ByGARS: Byzantine SGD with Arbitrary Number of Attackers

Jayanth R. Regatti*
Department of ECE
The Ohio State University
Columbus, OH 43210
regatti.1@osu.edu

Hao Chen
Department of ECE
The Ohio State University
Columbus, OH 43210
chen.6945@osu.edu

Abhishek Gupta
Department of ECE
The Ohio State University
Columbus, OH 43210
gupta.706@osu.edu

Abstract

We propose two novel stochastic gradient descent algorithms, ByGARS and ByGARS++, for distributed machine learning in the presence of any number of Byzantine adversaries. In these algorithms, reputation scores of workers are computed using an auxiliary dataset at the server. This reputation score is then used for aggregating the gradients for stochastic gradient descent. We use two-timescale stochastic approximation theory to show that using these reputation scores for gradient aggregation is robust to any number of multiplicative noise Byzantine adversaries. The computational complexity of ByGARS++ is the same as the usual distributed stochastic gradient descent method with only an additional inner product computation in every iteration. We establish its convergence for strongly convex loss functions and demonstrate the effectiveness of the algorithms for non-convex learning problems through empirical results on MNIST and CIFAR-10 datasets against various state of the art Byzantine attacks.

1 Introduction

There has been a significant interest in devising distributed machine learning schemes in the presence of Byzantine adversaries [1, 2, 3, 4]. A certain fraction of the workers are assumed to be adversarial; instead of sending the actual gradients computed using a randomly sampled mini batch to the server, the adversarial workers send arbitrary or potentially adversarial gradients that could derail the optimization at the server. Several techniques have been proposed to secure the gradient aggregation against adversarial attacks under different settings such as gradient encoding [5], asynchronous updates [6, 7], heterogeneous datasets [8], decentralized learning [9], [10, 11] and federated Learning [12, 13]. There has also been some work in developing attack techniques that break existing defenses [12, 14, 15]. One of the main assumptions in past studies about Byzantine attacks in machine learning is that the number of adversarial workers is less than half of the total number of workers. These approaches relied on techniques like majority voting, geometric median, median of means, coordinate wise median, etc. to aggregate gradients at the server. The fundamental reason for this assumption was that the underlying concept of geometric median (often used for robust aggregation) has a breakdown point of 0.5 [2]. In other words, it yields a robust estimator as long as less than half of the data (used for aggregation) is corrupted.

The assumption that less than half of the workers are adversarial might not be practical. A more challenging problem is to ensure convergence even in the presence of a large number of adversaries. Some prior works that address this case are [16, 17, 18, 7], among a few others. In these works, the server has some auxiliary data, which is used to identify adversarial workers and the gradients obtained from such workers are discarded at the server. In contrast, we address the issue of an arbitrary

*corresponding author

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Table 1: Summary of various attacks that ByGARS or ByGARS++ is robust to. Extensive simulations suggests that the proposed algorithms are resilient to most of the state-of-the-art attacks with any number of Byzantine adversaries; the checkmarks in the right column indicates that in simulations, we have found either ByGARS or ByGARS++ is able to perform SGD under a wide range of initial conditions. Here, f denotes the fraction of Byzantine adversaries in the system, with $f = 1$ implying that all workers are adversarial.

Type	Attack	Fraction of Adversaries f		
		$f < 0.5$	$f \in [0.5, 1)$	$f = 1$
Omniscient / Collusion	Inner Product Manipulation [14]	√	√	
Omniscient / Collusion	LIE [15]	√	-	
Omniscient / Collusion	OFOM [12]	√	√	
Omniscient / Collusion	PAF [12]	√	√	
Local / Failure	Sign Flip/Reverse Attack	√	√	√
Local / Failure	Random Sign Flip Attack	√	√	√
Local / Failure	Gaussian Attack [3]	√	√	
Local / Failure	Constant Attack [8]	√	√	
Data Poisoning	Label Flipping	√	√	
Mixed Attacks	Multiple types of attacks	√	√	√

number of adversaries by allowing the server to use an auxiliary dataset (a small dataset drawn from the same distribution as the training data) to compute the *reputation score* of each worker that is used for gradient aggregation. By assuming that the worker behavior is stationary, the *reputation score* of a worker signifies how relevant the corresponding gradient direction is to the optimization problem. As we will see, we achieve robustness to any number of adversaries by letting the reputation score of the workers take negative values.

