_+^{k+1}(z^{k+1}-z^k) - 2cQr^{k+1},$$

where $W^{k+1} = \frac{L_+^{k+1}+L_-^{k+1}}{2}$ and Q is a matrix that makes $2Qr^{k+1} = \sum_{s=0}^{k+1} L_-^s(\theta^s + e^s)$

Using this lemma, we are ready to show that, if each node blindly accepts information from neighboring workers, Byzantine attackers can change the distance between θ^k and θ^* by changing the model parameter during information transmission.

Theorem 1. *If Assumptions 1-5 hold, by choosing $u_k = \frac{1}{dk^2}$ for k iteration, for any $\delta \in (0,1)$, with optimal value*

$$p = \begin{bmatrix} 0 \\ \theta^* \end{bmatrix}, \quad (24)$$

then with probability at least $(1-\delta)^n$, we have

$$\|p^{k+1}-p\|_{G^{k+1}}^2 \le \frac{1}{1+\rho}\left(\|p^k - p\|_{G^{k+1}}^2 + \Delta(k+1)\right), \quad (25)$$

where

$$\begin{aligned}\Delta(k+1) &= c\frac{\sigma_{max}^2(L_+^{k+1})}{2\sigma_{min}(L_-^{k+1})}\|e^k\|^2 + \frac{\sqrt{n}M_f R}{\sqrt{m}k^2} + \Delta_1 R \\ &\quad + c^2\sigma_{max}^2(L_+^{k+1})\|e^k\|^2 + c^2\sigma_{max}^2(L_-^{k+1})\|e^{k+1}\|^2 \\ &\quad + c\langle e^{k+1}, 2Qr^{k+1}\rangle + 2(\mu-1)nV_{k+1}^2 + 8\Delta_2 R^2, \quad (26)\end{aligned}$$

and

$$\rho = \min\left\{\frac{(\mu-1)\sigma_{min}^2(L_-^{k+1})}{2\mu\sigma_{max}^2(L_+^{k+1})\sigma_{max}(L_-^0)}, \frac{m_f}{\frac{c\sigma_{max}^2(L_+)}{2} + \frac{\mu}{c}2M_f^2\sigma_{min}^{-2}(L_-^{k+1})\sigma_{max}(L_-^0)}\right\} > 0. (27)$$

From this theorem, we can see that when there is no attacker, i.e., $\|e^k\| = \|e^{k+1}\| = 0$, then $\Delta(k+1)$ decreases and goes to $2(\mu-1)\Delta_1^2 + \Delta_1 R + 8\Delta_2 R^2$ as $k \to \infty$, which is generated from the approximation of population risk function by using empirical risk function. We can find the sequence $\|p^k - p\|_{G^k}^2$ converges linearly to the neighbor of optimal p with a rate of $\frac{1}{1+\rho}$ when there is no attacker in the network. However, when there are attackers, this theorem shows how the error values $\|e^k\|$ introduced by the attackers affect the term $\Delta(k+1)$, and these errors will accumulate after each iteration. These error values can be any value decided by the Byzantine attackers. The bound will become larger and larger, the ADMM algorithm will not converge.

To provide further insights on how attackers can impact the algorithm, we analyze how the convergence rate is related to the value of ρ. In the no attacker case, by maximizing ρ, we can have a better convergence result. Then we will show how to maximal ρ.

Proposition 1. *If the algorithm parameter c is chosen as*

$$c = \frac{2M_f\sqrt{\sigma_{max}(L_-^0)\sqrt{\mu}}}{\sigma_{max}(L_+^{k+1})\sigma_{min}(L_-^{k+1})}, \quad (28)$$

and

$$\mu = 1 + \frac{K_L^2 \sigma_{max}(L_-^0)}{K_f^2} - \frac{K_L \sigma_{max}(L_-^0)}{2K_f}\sqrt{\frac{8}{\sigma_{max}(L_-^0)} + 4\frac{K_L^2}{K_f^2}},$$

then we have

$$\rho = \frac{1}{2K_f}\sqrt{\frac{8}{\sigma_{max}(L_-^0)K_{L^{k+1}}^2} + \frac{4}{K_f^2}} - \frac{1}{2K_f^2} \qquad (29)$$

maximizes the value of ρ in iteration $k+1$, where $K_{L^{k+1}} = \frac{\sigma_{max}(L_+^{k+1})}{\sigma_{min}(L_-^{k+1})}$ and $K_f = \frac{M_f}{m_f}$.

