2022 IEEE 61st Conference on Decision and Control (CDC)
December 6-9, 2022. Cancún, Mexico

Decentralized Learning Robust to Data Poisoning Attacks

Yanwen Mao Deepesh Data Suhas Diggavi Paulo Tabuada

2022 IEEE 61st Conference on Decision and Control (CDC) | 978-1-6654-6761-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/CDC51059.2022.9992702

***Abstract*— This paper addresses the problem of decentralized learning in the presence of data poisoning attacks. In this problem, we consider a collection of nodes connected through a network, each equipped with a local function. The objective is to compute the global minimizer of the aggregated local functions, in a decentralized manner, i.e., each node can only use its local function and data exchanged with nodes it is connected to. Moreover, each node is to agree on the said minimizer despite an adversary that can arbitrarily change the local functions of a fraction of the nodes. This problem setting has applications in robust learning, where nodes in a network are collectively training a model that minimizes the empirical loss with possibly attacked local data sets. In this paper, we propose a novel decentralized learning algorithm that enables all nodes to reach consensus on the optimal model, in the absence of attacks, and approximate consensus in the presence of data poisoning attacks.**

I. Introduction

This paper concerns decentralized federated learning that has seen several applications in the past decade [1]–[4]. In decentralized federated learning, we have a set of nodes connected via a communication network which are collaboratively learning a model that minimizes the empirical risk. However, in some scenarios, nodes may be subjected to malicious attacks, which break most learning algorithms developed for faultless networks [5], [6]. Therefore, it is of significant importance to develop learning algorithms that are robust to attacks.

The decentralized federated learning problem in an adversarial environment can also be considered as a Robust Decentralized Global Optimization (RDGO) problem. From this perspective, the data set at each node implicitly defines a local function $f_i(x)$, where $x \in \mathbb{R}^n$ is the optimization variable, and all nodes are required to approximate the minimizer x^* of the sum of their local functions. This problem has been studied in the past, but only for limited cases. For example, [7] and [8] solved the RDGO problem under a scalar model assumption. Different from these works, in this paper, we consider a milder type of attacks known as data poisoning attacks. These are conducted by an omniscient attacker who changes the local functions of the attacked node. Note, however, that attacked nodes are still able to execute its optimization/communication protocol. Moreover, our algorithm is developed for the high-dimensional case, which is much more challenging than the low-dimensional case since high-dimensionality poses a higher requirement on the computational complexity and the scaling of the approximation error with respect to dimension. Moreover, compared with [7] and [8], we provide a much tighter bound on the distance between the estimation and the true minimizer.

This work was funded in part by the Army Research Laboratory under Cooperative Agreement W911NF-17-2-0196 and in part by NSF grants 2007714 and 2139304.

Yanwen Mao, Deepesh Data, Suhas Diggavi, and Paulo Tabuada are with the Department of Electrical and Computer Engineering, University of California, Los Angeles, California, 90095, USA {yanwen.mao,deepesh.data,suhasdiggavi,tabuada}@ucla.edu

A. Our Contributions

The main contributions of this paper are:

1. We propose a novel filtering algorithm which robustly estimates the weighted sum of a set of vectors in $\mathbb{R}^n$ in the presence of data poisoning attacks. It is presented as Algorithm 2. Moreover, the distance between the computed weighted sum and the true value scales well with the dimension n ($\propto \sqrt{n}$), and the fraction ϵ of attacked vectors ($\propto \epsilon$). The algorithm is also light-weight since its computational complexity scales linearly with n.

2. We propose an algorithm that solves the RDGO problem (or the decentralized federated learning problem) in the presence of data-poisoning attacks. The algorithm guarantees the Euclidean distance between the obtained minimizer and the true one (in the absence of attacks) to be proportional to $\sqrt{n}$ and ϵ, which is proved in Theorem 1. Moreover, the proposed algorithm tolerates an attack up to half of the nodes.

We also note that, any result in this paper can be conveniently applied to the distributed case, even when the attack is Byzantine. More details regarding this claim can be obtained in Remark 5.