If a worker consistently sends the gradient scaled with a negative value, then the reputation score accumulates negative values, since the inner product is negative in expectation. Therefore by multiplying the received gradient with $q_{t,j}$ we can recover the actual direction. When the parameters are far away from the optima, we compute the reputation score of each stochastic gradient by taking an inner product with the stochastic gradient computed on the auxiliary data.

Our Contributions: In this work, we do not identify the adversarial workers and discard their gradients; instead, we use the auxiliary data at the server to compute a reputation score for the workers, and use the reputation scores to weigh the gradients of the workers to carry out the parameter update. Our primary contributions are:

1. We show that our algorithm is Byzantine tolerant [14] to an arbitrary number of attackers.
2. We use two time-scale stochastic approximation theory to establish the convergence of the proposed algorithm under reasonable assumptions (with strongly convex loss function).
3. In the previous works, the algorithms filtered out the adversaries and reduced the effective data size for training, which affected the test error and the generalization ability of the trained model. By using reputation scores for aggregation, we improve test error and generalization of trained model, especially against local attacks or consistent. system failures (such as corrupted bits, flipped bits, etc). Empirical evidence suggests that our algorithm is robust to a large class of Byzantine attacks (summarized in Table 1).

2 Problem setup

We consider distributed machine learning with a parameter server - worker setup. The parameter server maintains the model parameters, and updates the parameters with gradients received from the workers. We denote the model parameters by $\mathbf{w} \in \mathcal{W} \subset \mathbb{R}^d$, and the number of workers by m. We assume that each worker j has access to dataset, $D_j := \{x_i^j, y_i^j\}_{i=1}^{n_j} \sim \mathcal{D}$, where $N = \sum_j n_j$ is the total number of data points. In the Federated Learning scenario, this translates to each worker having its own dataset, which is not shared with anyone. Given a loss function $f(\cdot, x, y) : \mathbb{R}^d \to \mathbb{R}$, $x, y \sim \mathcal{D}$, the objective is to minimize the population loss $F : \mathbb{R}^d \to \mathbb{R}$, $\mathbf{w}_* = \arg\min_{\mathbf{w} \in \mathbb{R}^d} F(\mathbf{w}) := \mathbb{E}_{x,y \sim \mathcal{D}}[f(\mathbf{w}, x, y)]$.

We denote the true gradient of the population loss at $\mathbf{w}_t$ by $\nabla F(\mathbf{w}_t)$. A good worker samples a subset of the data $\mathcal{D}_{j,t} \subset \mathcal{D}_j$, and computes a stochastic gradient $\tilde{h}_{t,j} := \frac{1}{|\mathcal{D}_{j,t}|} \sum_{x,y \in \mathcal{D}_{j,t}} \nabla f(\mathbf{w}_t, x, y)$. The good workers communicate the stochastic gradient $h_{t,j} := \tilde{h}_{t,j}$ to the server, where as adversarial workers inject a random multiplicative noise $\tilde{\kappa}_i$ and send $h_{t,j} := \tilde{\kappa}_i \tilde{h}_{t,j}$, where $\tilde{\kappa}_i$ is a random variable that the adversary draws at each time step from a fixed attack distribution. We assume that this attack distribution remains fixed for the adversary throughout the training. We denote the set of gradients received by the server as $H_t^T = [h_{t,1}, \cdots, h_{t,m}] \in \mathbb{R}^{d \times m}$. Note that we assume a synchronous setting here, i.e. all the workers communicate the gradients at the same time to the server. We assume that the server has access to an auxiliary dataset $D_{aux} := \{x_i, y_i\}_{i=1}^n \sim \mathcal{D}$. The server can sample a subset $\xi_{aux,t}$ of the auxiliary dataset and compute auxiliary loss $L_t(\mathbf{w}) = \frac{1}{|\xi_{aux,t}|} \sum_{(x,y) \in \xi_{aux,t}} f(\mathbf{w}, x, y)$, such that $\mathbb{E}[\nabla L_t(\mathbf{w}_t)] = \nabla F(\mathbf{w}_t)$.

