Distributed Statistical Min-Max Learning in the Presence of Byzantine Agents

Arman Adibi*, Aritra Mitra*, George J. Pappas and Hamed Hassani

Abstract—Recent years have witnessed a growing interest in the topic of min-max optimization, owing to its relevance in the context of generative adversarial networks (GANs), robust control and optimization, and reinforcement learning. Motivated by this line of work, we consider a multi-agent min-max learning problem, and focus on the emerging challenge of contending with worst-case Byzantine adversarial agents in such a setup. By drawing on recent results from robust statistics, we design a robust distributed variant of the extra-gradient algorithm - a popular algorithmic approach for min-max optimization. Our main contribution is to provide a crisp analysis of the proposed robust extra-gradient algorithm for smooth convex-concave and smooth strongly convex-strongly concave functions. Specifically, we establish statistical rates of convergence to approximate saddle points. Our rates are near-optimal, and reveal both the effect of adversarial corruption and the benefit of collaboration among the non-faulty agents. Notably, this is the first paper to provide formal theoretical guarantees for large-scale distributed min-max learning in the presence of adversarial agents.

I. INTRODUCTION

We consider a min-max learning problem of the form

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} f(x, y) \triangleq \mathbb{E}_{\xi \sim \mathcal{D}}[F(x, y; \xi)].$$

(1)

Here, $\mathcal{X}$ and $\mathcal{Y}$ are convex, compact sets in $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively; $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ are model parameters; $\xi$ is a random variable representing a data point sampled from the distribution $\mathcal{D}$; and $f(x, y)$ is the population function corresponding to the stochastic function $F(x, y; \xi)$. Throughout this paper, we assume that $f(x, y)$ is continuously differentiable in $x$ and $y$, and is convex-concave over $\mathcal{X} \times \mathcal{Y}$. Specifically, $f(\cdot, y) : \mathcal{X} \to \mathbb{R}$ is convex for every $y \in \mathcal{Y}$, and $f(x, \cdot) : \mathcal{Y} \to \mathbb{R}$ is concave for every $x \in \mathcal{X}$. Our goal is to find a saddle point $(x^*, y^*)$ of $f(x, y)$ over the set $\mathcal{X} \times \mathcal{Y}$, where a saddle point is defined as a vector pair $(x^*, y^*) \in \mathcal{X} \times \mathcal{Y}$ that satisfies

$$f(x^*, y) \leq f(x^*, y^*) \leq f(x, y^*), \forall x \in \mathcal{X}, y \in \mathcal{Y}.$$  

(2)

The min-max optimization problem described above features in a variety of applications: from classical developments in game theory [1] and online learning [2], to robust optimization [3] and reinforcement learning [4]. More recently, in the context of machine learning, min-max problems have found important applications in training generative adversarial networks (GANs) [5], and in robustifying deep neural networks against adversarial attacks [6]. Motivated by this recent line of work, we consider a min-max learning problem of the form in Eq. (1), where the data samples required for finding a saddle-point are distributed across multiple devices (agents). Specifically, we focus on a large-scale distributed setup comprising of $M$ agents, each of which can access i.i.d. data samples from the distribution $\mathcal{D}$. The agents collaborate under the orchestration of a central server to compute an approximate saddle point of statistical accuracy higher relative to the setting when they act alone. The intuition here is simple: since all agents receive data samples from the same distribution, exchanging information via the server can help reduce the randomness (variance) associated with these samples.\footnote{An example of the above setup that aligns with the modern federated learning paradigm is one where multiple devices (e.g., cell phones or tablets) collaborate via a server to train a robust statistical model; see, for instance, [7].}

To reap the benefits of collaboration in modern distributed computing systems, one needs to contend with the critical challenge of security. In particular, this challenge arises from the fact that the individual agents in such systems are easily susceptible to adversarial attacks. In fact, unless appropriately accounted for, even a single malicious agent can severely degrade the overall performance of the system by sending corrupted messages to the central server.

Objective. Thus, given the emerging need for security in large-scale computing, our objective in this paper is to design an algorithm that achieves near-optimal statistical performance in the context of distributed min-max learning, while being robust to worst-case attacks. To that end, we consider a setting where a fraction of the agents is Byzantine [8]. Each Byzantine agent is assumed to have complete knowledge of the system and learning algorithms; moreover, leveraging such knowledge, the Byzantine agents can send arbitrary messages to the server and collude with each other.