The minimum non-zero singular value of the signed Laplacian matrix L_- and the maximum singular value of signless Laplacian matrix L_+ are related to network connectedness but former is less. Roughly speaking, larger L_+ and L_- mean stronger connectedness, and a larger K_L means weaker connectedness. From this proposition, we can observe that the value of ρ is related to K_L. The value of ρ decreases as K_L increases. This proposition suggests that another way that the Byzantine attacker can influence the result is to reduce the network connectedness, which makes the convergence arbitrarily slow.

In summary, Theorem 1 and Proposition 1 provide useful insights the impact the adversarial attacks. In particular, when we consider the defending method as in the proposed ZOAR-ADMM, we are going to identify the Byzantine attackers and remove them from the network. Then in the network, the attackers may have two difference methods for attacking: 1) From insights in Theorem 1, the attacker may choose to make small changes at each step so that changed model parameter pass the test and workers will accumulate the wrong information; 2) From insights in Proposition 1, the attacker may choose to make large changes to the value so it does not pass the test, which will break the network and change the value of ρ and impact the convergence.

C. Convergence analysis of ZOAR-ADMM

Using the insights obtained in Section IV-B, in this section, we will prove the convergence of ZOAR-ADMM when there are Byzantine attackers in the network.

In Section III, we mention that, when there is no Byzantine attackers, the deviation statistic $\sum_{t=1}^{k}\|Q\theta^t\|$ will be bounded by some value no matter what the value k. The following lemma shows how to find such a bound.

Lemma 4. *Consider a network without attacker, starting from $\theta^0 = 0$ and $u_t = \frac{1}{dt^2}$, for any $\delta \in (0,1)$, with probability at least $(1-\frac{\delta}{3})^n$, we have*

$$\frac{1}{T}\sum_{t=1}^{T}\|Q\theta^t\| \leq \frac{1}{4T}\left(\sigma_{max}(L_+^0)R^2 + \frac{4C}{\sigma_{min}(L_-^0)c^2} + 4\right) + \frac{R}{2cT}\frac{\sqrt{n}M_f\pi^2}{12\sqrt{m}}, \qquad (30)$$

where $C = nV_1^2 + M_f^2R^2$.

Using this lemma, we can set the bound for testing as $U = \frac{1}{2\sqrt{2}}\left(\sigma_{max}(L_+^0)R^2 + \frac{4C}{\sigma_{min}(L_-^0)c^2} + 4\right) + \frac{R}{c}\frac{\sqrt{n}M_f\pi^2}{12\sqrt{2m}}$. When there is no attacker, from Lemma 4, $\sum_{t=1}^{T}\|Q\theta^t\| \leq U/\sqrt{2}$. Note that $\sum_{t=1}^{T}\|Q\theta^t\| = \frac{1}{\sqrt{2}}\sum_{t=1}^{T}\sum_{(i,j)\in\mathcal{A}}\|\theta_i^t - \theta_j^t\|$, thus, we will have $\frac{1}{\sqrt{2}}\sum_{t=1}^{T}\|\theta_i^t - \theta_j^t\| \leq U/\sqrt{2}, \forall(i,j)\in\mathcal{A}$. Then we can design our attacker testing method in the following way: in each iteration k, each worker i maintains the local deviation statistics $\sum_{t=1}^{k}\|\theta_i^t - \theta_j^t\|$ for every neighbor worker $j \in \mathcal{N}_i$. For an honest worker, this deviation statistics will not exceed U. If this value is greater than U, then worker j will be regarded as a Byzantine attacker by worker i, since if in one iteration, this value is greater than U, then after this iteration, the value will still be greater, so worker i will reject the information from worker j forever.

Next, we show that the proposed ZOAR-ADMM algorithm can converge to the optimal value in a consensus network. Considering after T iteration, the whole consensus network has been attacked to several small consensus networks. Assume first $\hat{n} \leq n$ workers are in one consensus network. Then consider the initial network between these workers, we will have $\hat{L}_+$, $\hat{L}_-$ for such network and $\hat{f}(\theta) = \sum_{i=1}^{\hat{n}}\overline{f}^{(i)}(\theta_i)$. Then we have the following theorem showing the proposed algorithm can work in a consensus network.