B. Related Works

As mentioned above, the Robust Distributed Learning (RDL) problem has been intensively studied. Recent works [9] and [10] solved the RDGO problem using a Robust Mean Estimation (RME) algorithm developed in [11]. The authors of these two works prove that, using their algorithms, the trained model parameter is close to the optimal up to some constant error. However, this error scales with $\sqrt{\epsilon}$ where ϵ is the fraction of attacked nodes, and at most $1/4$ nodes can be attacked. Papers [12] and [13] adopt the gradient compression technique and the trimmed-mean error filtering technique, respectively, which robustify their gradient descent algorithms against almost half of nodes being attacked, but leaves a problem that the distance between the computed model parameter and the true one scales linearly with the dimension n of the parameter space. Paper [14] made an attempt to limit the impact of the adversary by adding a regularization term in the loss function. On the other hand,

this term renders the error bound in [14] proportional to the number of attacked nodes, which is not desirable.

II. Preliminaries

A. Notation

Let $\mathbb{R}$, $\mathbb{R}^+$, and $\mathbb{N}$ denote the set of real, positive real, and natural numbers, respectively. Given a vector $v \in \mathbb{R}^n$ where n is a positive natural number, we use $\|v\|_2$ to denote the l_2 norm of v. Also, we define the all-ones vector of length n by $\mathbf{1}_n = (1, 1, \ldots, 1)^T$ and I_n to be the identity matrix of order n. The largest and smallest singular values of a matrix $A \in \mathbb{R}^{n\times p}$ are denoted by $\sigma_M(A)$ and $\sigma_m(A)$, respectively, where n, p are positive natural numbers. Moreover, we use $\nabla f(x)$ to denote the gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ evaluated at $x \in \mathbb{R}^n$. Further let $r \in \mathbb{R}^+$. We denote by $\mathcal{B}(x, r) = \{y \in \mathbb{R}^n | \|y - x\|_2 \leq r\}$ the ball centered at x with radius r.

A weighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ is a triple consisting of a set of vertices $\mathcal{V} = \{v_1, v_2, \ldots, v_p\}$ with cardinality p, a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, and a weighted adjacency matrix $\mathbf{A} \in \mathbb{R}^{p\times p}$ which will be defined very soon. The set of in-neighbors of a vertex $i \in \mathcal{V}$, denoted by $\mathcal{N}_i^{in} = \{j \in \mathcal{V} | (j, i) \in \mathcal{E}\}$, is the set of vertices connected to i by an edge. Similarly, the set of out-neighbors of a vertex $i \in \mathcal{V}$ is defined by $\mathcal{N}_i^{out} = \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$. We assume each vertex is both an in-neighbor and an out-neighbor of itself. The weighted adjacency matrix $\mathbf{A}$ of the graph $\mathcal{G}$ is defined entry-wise. The entry in the i-th row and j-th column, a_{ij}, satisfies $0 < a_{ij} < 1$ if $(i, j) \in \mathcal{E}$ and otherwise $a_{ij} = 0$.

B. Problem Formulation

We consider a set P of p nodes connected via a communication network, modelled as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$. The set V in $\mathcal{G}$ represents the set of nodes and the set $\mathcal{E}$ represents the set of communication links between all pairs of nodes. In particular, an edge $(i, j) \in \mathcal{E}$ exists if and only if node j can receive information from node i. Moreover, each communication link (i, j) is associated with a positive scalar value $a_{ij} > 0$, which, we recall, is the (i, j)-th entry of $\mathbf{A}$.

Definition 1. *Consider a set P of p nodes connected via a communication network. Each node $i \in P$ is equipped with a local function $f_i : \mathbb{R}^n \to \mathbb{R}$ where $x \in \mathbb{R}^n$ is the optimization variable. The RDGO problem asks each node to find an optimizer x^* of the aggregation of the local functions:*

$$f = \sum_{i\in P} f_i,$$

using its local function f_i and messages exchanged with its neighboring nodes, notwithstanding some local functions have been altered by a data poisoning attack.

In the decentralized federated learning problem, each node i has a local data set $\mathcal{Z}_i = \{z_{i1}, z_{i2}, \ldots, z_{iN}\}$ of cardinality N. The federated learning problem asks all nodes to collectively minimize the following risk function $\frac{1}{N}\sum_{i=1}^{p}\sum_{j=1}^{N} l(\mathbf{w}, z_{ij})$ with respect to $\mathbf{w}$. In this case, each local function $f_i(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N} l(\mathbf{w}, z_{ij})$ is implicitly defined by the local data set $\mathcal{Z}_i$ at node i. The aggregation f of local functions is also named as the global function in this paper.