Challenges. Even in the absence of noise or attacks, recent work [9] has shown that algorithms such as gradient descent ascent (GDA) can diverge for simple convex-concave functions. We have to contend with both noise (due to our statistical setup) and worst-case attacks - this makes the analysis for our setting non-trivial. In particular, the adversarial agents can introduce complex probabilistic dependencies across iterations that need to be carefully accounted for; we do so in this work by making the following contributions.

\footnote{This intuition will be made precise in Section IV.}
Contributions. Our contributions are summarized below.
• **Problem.** Given the importance and relevance of security, several recent works have studied distributed optimization/learning in the face of adversarial agents. However, we are unaware of any analogous paper for adversarially-robust distributed min-max learning. Our work closes this gap.

• **Algorithm.** In Section III, we develop an algorithm for finding an approximate saddle point to the min-max learning problem in Eq. (1), subject to the presence of Byzantine agents. Our proposed algorithm - called Robust Distributed Extra-Gradient (RDEG) - brings together two separate algorithmic ideas: (i) the classical extra-gradient algorithm due to Korpelevich [10] that has gained a lot of popularity due to its empirical performance in training GANs, and (ii) the recently proposed univariate trimmed mean estimator due to Lugosi and Mendelson [11].

• **Theoretical Results.** Our main contribution is to provide a rigorous theoretical analysis of the performance of RDEG for smooth convex-concave (Theorem 2), and smooth strongly convex-strongly concave (Theorem 3) settings. In each case, we establish that as long as the fraction of corrupted agents is “small”, RDEG guarantees convergence to approximate saddle points at near-optimal statistical rates with high probability. The rates that we derive precisely highlight the benefit of collaboration in effectively reducing the variance of the noise model. At the same time, they indicate the (unavoidable) additive bias introduced by adversarial corruption. Notably, our results in the context of min-max learning complement those of a similar flavor in [12] for stochastic optimization under attacks. However, our analysis differs significantly from that in [12]: unlike the covering argument employed in [12], our proofs rely on a simpler, and more direct probabilistic analysis. An immediate benefit of such an analysis is that one can build on it for the more challenging nonconvex-nonconcave setting as future work.

Related Work. In what follows, we discuss connections to relevant strands of literature.

• **Min-Max Optimization.** Convergence guarantees of first-order algorithms for saddle point problems over compact sets were studied in [13] and [14]. More recently, there has been a surge of interest in analyzing the performance of such algorithms from different perspectives: a dynamical systems approach in [15], [16], and a proximal point perspective in [17]. We refer to [18] for a detailed survey on this topic.

• **Robust Distributed Optimization and Learning.** Robustness to adversarial agents in distributed optimization has been extensively studied in [19]–[21]. However, these works consider deterministic settings, and do not provide statistical error rates like we do. In the context of statistical learning over a server-client computing architecture, several works have proposed and analyzed robust algorithms [12], [22]–[25]. Notably, none of the above works consider the min-max learning problem studied in this paper.

• **Robust Statistics.** Robust mean estimation in the presence of outliers is a classical topic in statistics pioneered by Huber [26], with follow-up work in [27]. In our work, we exploit some recent results on this topic from [11].
Algorithm 1 Robust Distributed Extra-Gradient (RDEG)

Require: Initial vectors $x_1 \in X$, $y_1 \in Y$; algorithm parameters: step-size $\eta > 0$ and trimming parameter $\epsilon$.