Theorem 2. *If Assumptions 1-5 holds, there exists optimal $p = \begin{bmatrix} r \\ \theta^* \end{bmatrix}$, with $r = 0$ and $\hat{\theta}_T = \frac{\sum_{k=1}^{T}\theta^k}{T}$, with $u_k = \frac{1}{dk^2}$ and for any $\delta \in (0,1)$, with probability $(1-\delta)^n$, it holds*

$$\hat{f}(\hat{\theta}_T) - \hat{f}(\theta^*) \leq \frac{1}{T}\left(\|\hat{p}^0 - p\|_{\hat{G}^1}^2 + c\frac{\sigma_{max}^2(\hat{L}_+^T)}{\sigma_{min}^2(\hat{L}_-^T)}8E^2U^2 + \frac{\pi^2}{6}\frac{\hat{n}\sqrt{\hat{n}}M_fR}{2n\sqrt{m}}\right). \qquad (31)$$

This theorem shows that, when the whole network is separated by Byzantine attackers into several smaller network, ZOAR-ADMM can work in each small consensus network. Now we consider the convergence of ZOAR-ADMM in the whole network. Consider different network in whole algorithm, for signless Laplacian matrix, we have $\|x^k - x^*\|_{\frac{L_+^k}{2}}^2 = \frac{1}{4}\sum_{i=1}^{m}\sum_{j\in\mathcal{N}_i}\|x_i - x^* + x_j - x^*\|^2$. Now consider the whole network, define $f_{all}(x) = \sum f(x) = \sum_{i=1}^{n} f_i(x_i)$, which consider the whole network, then we get the following theorem for whole network.

Theorem 3. *If Assumptions 1-5 holds, there exists optimal $p = \begin{bmatrix} r \\ \theta^* \end{bmatrix}$, with $r = 0$ and $\hat{\theta}_T = \frac{\sum_{k=1}^{T}\theta^k}{T}$, with $u_k = \frac{1}{dk^2}$ and for any $\delta \in (0,1)$, with probability $(1-\delta)^n$, it holds*

$$\begin{aligned} f(\hat{\theta}_T) - f(\theta^*) &= \sum \hat{f}(\hat{\theta}_T) - \hat{f}(\theta^*) \\ &\leq \frac{1}{T}\left(\|p^0 - p\|_{G^1}^2 + c\frac{\sigma_{max}^2(L_+^T)}{\sigma_{min}^2(L_-^T)}8E^2U^2 + \frac{\pi^2}{6}\frac{\sqrt{n}M_fR}{2\sqrt{m}}\right). \end{aligned} \qquad (32)$$

This theorem shows that the algorithm achieves a sublinear convergence rate of $\mathcal{O}(\frac{1}{T})$. The upper bound in (32) introduces two additional terms. The first term comes from the method for defending against Byzantine attackers and the second term comes from the estimate gradient by using zeroth-order approximation.

V. Numerical results

A. Synthesized data

We first use synthesized data. In this example, we focus on linear regression, in which

$$Y_i = H_i^T x^* + \epsilon_i, i = 1, 2, \cdots, N,$$

where $H_i \in \mathbb{R}^d$, x^* is a $d \times 1$ vector and ϵ_i is the noise. We set $\mathbf{H} = [H_1, \cdots, H_N]$ as $d \times N$ data matrix.

In the simulation, we set the dimension $d = 10$, the total number of data $N = 50000$. We use $\mathcal{N}(0, 9)$ to independently generate true model parameter x^*, where $\mathcal{N}(\nu, \sigma^2)$ denotes Gaussian variables with mean ν and variance σ^2. After x^* is generated, we fix it. The data matrix $\mathbf{H}$ is generated randomly by Gaussian distribution with $\nu = 0$ and fixed known maximal and minimal eigenvalues of the correlation matrix $\mathbf{H}^T\mathbf{H}$. Let $\lambda_{max}(\cdot)$ and $\lambda_{min}(\cdot)$ denote the maximal and minimal eigenvalue of $\mathbf{H}^T\mathbf{H}$ respectively. In the following figures, we use $\lambda_{max}(\mathbf{H}^T\mathbf{H}) = 100$ and $\lambda_{min}(\mathbf{H}^T\mathbf{H}) = 1$ to generate the data matrix $\mathbf{H}$. We set the white noise ϵ_i as i.i.d. $\mathcal{N}(0, 1)$ random variable. Finally, we generate Y_i using the linear relationship mentioned above. In the synthesized data simulation, we set the number of workers $n = 100$, and data are evenly distributed in each worker. The original network is generated by a connected Erdos-Renyi graph $ER(100, 0.2)$, meaning that 100 workers connect with each other with probability 0.2. We first randomly select 20 workers to be attackers. We illustrate our results with 2 different cases: 1) 20 Inverse attack, in which each attacker first calculates the gradient based on its local data but sends the inverse version of gradient information or vector information to the server; 2) 20 Random attack, in which the attacker randomly generates gradient value. In our simulation, we compare 2 algorithms: 1) The proposed ZOAR-ADMM as presented in Algorithm 2; 2) The DS-ADMM in [10] which considers zeroth-order ADMM with two times communication in each iteration.