3 Algorithm

In an ideal environment, where all the workers are benign, the gradient aggregation function simply averages the received stochastic gradients and uses the averaged gradient to update the parameters. However, in the presence of adversaries, to compute a meaningful estimate of the gradient, we maintain a reputation score $q_{t,j}$ for each worker j at the server. Since the adversary can be of any type, the reputation scores can take any real value. Suppose, at time t, the reputation score vector is $\mathbf{q}_t = [q_{t,1}, \ldots, q_{t,m}]^T$ and the received gradients are H_t, then the weighted aggregation of the gradients with the reputation score is $H_t^T \mathbf{q}_t = \sum_{i=1}^m q_{t,i} h_{t,i}$. If we have a good estimate of the reputation score $\mathbf{q}_t$ (say we know $\boldsymbol{\kappa}$ and set $q_{t,i} = \frac{1}{\kappa_i}$ for all i) at each time t, then $-H_t^T \mathbf{q}_t$ is a descent direction, then no adversary can affect the training provided that sufficiently small step size is used and the adversaries satisfy certain assumptions. The problem now is to compute a good reputation score, without the knowledge of $\boldsymbol{\kappa}$, for the workers using only the gradients sent to the server. The core of our algorithm lies at the auxiliary dataset available to the worker. It is a reasonable assumption to make since in practical scenarios, it is not difficult to procure a small amount of clean auxiliary data without violating the privacy of the worker's data. This data can be taken from publicly available datasets (that match the distribution of the data available at the workers), from prior data leaks (that is now publicly available), or data given voluntarily by workers. Making use of the availability of an auxiliary dataset and the stationary behavior of the workers, we propose two algorithms to compute the reputation score of the workers.

3.1 ByGARS: Byzantine Gradient Aggregation using Reputation Scores

We start with an initial reputation score of $\mathbf{q}_0 = \mathbf{0} \in \mathbb{R}^m$, and iteratively improve the estimate of the reputation score. At step t, we perform a *pseudo update* to $\mathbf{w}_t$ (γ_t is a step size parameter) as

$$\hat{\mathbf{w}}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_t \tag{1}$$

If $\mathbf{q}_t$ is a good reputation score and γ_t is sufficiently small, then $-H_t^T \mathbf{q}_t$ is a descent direction and thus $F(\hat{\mathbf{w}}_{t+1})$ must be lower in value than $F(\mathbf{w}_t)$ or other points in its neighborhood. However, we neither have access to the true function F nor the data from the workers. Instead, we have a small auxiliary dataset that is drawn from the same distribution as the data at the workers. This auxiliary dataset allows us to construct the loss function $L_t(\cdot)$ (see Section 2), and we can solve the following meta optimization problem to compute a better reputation score $\mathbf{q}_{t+1} = \arg\min_{\mathbf{q} \in \mathbb{R}^m} L_t(\mathbf{w}_t - \gamma_t H_t^T \mathbf{q})$. We solve this meta optimzation using iterative *meta updates*. We compute the gradient of the auxiliary loss L_t evaluated at $\widehat{\mathbf{w}}_{t+1}$, and perform a first order update on $\mathbf{q}_t$. The *meta update* is given by

$$\mathbf{q}_t \leftarrow \mathbf{q}_t - \alpha_t \frac{d}{d\mathbf{q}_t} L_t(\mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_t) = \mathbf{q}_t + \alpha_t \gamma_t H_t \nabla L_t(\hat{\mathbf{w}}_t) \tag{2}$$

The updated reputation score is used to find the updated gradient aggregation $H_t^T \mathbf{q}_t$. At each step t, the algorithm proceeds by successively applying the *pseudo update* (eq 1) and *meta update* to $\mathbf{q}_t$ (eq 2) for k iterations (or until a stopping criteria is reached, such as sufficient descent) to obtain $\mathbf{q}_{t+1}$ before finally performing an actual update to $\mathbf{w}_t$ as $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_{t+1}$.

a ByGARS

$\mathbf{w}_0$ initialized randomly and sent to workers
$\mathbf{q}_0 = \mathbf{0}$
for $t = 1, \cdots, T$ **do**
  $\mathbf{q}_{t+1}^0 = \mathbf{q}_t$; receive H_t^T
  for $i = 1, \cdots, k$ **do**
    $\widehat{\mathbf{w}}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_{t+1}^{i-1}$
    $\mathbf{q}_{t+1}^i \leftarrow \mathbf{q}_{t+1}^{i-1} + \alpha_t \gamma_t H_t \nabla L_t(\widehat{\mathbf{w}}_{t+1})$
  end for
  $\mathbf{q}_{t+1} = \mathbf{q}_{t+1}^k$
  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_{t+1}$
  Send $\mathbf{w}_{t+1}$ to workers
end for
Return $\mathbf{w}_{T+1}$

