SLSGD: Secure and Efficient Distributed On-device Machine Learning

Cong Xie$^{1[0000-0002-5682-0230]}$ ✉, Oluwasanmi Koyejo$^{1[0000-0002-4023-419X]}$, and Indranil Gupta$^{1[0000-0002-9372-5937]}$

University of Illinois at Urbana-Champaign {cx2,sanmi,indy}@illinois.edu

Abstract. We consider distributed on-device learning with limited communication and security requirements. We propose a new robust distributed optimization algorithm with efficient communication and attack tolerance. The proposed algorithm has provable convergence and robustness under non-IID settings. Empirical results show that the proposed algorithm stabilizes the convergence and tolerates data poisoning on a small number of workers.

Keywords: Distributed · SGD.

1 Introduction

Edge devices/IoT such as smart phones, wearable devices, sensors, and smart homes are increasingly generating massive, diverse, and private data. In response, there is a trend towards moving computation, including the training of machine-learning models, from cloud/datacenters to edge devices [1,24]. Ideally, since trained on massive representative data, the resulting models exhibit improved generalization. In this paper, we consider distributed on-device machine learning. The distributed system is a server-worker architecture. The workers are placed on edge devices, which train the models on the private data. The servers are placed on the cloud/datacenters which maintain a shared global model. Distributed settings require addressing some novel engineering challenges, including the following:

– **Limited, heterogeneous computation.** Edge devices, including smart phones, wearable devices, sensors, or vehicles typically have weaker computational ability, compared to the workstations or datacenters used in typical distributed machine learning. Thus, simpler models and stochastic training are usually applied in practice. Furthermore, different devices have different computation capabilities.
– **Limited communication** The connection to the central servers are not guaranteed. Communication can be frequently unavailable, slow, or expensive (in money or in the power of battery). Thus, frequent high-speed communication is typically unaffordable.
– **Decentralized, non-IID training data.** Privacy needs and legal requirements (e.g., US HIPAA laws [12] in a smart hospital, or Europe's GDPR law [8]) may necessitate that training be performed on-premises using IoT

devices and edge machines, and that data and models must not be deposited in the cloud or cloudlets. In more general cases, the users simply dislike sharing their on-device data which potentially reveals private information. As a result, the data distribution on different devices are neither mixed nor IID i.e. unlike standard settings, device data are non-identically distributed samples from the population. This is particularly true when each device is controlled by a specific user whose behavior is supposed to be unique. Furthermore, the sampled data on nearby devices are potentially non-independent, since such devices can be shared by the same user or family. For example, the data of a step counter from a wearable fitness tracker and a smart phone owned by the same user can have different distributions of motion data with mutual dependency. Imagine that the fitness tracker is only used when the user is running, and the smart phone is only used when the user is walking, which results in different distributions. On the other hand, the complementation yields dependency.
- **Untrusted workers and data poisoning.** The servers have limited control over the users' behavior. To protect the privacy, the users are in general anonymous to the servers. Although it is possible to verity the identity of workers/devices [25], nefarious users can feed poisoned data with abnormal behaviors without backdooring OS. As a result, some workers may push models learned on poisoned data to the servers.

To overcome the challenges above, we introduce Secure Local Stochastic Gradient Descent (SLSGD), which reduces the communication overhead with local updates, and secures the global model against nefarious users and poisoned data. We summarize the key properties of SLSGD below:

- **Local SGD.** SGD is widely used for training models with lower computation overhead. To reduce communication overhead, we use SGD with local updates. The workers do not synchronize with the server after each local gradient descent step. After several local iterations, the workers push the updated model to the servers, which is different from the traditional distributed synchronous SGD where gradients are pushed in each local gradient descent step. To further reduce the communication overhead, the training tasks are activated on a random subset of workers in each global epoch.
- **Secure aggregation.** In each global epoch, the servers send the latest global model to the activated workers, and aggregate the updated local models. In such procedure, there are two types of threats: i) poisoned models pushed from comprised devices, occupied or hacked by nefarious users; ii) accumulative error, variance, or models over-fitted on the local dataset, caused by infrequent synchronization of local SGD. To secure the global model against these two threats, we use robust aggregation which tolerates abnormal models, and moving average which mitigates the errors caused by infrequent synchronization.

To our knowledge, there is limited work on local SGD with theoretical guarantees [31,26]. The existing convergence guarantees are based on the strong assumption of IID training data or homogeneous local iterations, which we have argued is inappropriate for distributed learning on edge devices.

We propose SLSGD, which is a variant of local SGD with provable convergence under non-IID and heterogeneous settings, and tolerance to nefarious users. In summary, the main contributions are listed as follows:

– We show that SLSGD theoretically converges to global optimums for strongly convex functions, non-strongly convex functions, and a restricted family of non-convex functions, under non-IID settings. Furthermore, more local iterations accelerate the convergence.
– We show that SLSGD tolerates a small number of workers training on poisoned data. As far as we know, this paper is the first to investigate the robustness of local SGD.
– We show empirically that the proposed algorithm stabilizes the convergence, and protects the global model from data poisoning.

2 Related Work

Our algorithm is based on local SGD introduced in [31,26]. The major differences are:

1. We assume non-IID training data and heterogeneous local iterations among the workers. In previous work, local SGD and its convergence analysis required IID training data, or same number of local iterations within each global epoch (or both). However, these assumptions are unreasonable for edge computing, due to privacy preservation and heterogeneous computation.
2. Instead of using the averaged model to overwrite the current global model on the server, we take robust aggregation, and use a moving average to update the current model. These techniques not only secure the global model against data poisoning, but also mitigate the error caused by infrequent synchronization of local SGD.