1: for $t = 1, \ldots, T$ do
2: Server sends $(x_t, y_t)$ to each agent.
3: Each normal agent $i$ draws an i.i.d. sample $\xi_{1:t}^{(i)} \sim D_i$, and transmits $g_x(x_t, y_t; \xi_{1:t}^{(i)}), g_y(x_t, y_t; \xi_{1:t}^{(i)})$ to server.\footnote{Recall that $\{g_x(x_t, y_t; \xi_{1:t}^{(i)}), g_y(x_t, y_t; \xi_{1:t}^{(i)})\}$ could be arbitrary vectors for an adversarial agent $i \in S$.}
4: Server computes robust gradients:
   \[
   \tilde{g}_x(x_t, y_t) \leftarrow \text{Trim}(g_x(x_t, y_t; \xi_{1:t}^{(i)}): i \in [M])
   \]
   \[
   \tilde{g}_y(x_t, y_t) \leftarrow \text{Trim}(g_y(x_t, y_t; \xi_{1:t}^{(i)}): i \in [M]).
   \]
5: Server computes mid-points $(\hat{x}_t, \hat{y}_t)$ as follows, and transmits them to each agent.
   \[
   \hat{x}_t \leftarrow \Pi_X(x_t - \eta \tilde{g}_x(x_t, y_t))
   \]
   \[
   \hat{y}_t \leftarrow \Pi_Y(y_t + \eta \tilde{g}_y(x_t, y_t)).
   \]
6: Each normal agent $i$ draws an i.i.d. sample $\xi_{2:t}^{(i)} \sim D_i$, and transmits $g_x(\hat{x}_t, \hat{y}_t; \xi_{2:t}^{(i)}), g_y(\hat{x}_t, \hat{y}_t; \xi_{2:t}^{(i)})$ to server.
7: Server computes robust gradients:
   \[
   \tilde{g}_x(\hat{x}_t, \hat{y}_t) \leftarrow \text{Trim}(g_x(\hat{x}_t, \hat{y}_t; \xi_{2:t}^{(i)}): i \in [M])
   \]
   \[
   \tilde{g}_y(\hat{x}_t, \hat{y}_t) \leftarrow \text{Trim}(g_y(\hat{x}_t, \hat{y}_t; \xi_{2:t}^{(i)}): i \in [M]).
   \]
8: Server computes new updates $x_{t+1}$ and $y_{t+1}$:
   \[
   x_{t+1} \leftarrow \Pi_X(x_t - \eta \tilde{g}_x(\hat{x}_t, \hat{y}_t))
   \]
   \[
   y_{t+1} \leftarrow \Pi_Y(y_t + \eta \tilde{g}_y(\hat{x}_t, \hat{y}_t)).
   \]
9: end for

III. ROBUST DISTRIBUTED EXTRA-GRADIENT

In this section, we develop the Robust Distributed Extra-Gradient (RDEG) algorithm outlined in Algorithm 1. Our algorithm evolves in discrete-time iterations $t \in [T]$, where $T$ is the total number of iterations. There are two main steps in RDEG. In the first step, the server computes robust gradient estimates $(\tilde{g}_x(x_t, y_t), \tilde{g}_y(x_t, y_t))$ at the current iteration $(x_t, y_t)$ by applying a Trim operator to the gradients collected from all agents (line 4); we will describe this operator shortly. The robust gradient estimates are then used to compute a mid-point $(\hat{x}_t, \hat{y}_t)$ by performing a projected primal-dual update (line 5). In the second step, the server now computes robust gradients at the mid-point (line 7), and performs a projected primal-dual update using these gradients to generate the next iterate $(x_{t+1}, y_{t+1})$. We now describe the Trim operation.

The Trim operator in equations (5) and (7) takes as input $M$ vectors, and applies the univariate trimmed mean estimator in [11] - described in Algorithm 2 - to each coordinate of these vectors separately. To describe the trimmed mean estimator, suppose the data comprises of $M$ independent copies of a scalar random variable $Z$ with mean $\mu_Z$ and variance $\sigma_Z^2$. An adversary corrupts at most $\alpha M$ of these copies; the corrupted data-set is then made available to the estimator. The estimator splits the corrupted data set into two equal chunks, denoted by $Z_1, \ldots, Z_{M/2}$, $\tilde{Z}_1, \ldots, \tilde{Z}_{M/2}$. One of the chunks is used to compute appropriate quantile levels for truncation (line 2 of Alg. 2). The robust estimate $\hat{\mu}_Z$ of $\mu_Z$ is an average of the data points in the other chunk, with those data points falling outside the estimated quantile levels truncated prior to averaging (line 3 of Alg. 2).