b ByGARS++

$\mathbf{w}_0$ initialized randomly and sent to workers
$\mathbf{q}_0 = \mathbf{0}$
for $t = 1, \cdots, T$ **do**
  Receive $H_t^T = [h_{t,1}, \cdots, h_{t,m}]$
  Compute auxiliary gradient $\nabla L_t(\mathbf{w}_t)$
  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_t$
  Send $\mathbf{w}_{t+1}$ to workers
  $\mathbf{q}_{t+1} \leftarrow (1 - \alpha_t)\mathbf{q}_t + \alpha_t H_t \nabla L_t(\mathbf{w}_t)$
end for
Return $\mathbf{w}_{T+1}$

Figure 1: The proposed algorithms (a) ByGARS, and (b) ByGARS++

3.2 ByGARS++: Faster ByGARS

ByGARS has an additional computational overhead due to multiple parameter updates and multiple gradient computations to update the reputation score in the meta updates. This increased computation at the server keeps the workers idle and waiting, thus negating the computational speed-up achieved from distributed learning. In order to overcome this limitation, and driven by the motivation of ByGARS, we propose a variant which is computationally cheaper yet efficient.

We propose ByGARS++, in which we avoid computing multiple *pseudo updates* $\hat{\mathbf{w}}_t$ used for performing *meta updates*, by simulataneously updating $\mathbf{w}_t, \mathbf{q}_t$ as given by eq (3). Note that we perform an update to $\mathbf{q}_t$ using the auxiliary gradients evaluated at $\mathbf{w}_t$ (and not at $\hat{\mathbf{w}}_t$).

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_t, \qquad \mathbf{q}_{t+1} \leftarrow (1 - \alpha_t)\mathbf{q}_t + \alpha_t H_t \nabla L_t(\mathbf{w}_t) \tag{3}$$

In this case, the reputation score of each worker is updated using only the inner product between the gradient sent by the worker, and the auxiliary gradient, both evaluated at $\mathbf{w}_t$. The only additional computation as compared to traditional distributed SGD is the udpate of $\mathbf{q}_t$ which takes $\mathcal{O}(md)$ time. However, the server can update $\mathbf{q}_t$ when the workers are computing the gradients for next time step, therefore ByGARS++ has the same computational complexity as traditional distributed SGD.

When the parameters are closer to the optima, the inner product value is random [19] (since the directions of the stochastic gradients are random), and hence does not contribute to the reputation score. Therefore, we employ a decaying learning rate schedule for both γ_t and α_t. Thus, by the time the parameters are close enough to the optima or a flat region (in non-convex settings), the learning rates would have decayed significantly. This enables the reputation score to accumulate over time and converge; therefore, the score is robust to the noisy inner products near the optima.

4 Convergence of ByGARS++

We now analyze the convergence of ByGARS++, which follows the update equation (3).

Assumption 1. *The population loss F is c-strongly convex, with $\mathbf{w}_*$ as the unique global minimum, such that $\nabla F(\mathbf{w}_*) = 0$. Further, ∇F is a locally Lipschitz function with bounded gradients.*

Assumption 2. *The Byzantine adversaries corrupt the gradients using multiplicative noise. If the worker i computes a stochastic gradient $\tilde{h}_{t,i}$ which is an unbiased estimate of $\nabla F(\mathbf{w}_t)$, the worker sends $h_{t,i} := \tilde{\kappa}_{t,i}\tilde{h}_{t,i}$ to the parameter server, where $\tilde{\kappa}_{t,i}$ is an iid multiplicative noise with mean κ_i and finite second moment. The random noise satisfies $|\tilde{\kappa}_{t,i}| \leq \kappa_{\max}$ almost surely for all the workers. The workers have the following types:*

1. *Benign worker:* $\mathbb{E}h_{t,i} = \nabla F(\mathbf{w}_t)$ *with* $\kappa_i = 1$*;*
2. *Scaled adversary:* $\mathbb{E}h_{t,i} = \kappa_i \nabla F(\mathbf{w}_t)$*, where* κ_i *is a real number (negative or positive);*
3. *Random adversary:* $\mathbb{E}h_t = 0$*,where adversary sends random gradients with mean 0 (i.e.* $\kappa_i = 0$*).*

There is at least one benign or scaled adversary with $\kappa_i \neq 0$ *among the workers. Further, we assume the adversaries' noise distributions do not change with time.*

This is a reasonable assumption as the goal of the attacker is to derail the training progress by corrupting the aggregate gradient so that it is not a descent direction. This adversary model is used in several works including [20, 8, 7]. However, we also show empirical results for different types of attacks (summarized in Table 1). The following theorems capture the main result of this paper.