Figures 2 and 3 plot the value of the average optimality gap vs iteration with 20 inverse attacks and 20 random attacks respectively, where the average optimality gap is defined as: $\frac{1}{n}\sum_{j=1}^{n}[\sum_{i=1}^{n} f_i(x_j^k) - \sum_{i=1}^{n} f_i(x^*)]$. From Figures 2 and 3, we can see that DS-ADMM method does not converge, since computing average cannot defend Byzantine attacks. On the other hand, the proposed ZOAR-ADMM can still converge, since it helps workers to detect the Byzantine attackers and converge under the trusted sub network.

Figures 4 and 5 plot the value of $\|Q^0 x^k\|^2$ vs iteration with 20 random attacks and 20 inverse attacks respectively. As discussed above, $\|Q^0 x^k\|^2$ can be used to show the node disagreement. From Figures 4 and 5, we can observe that

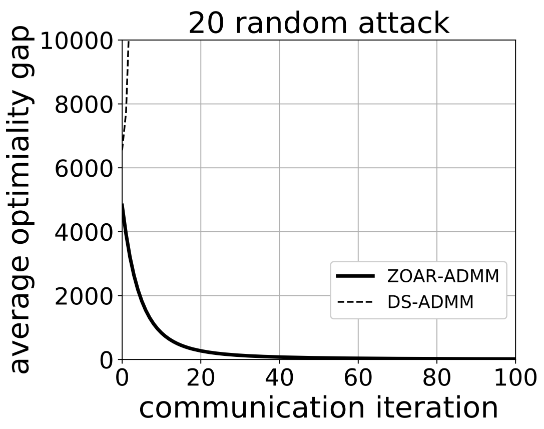

Fig. 2. Optimality gap comparison using synthesized data: 20 Random attack.

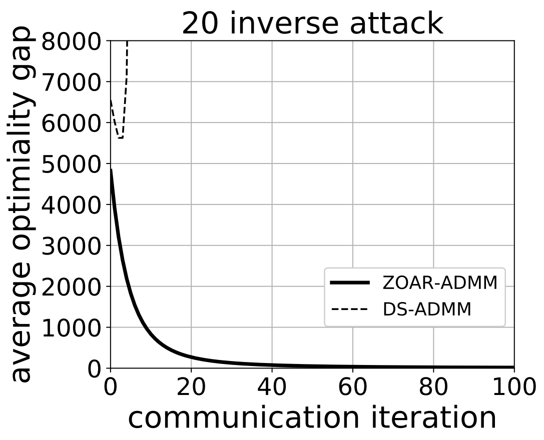

Fig. 3. Optimality gap comparison using synthesized data: 20 Inverse attack.

DS-ADMM has a large disagreement, since the attackers successfully make the algorithm fail. However the proposed ZOAR-ADMM has a small disagreement.

B. Real data

Now we test our algorithms on real datasets MNIST [33] and compare our algorithms with the existing zeroth order method in [10]. MNIST is a widely used computer vision dataset that consists of 70,000 28×28 pixel images of handwritten digits 0 to 9. We use the handwritten images of 3 and 5, which are the most difficult to distinguish in this dataset, to build a logistic regression model. After picking all 3 and 5 images from the dataset, the total number of images is 13454. It is divided into a training subset of size 12000 and a testing subset of size 1454. For the dataset, we set the number of workers to be 50, and generate network by a connected Erdos-Renyi graph $ER(50, 0.2)$. We then randomly select 20 workers from these 50 workers to be attackers. Similar to the synthesized data scenario, we illustrate our results with two cases, namely 20 inverse attack, 20 random attack, and compare the performance of two algorithms by comparing