Algorithm 2 Univariate Trimmed-Mean Estimator [11]

Require: Corrupted data set $Z_1, \ldots, Z_{M/2}$, $\tilde{Z}_1, \ldots, \tilde{Z}_{M/2}$, corruption fraction $\alpha$, and confidence level $\delta$.
1: Set $\epsilon = 8 \alpha + 24 \log(4/\delta) M$.
2: Let $Z_1 \leq Z_2 \leq \cdots \leq Z_{M/2}$ represent a non-decreasing arrangement of $\{Z_i\}_{i \in [M/2]}$. Compute quantiles: $\gamma = Z_{(1-\alpha)M/2}$ and $\beta = Z_{(1-\epsilon)M/2}$.
3: Compute robust mean estimate $\hat{\mu}_Z$ as follows:
   \[
   \hat{\mu}_Z = \frac{2}{M} \sum_{i=1}^{M/2} \phi_{\gamma, \beta}(\tilde{Z}_i) \phi_{\gamma, \beta}(x) = \begin{cases} \beta & x > \beta \\ x & x \in [\gamma, \beta] \\ \gamma & x < \gamma \end{cases}
   \]

The following result on the performance of Algorithm 2 will play a key role in our subsequent analysis of RDEG.

Theorem 1. [11, Theorem 1] Consider the trimmed mean estimator in Algorithm 2. Suppose $\alpha \in (0, 1/16)$, and let $\delta \in (0, 1)$ be such that $\delta \geq 4e^{-M/2}$. Then, there exists an universal constant $c$, such that with probability at least $1 - \delta$,
\[
|\hat{\mu}_Z - \mu_Z| \leq c \sigma_Z \left( \sqrt{\alpha} + \sqrt{\log(1/\delta) / M} \right).
\]

In the next section, we will provide rigorous guarantees on the performance of our proposed algorithm RDEG.

IV. PERFORMANCE GUARANTEES FOR RDEG

Before stating our main results, we first make a standard smoothness assumption on the function $f(x, y)$.

Assumption 1. There exists a constant $L > 0$ such that the following holds for all $x_1, x_2 \in X$, and all $y_1, y_2 \in Y$:
\[
\|\nabla_x f(x_1, y_1) - \nabla_x f(x_2, y_2)\| \leq L (\|x_1 - x_2\| + \|y_1 - y_2\|), \\
\|\nabla_y f(x_1, y_1) - \nabla_y f(x_2, y_2)\| \leq L (\|x_1 - x_2\| + \|y_1 - y_2\|).
\]

We now define a few key quantities that will show up in our main results. Let $\sigma_x = \sqrt{\sum_{j \in [n]} \sigma^2_x(j)}$, $\sigma_y = \sqrt{\sum_{k \in [m]} \sigma^2_y(k)}$, and $\sigma = \max\{\sigma_x, \sigma_y\}$. Moreover, let $d = \max\{n, m\}$, and $D = \max\{D_x, D_y\}$, where $D_x$ and $D_y$ are the diameters of the sets $X$ and $Y$, respectively. With the above notations in place, we state our first main result that provides a bound on the primal-dual gap $\bar{D}_{T} \triangleq \max_{y \in Y} f(\bar{x}_T, y) - \min_{x \in X} f(x, \bar{y}_T)$, where
\[
\bar{x}_T = (1/T) \sum_{t \in [T]} 1_t, \text{ and } \bar{y}_T = (1/T) \sum_{t \in [T]} 1_t
\]
Theorem 2. Suppose Assumption 1 holds, the fraction $\alpha$ of corrupted devices satisfies $\alpha \in [0, 1/16)$, and the number of agents $M$ is sufficiently large: $M \geq 48 \log(16dT^2)$. Then, with a step-size $\eta$ satisfying $\eta \leq 1/(2L)$, and the confidence parameter $\delta$ in Algorithm 2 set to $\delta = 1/(4dT^2)$, $\text{RDEG}$ guarantees the following with probability at least $1 - 1/T$:

$$
\phi_T \leq \frac{D^2}{\eta T} + \tilde{O}\left(\sigma D \left(\sqrt{\alpha} + \sqrt{\frac{1}{M}}\right)\right).
$$

(9)

Discussion. Theorem 2 tells us that with high probability, the primal-dual gap $\phi_T$ converges to a ball of radius $\tilde{O}\left(\sigma D \left(\sqrt{\alpha} + \sqrt{1/M}\right)\right)$ at a $O(1/T)$ rate.\(^5\) Notably, the primal-dual gap is zero if and only if $(\tilde{x}_T, \tilde{y}_T)$ is a saddle point of $f(x, y)$ over the set $\mathcal{X} \times \mathcal{Y}$. Thus, $\text{RDEG}$ provably generates approximate saddle points. The following result is one of the main implications of Theorem 2.