Theorem 1. *If Assumption 2 is satisfied, then ByGARS++ is DSSGD-Byzantine Tolerant (Def. 4 in [14]), that is,* $\mathbb{E}[\langle \nabla F(\mathbf{w}_t), H^T q_t \rangle] \geq 0$*.*

Thus, by accruing reputation scores using the inner products for every worker, the sign of $q_{t,i}$ is the same as that of κ_i. Due to this reason, even gradients of some adversaries can be helpful in training if multiplied with an appropriate reputation score. When the loss function is strongly convex and smooth, we can prove an even stronger result which follows from theory of two timescale stochastic approximation [21],

Theorem 2. *Suppose that* $\{\alpha_t\}, \{\gamma_t\}$ *are diminishing stepsizes, that is,* $\sum \alpha_t = \infty, \sum \gamma_t = \infty, \sum \alpha_t^2 < \infty, \sum \gamma_t^2 < \infty$*, with* $\gamma_t/\alpha_t \to 0$ *as* $t \to \infty$*. If Assumptions 1 and 2 are satisfied and* $\sup_t \|\mathbf{w}_t\|, \sup_t \|\mathbf{q}_t\| < \infty$*, then* $\{\mathbf{w}_t\}$ *generated by ByGARS++ converges almost surely* $\mathbf{w}_*$*.*

5 Simulations

We present the results of our algorithms on MNIST [22], CIFAR-10 [23] for multi-class classification using supervised learning. We used LeNet [24] for MNIST, and a two convolutional layer CNN for CIFAR-10. For each dataset, we set aside a small auxiliary dataset of size 250 (sampled randomly from the train set) at the server, and the remaining data is distributed uniformly to the workers.

The summary of the attacks used in this work is given in Table 1. In the **Omniscient / Collusion** attacks, the adversaries have complete information about all other workers including the benign ones, or only about the other adversaries. In particular we use LIE attack [15], OFOM, PAF [12], Inner Product attack [14] multiply the empirical mean of benign gradients with a negative value. In **Local** attacks, the attacker doesn't have any information about the other workers. Instead, the worker either sends an arbitrary gradient to the server or uses the gradient it computed. We use Gaussian Attack [3], Constant attack [8], Sign flipping attack [8, 20, 17]. In addition to the sign flipping attack, we propose a *random* sign flip attack, where at each iteration the adversary draws a real number from a fixed distribution multiplies with the local gradient and sends to the server. We use label flipping attack (**Data Poisoning**), where for example in MNIST dataset, the attacker maps the labels as $l \to (9 - l)$ for $l \in \{0, \cdots, 9\}$, and uses these labels for computing the gradients. Note that, label flipping attack is also a Local attack. In addition to these attacks, we propose a *Mixed Attack* where there can be multiple Local attacks and Data Poisoning attacks. In our experiments, we used one benign worker, and 7 adversaries of different Local and Data Poisoning attacks in *Mixed Attack*. Throughout the paper, we will assume 8 workers and compare our algorithms against a varying number of adversaries, in particular 0 (*No Attack*), 3, 6 and 8 (*All adversaries*). We consider the case of all 8 adversaries for Sign Flipping attack. As we will see, by allowing negative reputation scores for these workers, our algorithms will achieve similar performance as that of *No Attack*.

Existing Byzantine resilient algorithms that rely on filtering out adversarial gradients can at best remove all the adversaries, and use only the benign gradients. For this reason, we compare our algorithm against a model trained only using the data at non-adversarial workers, referred to as *Baseline*. In addition to this, for illustration purposes, we also consider plain averaging of all gradients (no defense) denoted by *Average*, and coordinate-wise median [25] denoted by *Median* in our empirical analysis.