C. Attack Model

The solution to the global optimization problem is trivial in the absence of attacks [15]–[18]. However, the problem gets more interesting, and also more complicated, when some nodes are subject to attacks. In this paper, we assume that a subset $P_b \subset P$ of nodes are subject to a data poisoning attack, which is able to replace the original function f_j of an attacked node $j \in P_b$ with any other function chosen by the adversary. Moreover, we define $S_g = S \backslash S_b$ to be the set of attack-free nodes. For simplicity, we also use $\tilde{f}_i$ to denote the local function of an attack-free node $i \in S_g$ after the data poisoning attack. It is trivially seen that, for an attack-free node i, $\tilde{f}_i = f_i$.

The adversary that launches the data poisoning attack is assumed to be omniscient, i.e., it has full knowledge of the communication graph, the local functions of all nodes, the algorithm each node executes, etc. Moreover, it is able to arbitrarily alter the local functions of the attacked nodes. However, differently from Byzantine attacks, we assume that all nodes, even those subject to data poisoning attacks, are able to execute their protocols correctly. Also, in the context of this paper, we assume the attack is perpetrated before the nodes start executing the algorithm that solves the RAGD problem, which we discuss in the next section. The attacked local functions will not change once the algorithm starts running. This definition of the data poisoning attack is in line with other works (for example [19], [20], [21], and [22]) where data poisoning attacks are studied.

Remark 1. *It was argued in [7] that it is impossible to exactly recover the optimizer x^* when some local functions are attacked by an adversary and when there is no special relationships between the local functions. Therefore, instead of exactly recovering the optimizer x^*, we study in this paper how well each node can approximate x^* using its possibly attacked local function and messages exchanged with its neighbors.*

D. Assumptions

We study the RGDO problem under the following assumptions, some of which have already been discussed:

Assumption 1. *The communication graph is fixed, connected, and doubly-stochastic (i.e., the adjacency matrix $\mathbf{A}$ of the graph is a doubly-stochastic matrix). Moreover, the weight associated with each link is known to the corresponding receiver node, for example, node j is aware of a_{ij} for any $i \in P$.*

Assumption 2. *There exists an $0 < \epsilon < \frac{1}{2}$, known to all nodes in the network, such that for any node j, the sum*

of link weights corresponding to its attacked in-neighbors is upper bounded by ϵ, i.e.:

$$\sum_{i\in\mathcal{N}_j^{in}\cap S_b} a_{ij} \leq \epsilon,\ \forall j \in P. \tag{1}$$

Assumption 3. *Each local function is differentiable, and the Euclidean distance between the gradients of any two local functions evaluated at any point x in the working domain is upper bounded by some constant $\kappa > 0$, i.e.:*

$$\|\nabla f_i(x) - \nabla f_j(x)\|_2 \leq \kappa,\ \forall i, j \in P. \tag{2}$$

Assumption 4. *The global function f is L-smooth and ν-strongly convex, each local function f_i is L'-smooth.*

Assumption 1 is a constraint on the communication network topology. This assumption is the simplest one that enables a solution to the Decentralized Average Consensus (DAC) problem, where a network of nodes, each having a local initial value, seeks to agree on the average of their initial values [23], [24]. Assumption 2 restricts the power of the adversary. For example, if node j has k neighbors and the weight on each link to node j is $1/k$, Assumption 2 requires that less than half of the links can be attacked since ϵ is required to be smaller than $1/2$. Similar assumptions were made in [8], with the slight difference that in [8] it is assumed that the number of attacked nodes in a neighborhood is upper-bounded. Assumption 3 is widely accepted in decentralized (distributed) machine learning literature, for example, [25] and [9] adopted very similar assumptions to enable filtering information from attacked nodes. Assumption 4 is also widely accepted in decentralized optimization literature. Also note that by Assumption 4 the optimizer x^* of function f is unique, and hence all nodes are essentially asked to reach consensus on this unique optimizer.

III. The Resilient Averaging Gradient Descent Algorithm

In this section, we introduce an algorithm called Resilient Averaging Gradient Descent (RAGD), which enables all nodes to approximate the global minimum x^* and thus solves the RDGO problem. Algorithm 1 is a pseudo-code description of the RAGD algorithm.

The RAGD algorithm has two loops, an inner loop (lines 4-8) and an outer loop (lines 3-16). In the inner loop, all the nodes are asked to run a linear iterative algorithm aiming at reaching consensus on the average of their local estimates. The input parameter τ controls the number of iterations executed in the inner loop. The estimate at node j in t-th iteration of the outer loop and k-th iteration of the inner loop is denoted by $x_j^k[t]$. To proceed, we directly provide the following result on the convergence property of the inner loop.