Corollary 1. Suppose the conditions in Theorem 2 hold. Then, $\text{RDEG}$ guarantees the following with probability at least $1 - 1/T$:

$$
|f(\tilde{x}_T, \tilde{y}_T) - f(x^*, y^*)| \leq \frac{D^2}{\eta T} + \tilde{O}\left(\sigma D \left(\sqrt{\alpha} + \sqrt{1/M}\right)\right).
$$

(10)

Corollary 1 tells us that with high probability, the function values $f(\tilde{x}_t, \tilde{y}_t)$ of the averaged iterates generated by $\text{RDEG}$ converge to the saddle-point value $f(x^*, y^*)$ up to an error floor of $\tilde{O}\left(\sigma D \left(\sqrt{1/M}\right)\right)$, at a $O(1/T)$ rate. There are several key messages from this result. First, in the absence of adversaries (i.e., when $\alpha = 0$), the classical extra-gradient algorithm with a constant step-size would yield convergence to the saddle-point value with an error floor of $\tilde{O}(\sigma \sqrt{1/M})$ at a $O(1/T)$ rate. Thus, the biasing effect of the adversaries, the statistical performance of $\text{RDEG}$ is near-optimal. Second, the additive biasing effect due to adversarial corruption shows up even in the context of stochastic minimization [12]. In fact, the authors in [12] argue that an additive biasing effect of order $\tilde{O}(\alpha)$ is unavoidable, albeit for the minimization setting. This is all to say that the dependence of our rate on the corruption level in Eq. (10) is only to be expected. Third, when the corruption level is small, the benefit of collaboration is evident from the second term in Eq. (10): the variance $\sigma$ arising from the noise term is effectively reduced by a factor of $\sqrt{M}$ due to the averaging effect of the normal agents. This effect will be aptly demonstrated by the simulations in Section VI.

We now turn to the goal of achieving faster convergence rates than those in Theorem 2. To that end, we study the performance of the $\text{RDEG}$ algorithm for strongly convex-strongly concave (SC-SC) functions. Accordingly, we first make the following assumption on $f(x, y)$.

Assumption 2. The function $f(x, y)$ is $\mu$-strongly convex-$\mu$-strongly concave (SC-SC) over $\mathcal{X} \times \mathcal{Y}$, i.e., for all $x_1, x_2 \in \mathcal{X}$ and $y_1, y_2 \in \mathcal{Y}$, the following holds:

$$
f(x_2, y_1) \geq f(x_1, y_1) + \langle \nabla_x f(x_1, y_1), x_2 - x_1 \rangle + \frac{\mu}{2} \|x_2 - x_1\|^2,
$$

and

$$
f(x_1, y_2) \leq f(x_1, y_1) + \langle \nabla_y f(x_1, y_1), y_2 - y_1 \rangle - \frac{\mu}{2} \|y_2 - y_1\|^2.
$$

For $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, define $z \triangleq [x; y]$. We have the following result for functions satisfying Assumption 2.

Theorem 3. Suppose Assumptions 1 and 2 hold in conjunction with the assumptions on $\alpha$ and $M$ in Theorem 2. Then, with $\delta = 1/(4dT^2)$ and step-size $\eta \leq 1/(4L)$, $\text{RDEG}$ guarantees the following with probability at least $1 - 1/T$:

$$
\|z^* - z_{T+1}\|^2 \leq 2e^{-\frac{dT}{10}} D^2 + \tilde{O}\left(\frac{\sigma D \sqrt{\kappa \eta}}{L} \left(\sqrt{\alpha} + \sqrt{1/M}\right)\right),
$$

(11)

where $\kappa = \mu/L$.