Discussion We can observe from Figure 2 that both ByGARS and ByGARS++ achieve Byzantine robustness against most of the threat models used under varying number of adversaries. We used the same learning rate, learning rate decay for all the models evaluated on each dataset, with the only

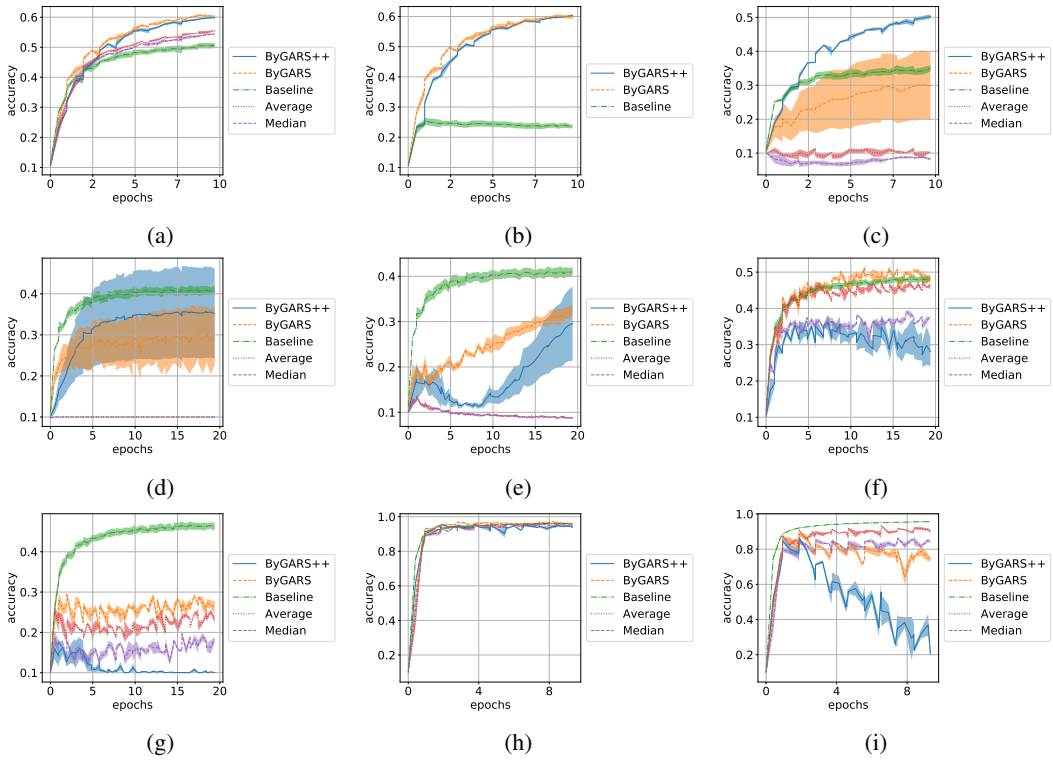

Figure 2: Top-1 accuracy for models trained on CIFAR-10 unless otherwise specified. (2a) *No Attack*, (2b) 8 Sign Flip attackers, (2c) *Mixed Attack*, (2d) 6 Constant Attack attackers, (2e) 6 Label Flip attackers, (2f) 3 LIE attackers, (2g) 4 LIE attackers, (2h) 3 LIE attackers data (MNIST), (2i) 4 LIE attackers data (MNIST). We omit the results for other attacks which were easily defended

difference being the meta learning rate and meta learning rate decay schedules for both ByGARS, and ByGARS++. From Fig 2a it is evident that there is no trade-off in employing our algorithm in the case of *No Attack*. It is important to note that, one would expect the *Baseline* to be the best in all scenarios. However, we point out that by using the reputation scores, we are directly affecting the step size of each update performed, and hence it is not surprising to observe that our algorithms perform better than the *Baseline* under *No Attack* or weaker adversary models such as Sign Flip. Note that the robustness of our algorithm comes from the fact that we did not design the defense based on criteria such as norm, or majority ideas (against which most attacks were devised), instead our algorithm aims to find a descent direction, and hence the superior performance across a range of attacks. However, while ByGARS is robust against LIE attack when < 0.5 fraction adversaries (Fig 2f, 2h, and reasonably well with $= 0.5$ adversaries (Fig 2g, 2i), we observe that ByGARS++ fails miserably under the attack, which needs further investigation.