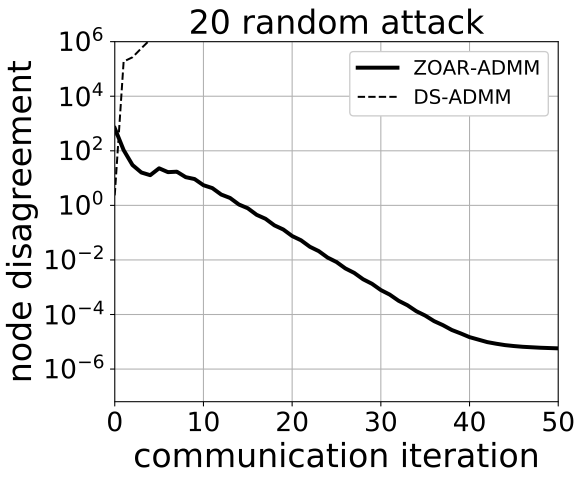

Fig. 4. Node disagreement comparison using synthesized data: 20 Random attack.

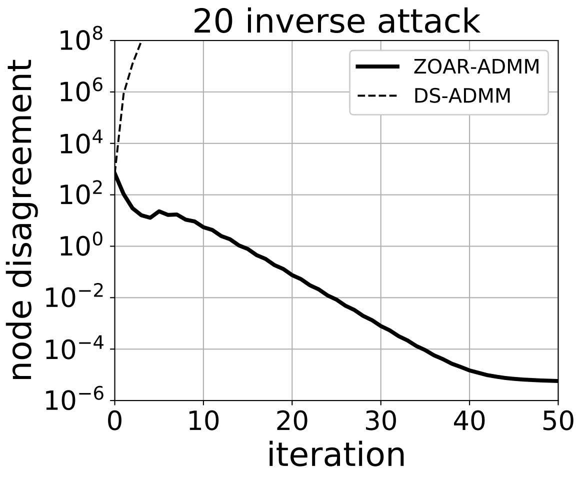

Fig. 5. Node disagreement comparison using synthesized data: 20 Inverse attack.

the testing accuracy and node disagreement. When testing accuracy, we consider $\overline{x} = \frac{1}{50}\sum_{i=1}^{50} x_i$ to be the output testing model parameter and testing with testing data. The following figures show the result.

Figures 6 and 7 illustrate the impact of two cases on different algorithms using MNIST respectively. Figures 6 and 7 show that the DS-ADMM fails to predict if there are 20 attackers. On the other hand, the proposed ZOAR-ADMM algorithm still show high accuracy.

We then plot the impact of 20 attackers case on real data with value of $\|Q^0 x^k\|^2$ to show node disagreement in Figures 8 and 9 using MNIST respectively. When there are 20 attackers, DS-ADMM has large disagreement, it cannot properly work. Our proposed ZOAR-ADMM has a low disagreement. As the iterations increase, the simulation result shows that our proposed ZOAR-ADMM has better accuracy and lower disagreement.

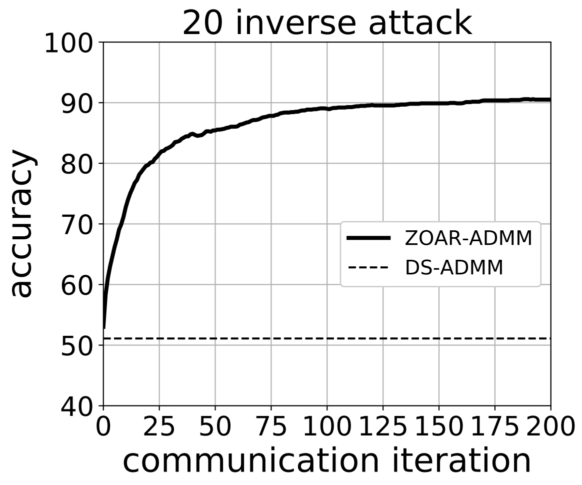

Fig. 6. Accuracy comparison using MNIST: 20 Inverse attack.