Theorem 3 says that for smooth strongly convex-strongly concave functions, the iterates generated by $\text{RDEG}$ converge linearly to a ball around the saddle point $(x^*, y^*)$ with high probability. The size of the ball is dictated by the second term in Eq. (11).

Remark 1. (Comments on $\alpha$) The requirement that the fraction of corruption $\alpha \in [0, 1/16)$ in our results is inherited from the analysis of the trimmed mean estimator in [11]. One can potentially tolerate a larger fraction of corruption (up to $\alpha < 1/2$) by using the robust estimators in [12]. However, this would likely come at a price: the authors in [12] impose additional statistical assumptions on the partial gradients; we do not make such assumptions.

Remark 2. (Comments on $M$) In our results, we need the number of agents $M$ to scale with $\log(dT)$. We note that similar conditions show up in the context of adversarially-robust distributed statistical learning; see, for instance, [12] and [25]. In fact, the covering argument in [12] requires $M$ to scale linearly with the model dimension $d$. By avoiding such an argument in our analysis, we can get by with a far milder logarithmic dependence on $d$. As an example, for $d = 100$, and number of iterations $T = 10^2$ (which should suffice for all practical purposes), $\log(dT) \approx 12$. This is a very reasonable requirement for large-scale computing systems where the number of devices is of the order of thousands. Furthermore, with $T = 10^2$, our guarantees in Theorems 2 and 3 hold with probability $1 - 1/T \approx 1$.

V. PROOF SKETCH OF THEOREM 2

In this section, we provide a sketch of the proof of Theorem 2. Due to space constraints, detailed proofs of our main results (including that of Theorem 3) are omitted here, but can be found in [28]. Essentially, our proofs comprise of a perturbation analysis of the extra-gradient algorithm, where the perturbations arise due to adversarial corruption. As the starting point of such an analysis, we establish some simple relations in the following lemma.
Lemma 1. For the RDEG algorithm, the following inequalities hold for all $t \in [T]$, $x \in X$, and $y \in Y$:
\[2\eta \langle \tilde{g}_x(x_t, y_t), \tilde{g}_y(x_t, y_t) \rangle = \max_{y \in Y} f(\bar{x}_t, y) - \min_{x \in X} f(x, \bar{y}_t)\]

Using the previous result, our next goal is to track the progress made by the mid-point vector $(\bar{x}_t, \bar{y}_t)$ in each iteration, as a function of the errors introduced by adversarial corruption. To that end, for each $\bar{x} \in X$ and $\bar{y} \in Y$, we define the following error vectors:
\[e_x(\bar{x}, \bar{y}) \triangleq \bar{g}_x(\bar{x}, \bar{y}) - \nabla_x f(x, \bar{y});
\]
\[e_y(\bar{x}, \bar{y}) \triangleq \bar{g}_y(\bar{x}, \bar{y}) - \nabla_y f(\bar{x}, \bar{y}).\]

We have the following key lemma.

Lemma 2. Suppose Assumption 1 holds and $\eta \leq 1/(2L)$. For the RDEG algorithm, the following then holds for all $t \in [T]$, $x \in X$, and $y \in Y$:
\[\eta \langle \nabla_x f(\bar{x}_t, \bar{y}_t), \bar{x}_t - x \rangle - \eta \langle \nabla_y f(\bar{x}_t, \bar{y}_t), \bar{y}_t - y \rangle \leq \frac{\eta}{2} \left( \| x - \bar{x}_t \|^2 + \| y - \bar{y}_t \|^2 \right) + \eta \Delta\]

where $\Delta = c\sigma \sqrt{g^2 + \log(4dT^2)}$.

For the RDEG algorithm, we have:
\[\mathbb{P}(\tau \geq 1 - \frac{1}{T^2}) \text{ for each } t \in [T].\]

Proof. By defining certain "good" events:
\[\mathcal{G}_{x,t} \triangleq \{ \| e_x(x_t, y_t) \| \leq \Delta \}, \mathcal{G}_{y,t} \triangleq \{ \| e_y(x_t, y_t) \| \leq \Delta \},\]
\[\mathcal{G}_{x,t} \triangleq \{ \| e_x(x_t, \bar{y}_t) \| \leq \Delta \}, \mathcal{G}_{y,t} \triangleq \{ \| e_y(\bar{x}_t, \bar{y}_t) \| \leq \Delta \}.\]