6 Conclusion

We devise a novel, Byzantine resilient, stochastic gradient aggregation algorithm for distributed machine learning with arbitrary number of adversarial workers. This is achieved by exploiting a small auxiliary dataset to compute a reputation score for every worker, and the scores are used to aggregate the workers' gradients. We show that even gradients from adversarial workers can be useful to the training if multiplied with an appropriate reputation score. We showed that under reasonable assumptions, ByGARS++ converges to the optimal solution using a result from two timescale stochastic approximation theory [21]. Through simulations, we showed that the proposed algorithms exhibit remarkable robustness property even for non-convex problems under a wide range of Byzantine attacks. This algorithm can be extended to learning from heterogeneous datasets, learning under privacy constraints, poisoned data attacks, non-stationary attacks, etc. We leave such analyses for a future work.

Acknowledgements

The authors thank ARPA-E NEXTCAR program and ARO Grant W911NF1920256 for supporting the research. Results presented in this poster were obtained using the Chameleon testbed supported by the National Science Foundation.

References

[1] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 4613–4623, 2018.

[2] Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):1–25, 2017.

[3] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pages 119–129, 2017.

[4] Nirupam Gupta and Nitin H Vaidya. Byzantine fault-tolerant parallelized stochastic gradient descent for linear regression. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 415–420. IEEE, 2019.

[5] Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Draco: Byzantine-resilient distributed training via redundant gradients. *arXiv preprint arXiv:1803.09877*, 2018.

[6] Georgios Damaskinos, El Mahdi El Mhamdi, Rachid Guerraoui, Rhicheek Patra, and Mahsa Taziki. Asynchronous byzantine machine learning (the case of SGD). *arXiv preprint arXiv:1802.07928*, 2018.

[7] Cong Xie, Sanmi Koyejo, and Indranil Gupta. Zeno++: Robust fully asynchronous SGD. *arXiv preprint arXiv:1903.07020*, 2019.

[8] Liping Li, Wei Xu, Tianyi Chen, Georgios B Giannakis, and Qing Ling. Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1544–1551, 2019.

[9] Zhixiong Yang and Waheed U Bajwa. ByRDiE: Byzantine-resilient distributed coordinate descent for decentralized learning. *IEEE Transactions on Signal and Information Processing over Networks*, 5(4):611–627, 2019.

[10] Zhixiong Yang, Arpita Gang, and Waheed U Bajwa. Adversary-resilient inference and machine learning: From distributed to decentralized. *arXiv preprint arXiv:1908.08649*, 2019.

[11] El-Mahdi El-Mhamdi, Rachid Guerraoui, Arsany Guirguis, and Sebastien Rouault. SGD: Decentralized byzantine resilience. *arXiv preprint arXiv:1905.03853*, 2019.

[12] Hongyan Chang, Virat Shejwalkar, Reza Shokri, and Amir Houmansadr. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. *arXiv preprint arXiv:1912.11279*, 2019.

[13] Amit Portnoy and Danny Hendler. Towards realistic Byzantine-robust federated learning. *arXiv preprint arXiv:2004.04986*, 2020.

[14] Cong Xie, Sanmi Koyejo, and Indranil Gupta. Fall of empires: Breaking byzantine-tolerant SGD by inner product manipulation. *arXiv preprint arXiv:1903.03936*, 2019.

[15] Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. In *Advances in Neural Information Processing Systems*, pages 8635–8645, 2019.

[16] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Zeno: Byzantine-suspicious stochastic gradient descent. *arXiv preprint arXiv:1805.10032*, 2018.

[17] Richeng Jin, Xiaofan He, and Huaiyu Dai. Distributed Byzantine tolerant stochastic gradient descent in the era of big data. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2019.

[18] Xinyang Cao and Lifeng Lai. Distributed gradient descent algorithm robust to an arbitrary number of byzantine attackers. *IEEE Transactions on Signal Processing*, 67(22):5850–5864, 2019.

[19] Jerry Chee and Panos Toulis. Convergence diagnostics for stochastic gradient descent with constant learning rate. In *International Conference on Artificial Intelligence and Statistics*, pages 1476–1485, 2018.

[20] Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. signSGD with majority vote is communication efficient and fault tolerant. *arXiv preprint arXiv:1810.05291*, 2018.

[21] Vladislav B Tadic. Almost sure convergence of two time-scale stochastic approximation algorithms. In *Proceedings of the 2004 American Control Conference*, volume 4, pages 3802–3807. IEEE, 2004.

[22] Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009.

[24] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[25] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498*, 2018.