Lemma 1. *Consider a set P of nodes and assume each starts with an initial value $x_i^0[t]$ and executes lines 4-8 of the RAGD algorithm in parallel. Define $\bar{x}[t] = \frac{1}{p}\sum_{i\in P} x_i^0[t]$ and $d^k[t] = \max_{i,j\in P} \|x_i^k[t] - x_j^k[t]\|_2$. The following two properties hold for any $t \in \mathbb{N}$:*

1) $\frac{1}{p}\sum_{i\in P} x_i^\tau[t] = \bar{x}[t],\ \forall \tau \in \mathbb{N}$,
2) *there exists an $\alpha \in \mathbb{R}^+$ and a $\rho \in (0,1)$ such that for any $\tau \in \mathbb{N}$, $d^\tau[t] \leq \alpha\rho^\tau d^0[t]$.*

A similar result can be found in [26]. Due to page limitations certain proofs are omitted.

Algorithm 1: Resilient Averaging Gradient Descent (RAGD) Algorithm for Node j

Input: $\{a_{ij} | i \in \mathcal{N}_j^{in}\}$, $\tilde{f}_j$, ϵ, η, $\tau \in \mathbb{N}$;
Initialization: $x_j^0[0] := 0$;
1 **for** $t = 0, 1, 2, \ldots$ **do**
2 **for** $k = 0, 1, 2, \ldots, \tau - 1$ **do**
3 Broadcast $x_j^k[t]$;
4 Receive $x_i^k[t]$ from $i \in \mathcal{N}_j^{in}$;
5 $x_j^{k+1}[t] := \sum_{i\in\mathcal{N}_j^{in}} a_{ij} x_i^k[t]$;
6 **end**
7 Compute and broadcast the gradient $X_j[t] := \nabla\tilde{f}_j(x_j^\tau[t])$ of its local function;
8 Receive $X_i[t]$ from $i \in \mathcal{N}_j^{in}$;
9 $\hat{\mu}_j[t] := \text{RWSE}(\{(a_{ij}, X_i[t]), i \in \mathcal{N}_j^{in}\}, \epsilon)$ where the RWSE algorithm will be introduced in Section IV;
10 $x_j^0[t+1] := x_j^\tau[t] - \eta\hat{\mu}_j[t]$;
11 **end**
Output: $x_j^\tau[t]$;

In the outer loop, all nodes are first asked to reach consensus on the average of their local estimates by executing the inner loop. Then each node is asked to compute and broadcast the gradient of its (possibly attacked) local function (lines 9-10). We note that some gradients are not reliable since some local functions have been altered by the data poisoning attack. Upon receiving gradients from all its neighbors, each node runs a screening algorithm (the RWSE algorithm) which allows each node to resiliently approximate the weighted average of the gradients it receives (line 11), and in the end updates its local parameter by performing a gradient descent step based on the output of the RWSE algorithm (line 15).

Remark 2. *In line 9 of Algorithm 1, we ask each node j to compute the gradient of its (possibly attacked) local function evaluated at its current local estimate $x_j^\tau[t]$. If node j is free from attack, then $X_j[t] = \nabla f_j(x_j^\tau[t])$, i.e., the computed gradient equals the gradient of its original local function evaluated at the same point. However, if node j is attacked, we make no assumptions on the relationship between $X_j[t]$ and $\nabla f_j(x_j^\tau[t])$ except that $X_j[t]$ exists.*

IV. The Robust Weighted Sum Estimation Algorithm

In this section, we study the problem of how each node can resiliently compute the weighted sum of its neighbors' gradients under assumptions 1-3, despite a portion of the gradients having been attacked. To solve this problem, we

propose a novel algorithm termed the Robust Weighted Sum Estimation (RWSE) algorithm.

The RWSE algorithm is not only the key for solving the RDGO problem, but also has other applications, for example, it can be conveniently applied to solve the distributed Byzantine-resilient federated learning problem[1]. Moreover, we note that the robust weighted sum estimation problem is a generalization of the well-known RME problem, hence the RWSE algorithm solves the robust mean estimation problem assuming Assumptions 1-3 hold.