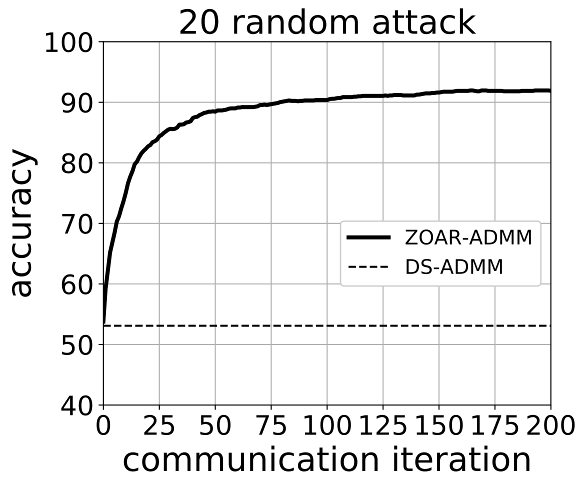

Fig. 7. Accuracy comparison using MNIST: 20 Random attack.

VI. Conclusion

In this paper, we have proposed a robust zeroth-order ADMM named ZOAR-ADMM algorithm that can tolerate Byzantine attackers in a distributed network. We have analyzed the effect of Byzantine attacks, and have proved that the proposed algorithm can converge to optimal value. We also have provided numerical examples to illustrate the performance of the proposed algorithm.

References

[1] J. Predd, S. Kulkarni, and H. Poor, "A collaborative training algorithm for distributed learning," *IEEE Trans. Inform. Theory*, vol. 55, pp. 1856–1871, Mar. 2009.

[2] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Automatic Control*, vol. 54, pp. 48–61, Jan. 2009.

[3] J. Duchi, A. Agarwal, and M. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Trans. Automatic Control*, vol. 57, pp. 592–606, Jun. 2011.

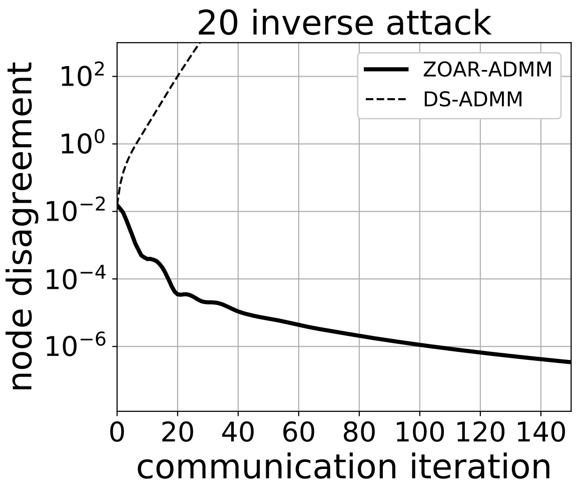

Fig. 8. Node disagreement comparison using MNIST: 20 Inverse attack.

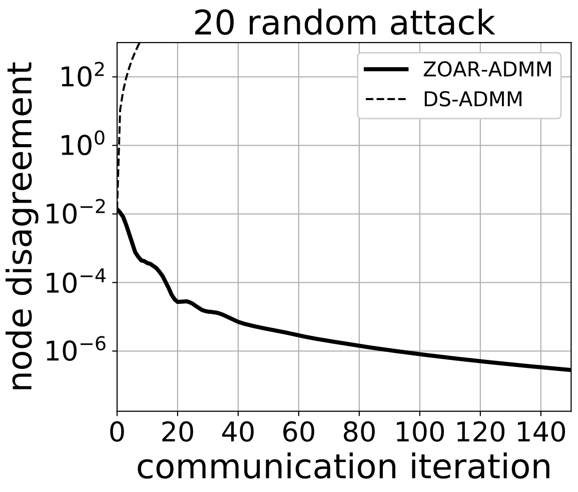

Fig. 9. Node disagreement comparison using MNIST: 20 Random attack.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, pp. 1–122, Jan. 2011.

[5] S. Liu, J. Chen, P. Chen, and A. Hero, "Zeroth-order online alternating direction method of multipliers: Convergence analysis and applications," in *International Conference on Artificial Intelligence and Statistics*, pp. 288–297, PMLR, Apr. 2018.

[6] Z. Yu, D. Ho, and D. Yuan, "Distributed randomized gradient-free mirror descent algorithm for constrained optimization," *IEEE Trans. Automatic Control*, pp. 1–1, Apr. 2021.