To analyze the probability of occurrence of the above events, we need to next define an appropriate filtration. Accordingly, let $\mathcal{F}_t$ denote the sigma field generated by $\{ x_k, y_k \}_{k \in [t]}$ and $\{ \tilde{x}_k, \tilde{y}_k \}_{k \in [t-1]}$; and $\mathcal{F}_t$ denote the sigma field generated by $\{ x_k, y_k \}_{k \in [t]}$ and $\{ \tilde{x}_k, \tilde{y}_k \}_{k \in [t]}$. From definition, we have
\[\mathcal{F}_1 \subset \mathcal{F}_2 \subset \cdots \subset \mathcal{T} \subset \mathcal{F}_T.\]

Clearly, $(x_t, \bar{y}_t)$ is $\mathcal{F}_t$-measurable. Thus, conditioned on $\mathcal{F}_t$, for each coordinate $j \in [n]$, the data set $\{ [g_y(x_t, y_t; \xi_i^{(t)})]_j : i \in [M] \}$ has the following properties: (i) at most $\alpha M$ of the samples are corrupted; and (ii) the uncorrupted samples are i.i.d. scalar random variables with mean $[\nabla_x f(x_t, \bar{y}_t)]_j$ and variance bounded above by $\sigma_x^2(j)$.

The proof of Theorem 2 is a fairly simple consequence of Lemma’s 1, 2, and 3. In particular, it follows by conditioning on the clean event $\mathcal{H} = \bigcap_{t \in [T]} \mathcal{H}_t$, where $\mathcal{H}_t$ is as defined in Lemma 3, and exploiting the convex-concave property of $f(x, y)$ in a standard way. For details, see [28].

VI. Simulations

In this section, we study a specific instance of problem (1), namely a bilinear game of the following form:
\[
\min_{\| x \|_2 \leq \rho} \max_{\| y \|_2 \leq \rho} f(x, y) \triangleq \mathbb{E}[x^T A y + 2b + \zeta x - 2c + \zeta^T y].
\]

Here, $x, y, b, c \in \mathbb{R}^{10}$, $A \in \mathbb{R}^{10 \times 10}$, and $\rho = 100$. The parameters $A, b, c$ are fixed, and $\zeta \sim N(0, \sigma^2 I)$. As our measure of performance, we consider the instantaneous primal-dual gap
\[\phi_t = \max_{y \in Y} f(x_t, y) - \min_{x \in X} f(x, \bar{y}_t).\]

We simulate two algorithms: the vanilla extra-gradient algorithm that does not...
account for adversaries, and the proposed RDEG algorithm. In Fig. 2(a), we plot the performance of these algorithms with \( \alpha = 0.06 \), \( M = 100 \), and \( \sigma^2 = 10 \). We observe that even a small number of Byzantine workers can cause the extra-gradient algorithm to diverge from the saddle point. In Fig. 2(b), with \( M = 100 \) and \( \sigma^2 = 10 \), we explore the impact of varying the corruption fraction \( \alpha \). Complying with Theorem 2, the error floor of RDEG increases as a function of \( \alpha \). Next, in Fig. 2(c), to demonstrate the benefit of collaboration, we fix \( \alpha = 0.06 \) and \( \sigma^2 = 10 \), and plot the performance of RDEG as a function of the number of agents \( M \). As expected, by increasing \( M \), RDEG converges to a smaller ball around the saddle point, highlighting the benefit of collaboration in reducing the variance of the noise model. Finally, in Fig. 2(d), we fix \( M = 100 \) and \( \alpha = 0.06 \), and change the variance of the noise \( \sigma^2 \). We observe that increasing \( \sigma^2 \) leads to a higher error-floor. Importantly, all of the above plots verify the bound in Theorem 2.

**VII. CONCLUSION**

We studied the problem of distributed min-max learning under adversarial agents for the first time. By exploiting recent ideas from robust statistics, we developed a novel robust distributed extra-gradient algorithm. For both smooth convex-concave and smooth strongly convex-strongly concave functions, we showed that with high probability, our proposed approach guarantees convergence to approximate saddle points at near-optimal statistical rates.

**REFERENCES**