A. Algorithm Description

Since the execution node j is fixed, in this section we drop this index and represent the weights $\{a_{ij} : i \in P\}$ by $\{a_i : i \in P\}$. Note that $\sum_{i\in P} a_i = 1$ by Assumption 1. Let S be the set of in-neighbors of node j, the message X_i that node j receives from node i satisfies:

$$X_i = \nabla f_i, \quad i \in S_g \tag{3}$$

where we also dropped time indices t and τ and used ∇f_i in lieu of $\nabla f_i(x_i^\tau[t])$ for simplicity. The goal is to approximate the weighted average $\mu_g = \sum_{i\in S} a_i \nabla f_i$ using the data $\{(a_1, X_1), (a_2, X_2), \ldots, (a_p, X_p)\}$ under Assumptions 2 and 3. Algorithm 2 is a pseudo code description of the RWSE algorithm.

Algorithm 2: Robust Weighted Sum Estimation (RWSE)

Input: $\{(a_1, X_1), (a_2, X_2), \ldots, (a_p, X_p)\}, \epsilon$;
Initialization: $g := 1$, $temp := 0$, $w_0 := 0$,
$V := \{X_1, X_2, \ldots, X_p\}$, $V_e := \{\}$, $g_e := 0$;
while $g > 1 - \epsilon$ **do**
  compute the Euclidean distance between every pair of nodes and find out a pair with the maximum distance. If there are multiple pairs, pick one of them arbitrarily. Without loss of generality we assume that vectors X_i and X_h are picked;
  $s_i := \sum_{z\in V}(a_z \|X_i - X_z\|_2)$;
  $s_h := \sum_{z\in V}(a_z \|X_h - X_z\|_2)$;
  if $s_i > s_h$ **then**
    $u := i$;
  else
    $u := h$;
  end
  $V := V \backslash \{X_u\}$, $g := g - a_u$, $temp := u$;
end
if $a_{temp} > \epsilon$ **then**
  $V_e =: V \cup \{X_{temp}\}$, $g_e := g + a_{temp}$;
else
  $V_e := V$, $g_e := g$;
end
Output: $\hat{\mu} := \frac{1}{g_e} \sum_{i\in V_e} a_i X_i$;

[1]See [9] for a formal definition of the distributed Byzantine-resilient federated learning problem.

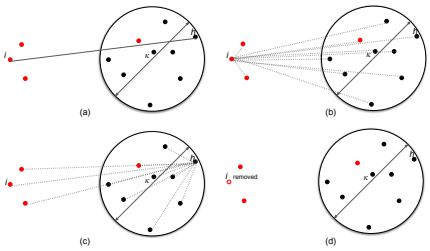

Fig. 1. Visualization of Algorithm 2

To explain the RWSE algorithm we describe its execution on the example in Figure 1. In Figure 1, a red dot denotes an attacked vector while a black dot denotes an attack-free vector. All attack-free vectors (black dots) lie in a ball with diameter κ whereas there are no restrictions on the position of attacked vectors. In each iteration, the execution node first finds the pair of vectors (i, h) with the maximum Euclidean distance (subgraph (a), also line 4 in Algorithm 2), then computes and compares the weighted sum of the distance between vector i and all other vectors and the weighted sum of the distance between vector h and all other vectors (subgraph (b) and (c), also lines 5-6 in Algorithm 2). In the problem instance represented by Figure 1, vector h is closer to the rest of vectors compared with the attacked vector i, hence in the last subgraph (d) vector i is removed according to lines 6-12 in Algorithm 2.

In addition, we use a scalar variable $temp$ to store the identity of the latest removed vector. By Assumption 2 if the weight a_i associated to a vector X_i satisfies $a_i > \epsilon$, then this vector cannot be attacked. Therefore, if a vector X_i with weight $a_i > \epsilon$ is removed at some iteration and then the algorithm terminates, there is no harm restoring this vector X_i since it must be a good vector. By doing so we have the following guarantee of the weight sum of the remaining vectors: $g_e \geq 1 - 2\epsilon$, where g_e is defined in line 14 and line 16 in Algorithm 2.

B. Performance

The termination of Algorithm 2 (RWSE) is guaranteed by line 4, and its performance is characterized by the following lemma, which shows how close the output $\hat{\mu}$ of the RWSE algorithm is from the true average μ_g.