[7] Y. Tang, J. Zhang, and N. Li, "Distributed zero-order algorithms for nonconvex multi-agent optimization," *IEEE Transactions on Control of Network Systems*, pp. 269–281, Sep. 2020.

[8] A. Sahu, D. Jakovetic, D. Bajovic, and S. Kar, "Distributed zeroth order optimization over random networks: A Kiefer-Wolfowitz stochastic approximation approach," in *Proc. IEEE Conference on Decision and Control*, pp. 4951–4958, Dec. 2018.

[9] S. Liu, B. Kailkhura, P. Chen, P. Ting, S. Chang, and L. Amini, "Zeroth-order stochastic variance reduction for nonconvex optimization," *arXiv preprint arXiv:1805.10367*, Jun. 2018.

[10] A. Makhdoumi and A. Ozdaglar, "Convergence rate of distributed ADMM over networks," *IEEE Trans. Automatic Control*, vol. 62, pp. 5082–5095, Mar. 2017.

[11] 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 Processing*, vol. 62, pp. 1750–1761, Feb. 2014.

[12] P. Giselsson and S. Boyd, "Linear convergence and metric selection for Douglas-Rachford splitting and ADMM," *IEEE Trans. Automatic Control*, vol. 62, pp. 532–544, May 2016.

[13] L. Majzoobi, F. Lahouti, and V. Shah-Mansouri, "Analysis of distributed ADMM algorithm for consensus optimization in presence of node error," *IEEE Trans. Signal Processing*, vol. 67, pp. 1774–1784, May 2019.

[14] S. Lu, J. Lee, M. Razaviyayn, and M. Hong, "Linearized ADMM converges to second-order stationary points for non-convex problems," *IEEE Trans. Signal Processing*, vol. 69, pp. 4859–4874, Aug. 2021.

[15] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM on Asia Conference on Computer and Communications Security*, pp. 506–519, Apr. 2017.

[16] P. Chen, H. Zhang, Y. Sharma, J. Yi, and C. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Pro. ACM Workshop on Artificial Intelligence and Security*, pp. 15–26, Nov. 2017.

[17] T. Chen and G. Giannakis, "Bandit convex optimization for scalable and dynamic IoT management," *IEEE Internet of Things Journal*, vol. 6, pp. 1276–1286, May 2018.

[18] X. Lian, H. Zhang, C. Hsieh, Y. Huang, and J. Liu, "A comprehensive linear speedup analysis for asynchronous stochastic parallel optimization from zeroth-order to first-order," *Advances in Neural Information Processing Systems*, vol. 29, pp. 3054–3062, Dec. 2016.

[19] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proc. of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, p. 44, Dec. 2017.

[20] P. Blanchard, E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, pp. 119–129, Dec. 2017.

[21] X. Cao and L. Lai, "Robust distributed gradient descent with arbitrary number of byzantine attackers," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6373–6377, 2018.

[22] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," *arXiv preprint arXiv:1803.01498*, Mar. 2018.

[23] L. Chen, Z. Charles, D. Papailiopoulos, *et al.*, "Draco: Robust distributed training via redundant gradients," *arXiv preprint arXiv:1803.09877*, Jun. 2018.

[24] L. Su and J. Xu, "Securing distributed gradient descent in high dimensional statistical learning," *Proc. ACM on Measurement and Analysis of Computing Systems*, vol. 3, p. 12, Mar. 2019.

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

[26] X. Cao and L. Lai, "Distributed gradient descent algorithm robust to an arbitrary number of Byzantine attackers," *IEEE Trans. Signal Processing*, vol. 67, pp. 5850–5864, Nov. 2019.

[27] 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. of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 1544–1551, Jul. 2019.

[28] Z. Yang, A. Gang, and W. Bajwa, "Adversary-resilient inference and machine learning: From distributed to decentralized," *arXiv preprint arXiv:1908.08649*, Feb. 2020.

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

[30] X. Cao and L. Lai, "Distributed approximate Newton's method robust to Byzantine attackers," *IEEE Trans. Signal Processing*, vol. 68, pp. 6011–6025, Oct. 2020.

[31] L. Bottou, F. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.

[32] A. Agarwal, O. Dekel, and L. Xiao, "Optimal algorithms for online convex optimization with multi-point bandit feedback," in *COLT*, pp. 28–40, Citeseer, 2010.

[33] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, 2010.