Lemma 2. *Consider the RWSE algorithm with inputs $\{(a_1, X_1), (a_2, X_2), \ldots, (a_p, X_p)\}$ satisfying: (1) $a_1 + a_2 + \cdots + a_p = 1$; (2) $a_i > 0$, $\forall i \in P$; (3) $\|\nabla f_i - \nabla f_h\|_2 \leq \kappa$, $\forall i, h \in P$, and ϵ satisfying (4) $\epsilon < \frac{1}{2}$. Define $\mu_g = \sum_{i=1}^p a_i \nabla f_i$. The output $\hat{\mu}$ of the RWSE algorithm satisfies:*

$$\|\hat{\mu} - \mu_g\|_2 \leq \left(2 + \frac{3 - 4\epsilon}{(1 - 2\epsilon)^2}\right) \epsilon\kappa. \tag{4}$$

Remark 3. *We see that the RWSE algorithm scales well with the dimension n, since the computational complexity of the RWSE algorithm grows linearly with of n. Moreover, according to Lemma 2, the error of the RWSE algorithm scales with ϵ, which outperforms many RME algorithms*

whose error scales with $\sqrt{\epsilon}$ [11]. It is also noteworthy that the error of the RWSE algorithm scales with κ, which implicitly grows with $\sqrt{n}$.

V. Performance of the RAGD Algorithm

In this section we prove the correctness of the RAGD algorithm. We start with an intuitive explanation of the RAGD algorithm: the inner loop can be considered as an initialization step, in which each node initializes their estimate to be the average of estimates of all nodes throughout the network, up to some error which decreases exponentially with τ by Lemma 1. Executing lines 9-12 in Algorithm 1 brings the following two consequences:

1. The average of local estimates moves towards the minimum point (or a minimum point if there are multiple), up to some constant error.

2. The distance between two local estimates may increase.

From the previous discussion we observe the following two facts: (1) the local estimates of all nodes are clustered in a ball, and (2) the centroid of the ball moves towards the minimizer up to a constant error. It is a natural consequence of these two facts that the estimate at any node is close to the minimizer. We recall the definition $d^k[t] = \max_{i,j\in P} \|x_i^k[t] - x_j^k[t]\|_2$ and proceed with the following two propositions which formally state the observations mentioned above:

Proposition 1. *Consider a set P of nodes in a communication network satisfying Assumption 1. Each node has a local function which satisfies Assumptions 3 and 4 and some local functions are altered by a data poisoning attack which satisfies Assumption 2. Let all nodes run the RAGD algorithm in parallel. For any step size $\eta > 0$ and any given input $r > 0$, there always exist a $\tau_0 \in \mathbb{N}$ such that $d^\tau[t] \leq r$ implies $d^\tau[t+1] \leq r$ for any $t \in \mathbb{N}$, if $\tau \geq \tau_0$.*

Proposition 2. *Let the assumptions in Proposition 1 hold and further let $d^\tau[t] \leq r$ for any $t, \tau \in \mathbb{N}$ and $r > 0$. Then the following equation holds for any $\eta > 0$:*

$$\bar{x}[t+1] - \bar{x}[t] = -\frac{\eta}{p}\nabla f(\bar{x}[t]) + \frac{\eta}{p} l[t], \tag{5}$$

for some $l[t]$ satisfying $\|l[t]\|_2 \leq pc_\epsilon(\kappa + 2L'r) + pL'r$, where $c_\epsilon = \frac{3\epsilon - 4\epsilon^2}{(1-2\epsilon)^2}$.

For simplicity, we define $\xi = pc_\epsilon(\kappa + 2L'r) + pL'r$. Inequality (5) shows that, if we compare the average at the iteration $t+1$ and the average at iteration t, we determine that the average $\bar{x}[t]$ is updated with a "polluted" gradient, which differs from the true gradient $\nabla f(\bar{x}[t])$ by a vector $l[t]$ whose Euclidean norm is upper bounded by ξ, using step size $\frac{\eta}{p}$. As we will soon see, this "polluted" gradient only leads to a constant error.

Theorem 1. *Consider a set of nodes in a communication network satisfying Assumption 1, each equipped with a local function satisfying Assumption 3. Moreover, assume a subset of nodes are subject to a data poisoning attack satisfying Assumption 2. For any user input $r > 0$, suppose all nodes in the network run the RAGD algorithm with $\eta = \frac{p}{L}$ and parameter $\tau \geq \tau_0$ defined in Proposition 1, then the output of every node $j \in P$ at any time $t \in \mathbb{N}$ satisfies:*

$$\|x_j^\tau[t] - x^*\|_2 \leq \beta^k \|x_j^0[0] - x^*\|_2 + \frac{\xi}{(1-\beta)L} + r, \tag{6}$$

where $\beta = \sqrt{1 - \frac{\nu}{L}}$.

Proof. We observe that, by choosing $\eta = \frac{p}{L}$, the update rule (5) for the average degenerates to:

$$\bar{x}[t+1] - \bar{x}[t] = -\frac{1}{L}(\nabla f(\bar{x}[t] - l[t]). \tag{7}$$

By Assumption 4, the function f is both ν-strongly convex and L-smooth. This implies for any pair $x, y \in \mathbb{R}^n$, the following two inequalities hold:

$$f(y) - f(x) \leq \nabla f^T(x)(y-x) + \frac{L}{2}\|y-x\|_2^2, \tag{8}$$

$$f(x) - f(y) \geq \nabla f^T(y)(x-y) + \frac{\nu}{2}\|y-x\|_2^2. \tag{9}$$

A simple reorganization of Equation (9) yields:

$$f(y) - f(x) \leq \nabla f^T(y)(y-x) - \frac{\nu}{2}\|y-x\|_2^2. \tag{10}$$

We consider the following set of equalities and inequalities for any $x, y \in \mathbb{R}^n$ and $x^+ = x - \frac{1}{L}\nabla f(x)$, which will be used later.

$$\begin{aligned}
& f(x^+) - f(y) \\
= \;& f(x^+) - f(x) + f(x) - f(y) \\
\leq \;& \nabla f^T(x)(x^+ - x) + \frac{L}{2}\|x^+ - x\|_2^2 \\
& + \nabla f^T(x)(x-y) - \frac{\nu}{2}\|x-y\|_2^2 \\
= \;& \nabla f^T(x)(x^+ - y) + \frac{1}{2L}\|\nabla f(x)\|_2^2 - \frac{\nu}{2}\|x-y\|_2^2 \\
= \;& \nabla f^T(x)(x - \frac{1}{L}\nabla f(x) - y) \\
& + \frac{1}{2L}\|\nabla f(x)\|_2^2 - \frac{\nu}{2}\|x-y\|_2^2 \\
= \;& \nabla f(x)^T(x-y) - \frac{1}{2L}\|\nabla f(x)\|_2^2 - \frac{\nu}{2}\|x-y\|_2^2.
\end{aligned}$$

In particular, when $y = x^*$, we have:

$$\begin{aligned}
0 \;\leq\; & f(x^+) - f(x^*) \\
\leq\; & \nabla f^T(x)(x - x^*) - \frac{1}{2L}\|\nabla f(x)\|_2^2 \\
-\; & \frac{\nu}{2}\|x - x^*\|_2^2.
\end{aligned} \tag{11}$$

In the following we prove the result stated in the theorem.

$$\begin{aligned}
& \|\bar{x}[t+1] - x^*\|_2 \\
= \;& \|\bar{x}[t] - x^* - \frac{1}{L}(\nabla f(\bar{x}[t]) - l[t])\|_2 \\
\leq \;& \|\bar{x}[t] - x^* - \frac{1}{L}\nabla f(\bar{x}[t])\|_2 + \frac{1}{L}\|l[t]\|_2 \\
\leq \;& \|\bar{x}[t] - x^* - \frac{1}{L}\nabla f(\bar{x}[t])\|_2 + \frac{\xi}{L} \\
\overset{(a)}{\leq} \;& \sqrt{(1 - \frac{\nu}{L})\|\bar{x}[t] - x^*\|_2^2 +} + \frac{\xi}{L} \\
\overset{(b)}{=} \;& \beta\|\bar{x}[t] - x^*\|_2 + \frac{\xi}{L}, \qquad (12)
\end{aligned}$$

where in step (a) we plug in Equation (11) with $x = \bar{x}[t]$ and in step (b) we use the definition $\beta = \sqrt{1 - \frac{\nu}{L}}$. We note that $\beta \in (0, 1)$. Solving Equation (12) recursively gives:

$$\|\bar{x}[t] - x^*\|_2 \leq \beta^t \|\bar{x}[0] - x^*\|_2 + \frac{\xi}{(1-\beta)L}. \qquad (13)$$

Together with Proposition 1 we finish the proof. □

Remark 4. *Our RAGD algorithm guarantees that the distance between the computed minimizer and the true global minimizer (in the absence of attacks) is bounded by a constant error term. This differs some existing works [8], [27] in which the computed minimizer is only guaranteed to lie in the smallest hyper-rectangle that contains all local minimizers. Moreover, our error term scales linearly with $\sqrt{n}$ and ϵ when at most half of nodes are under attack.*

Remark 5. *We also note that, the RAGD algorithm can be conveniently extended to solving the Byzantine-resilient distributed optimization problem*[2]*. To solve this problem, we ask, in each iteration, the central server to collect the gradients $X_1, X_2, \ldots, X_p$ from all workers and then execute the RWSE algorithm using identical weight values (i.e., $a_i = \frac{1}{p},\ \forall i = 1, 2, \ldots p$). Then the central server is asked to send the output of the RWSE algorithm to all the workers and all workers update its estimate with the received value.*

REFERENCES

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

[2] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[3] M. Ma, A. N. Nikolakopoulos, and G. B. Giannakis, "Fast decentralized learning via hybrid consensus admm," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 3829–3833.

[4] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[5] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[6] K. Driscoll, B. Hall, M. Paulitsch, P. Zumsteg, and H. Sivencrona, "The real byzantine generals," in *The 23rd Digital Avionics Systems Conference (IEEE Cat. No. 04CH37576)*, vol. 2. IEEE, 2004, pp. 6–D.

[7] L. Su and N. H. Vaidya, "Fault-tolerant multi-agent optimization: optimal iterative distributed algorithms," in *Proceedings of the 2016 ACM symposium on principles of distributed computing*, 2016, pp. 425–434.

[8] S. Sundaram and B. Gharesifard, "Distributed optimization under adversarial nodes," *IEEE Transactions on Automatic Control*, vol. 64, no. 3, pp. 1063–1076, 2018.

[9] D. Data and S. Diggavi, "Byzantine-resilient high-dimensional sgd with local iterations on heterogeneous data," in *International Conference on Machine Learning*. PMLR, 2021, pp. 2478–2488.

[10] ——, "Byzantine-resilient sgd in high dimensions on heterogeneous data," *arXiv preprint arXiv:2005.07866*, 2020.

[11] I. Diakonikolas, G. Kamath, D. Kane, J. Li, A. Moitra, and A. Stewart, "Robust estimators in high-dimensions without the computational intractability," *SIAM Journal on Computing*, vol. 48, no. 2, pp. 742–864, 2019.

[12] A. Ghosh, R. K. Maity, S. Kadhe, A. Mazumdar, and K. Ramchandran, "Communication-efficient and byzantine-robust distributed learning with error feedback," *IEEE Journal on Selected Areas in Information Theory*, 2021.

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

[14] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 1544–1551.

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

[16] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "A decentralized second-order method with exact linear convergence rate for consensus optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 507–522, 2016.

[17] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network newton distributed optimization methods," *IEEE Transactions on Signal Processing*, vol. 65, no. 1, pp. 146–161, 2016.

[18] F. Bullo, *Lectures on network systems*. Kindle Direct Publishing, 2019.

[19] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli, "Is feature selection secure against training data poisoning?" in *international conference on machine learning*. PMLR, 2015, pp. 1689–1698.

[20] N. Baracaldo, B. Chen, H. Ludwig, and J. A. Safavi, "Mitigating poisoning attacks on machine learning models: A data provenance based approach," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 103–110.

[21] Y. Shoukry, P. Martin, P. Tabuada, and M. Srivastava, "Non-invasive spoofing attacks for anti-lock braking systems," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2013, pp. 55–72.

[22] Y. Shoukry, P. Martin, Y. Yona, S. Diggavi, and M. Srivastava, "Pycra: Physical challenge-response authentication for active sensors under spoofing attacks," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1004–1015.

[23] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of parallel and distributed computing*, vol. 67, no. 1, pp. 33–46, 2007.

[24] K. Cai and H. Ishii, "Average consensus on arbitrary strongly connected digraphs with time-varying topologies," *IEEE Transactions on Automatic Control*, vol. 59, no. 4, pp. 1066–1071, 2014.

[25] Z. Yang and W. U. Bajwa, "Byrdie: Byzantine-resilient distributed coordinate descent for decentralized learning," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 4, pp. 611–627, 2019.

[26] S. S. Kia, B. Van Scoy, J. Cortes, R. A. Freeman, K. M. Lynch, and S. Martinez, "Tutorial on dynamic average consensus: The problem, its applications, and the algorithms," *IEEE Control Systems Magazine*, vol. 39, no. 3, pp. 40–72, 2019.

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

[2] We refer interested readers to [9] for the definition of the Byzantine-resilient distributed optimization problem.