The limited communication power of edge devices also motivates federated learning [16,17,22], whose algorithm is similar to local SGD, and scenario is similar to our non-IID and heterogeneous settings. Unfortuntaely, federated learning lacks provable convergence guarantees. Furthermore, the issues of data poisoning have not been addressed in previous work. To the best of our knowledge, our proposed work is the first that considers both convergence and robustness, theoretically and practically, on non-IID training data.

Similar to the traditional distributed machine learning, we use the server-worker architecture, which is similar to the Parameter Server (PS) architecture. Stochastic Gradient Descent (SGD) with PS architecture, is widely used in typical distributed machine learning [19,13,20]. Compared to the traditional distributed learning on PS , SLSGD has much less synchronization. Furthermore, in SLSGD, the workers push trained models instead of gradients to the servers.

Approaches based on robust statistics are often used to address security issues in the PS architecture [30,27]. This enables procedures which tolerate multiple types of attacks and system failures. However, the existing methods and theoretical analysis do not consider local training on non-IID data. So far, the convergence guarantees are based on robust gradient aggregation. In this paper, we provide convergence guarantees for robust model aggregation. Note that gradients and models (parameters) have different properties. For example, the gradients converge to 0 for unconstrained problems, while the models do not have such property. On the other hand, recent work has considered attacks not only targeting traditional distributed SGD [28], but also federated learning [3,9,4], but do not propose defense techniques with provable convergence. There are other robust SGD algorithms whose defense techniques are not based on robust stochastic [29].

There is growing literature on the practical applications of edge and fog computing [10,14] in various scenarios such as smart home or sensor networks. More and more big-data applications are moving from the cloud to the edge, including for machine-learning tasks [5,21,32]. Although computational power is growing, edge devices are still much weaker than the workstations and datacenters used in typical distributed machine learning e.g. due to the limited computation and communication capacity, and limited power of batteries. To this end, there are machine-learning frameworks with simple architectures such as MobileNet [15] which are designed for learning with weak devices.

3 Problem Formulation

Consider distributed learning with n devices. On each device, there is a worker process that trains the model on local data. The overall goal is to train a global model $x \in \mathbb{R}^d$ using data from all the devices.

To do so, we consider the following optimization problem: $\min_{x \in \mathbb{R}^d} F(x)$, where $F(x) = \frac{1}{n} \sum_{i \in [n]} \mathbb{E}_{z^i \sim \mathcal{D}^i} f(x; z^i)$, for $\forall i \in [n]$, z^i is sampled from the local data $\mathcal{D}^i$ on the ith device.

3.1 Non-IID Local Datasets

Note that different devices have different local datasets, i.e., $\mathcal{D}^i \neq \mathcal{D}^j, \forall i \neq j$. Thus, samples drawn from different devices have different expectations, i.e., $\mathbb{E}_{z^i \sim \mathcal{D}^i} f(x; z^i) \neq \mathbb{E}_{z^j \sim \mathcal{D}^j} f(x; z^j), \forall i \neq j$. Further, since different devices can be possessed by the same user or the same group of users (e.g., families), samples drawn from different devices can be potentially dependent on each other.

3.2 Data Poisoning

The users are anonymous to the servers. Furthermore, it is impossible for the servers to verify the benignity of the on-device training data. Thus, the servers can not trust the edge devices. A small number of devices may be susceptible to data poisoned by abnormal user behaviors or in the worst case, are controlled by users or agents who intend to directly upload harmful models to the servers.

Table 1. Notations and Terminologies

Notation/Term	Description
n	Number of devices
k	Number of simutaneously updating devices
T	Number of communication epochs
$[n]$	Set of integers $\{1, \ldots, n\}$
S_t	Randomly selected devices in the t^{th} epoch
b	Parameter of trimmed mean
H_{min}	Minimal number of local iterations
H_t^i	Number of local iterations in the t^{th} epoch on the ith device
x_t	Initial model in the t^{th} epoch
$x_{t,h}^i$	Model updated in the t^{th} epoch, hth local iteration, on the ith device
$\mathcal{D}^i$	Dataset on the ith device
$z_{t,h}^i$	Data (minibatch) sampled in the t^{th} epoch, hth local iteration, on the ith device
γ	Learning rate
α	Weight of moving average
$\|\cdot\|$	All the norms in this paper are l_2-norms
Device	Where the training data are placed
Worker	One worker on each device, process that trains the model
User	Agent that produces data on the devices, and/or controls the devices
Nefarious user	Special user that produces poisoned data or has abnormal behaviors

In this paper, we consider a generalized threat model, where the workers can push arbitrarily bad models to the servers. The bad models can cause divergence of training. Beyond more benign issues such as hardware, software or communication failures, there are multiple ways for nefarious users to manipulate the uploaded models e.g. data poisoning [2]. In worst case, nefarious users can even directly hack the devices and replace the correct models with arbitrary values. We provide a more formal definition of the threat model in Section 4.1.

4 Methodology

In this paper, we propose SLSGD: SGD with communication efficient local updates and secure model aggregation. A single execution of SLSGD is composed of T communication epochs. At the beginning of each epoch, a randomly selected group of devices S_t pull the latest global model from the central server. Then, the same group of devices locally update the model without communication with the central server. At the end of each epoch, the central server aggregates the updated models and then updates the global model.

In the t^{th} epoch, on the ith device, we locally solve the following optimization problem using SGD for H_t^i iterations: $\min_{x \in \mathbb{R}^d} \mathbb{E}_{z^i \sim \mathcal{D}^i} f(x; z^i)$. Then, the server collects the resulting local models x^i_{t,H^i_t}, and aggregates them using $\texttt{Aggr}\left(\{x^i_{t,H^i_t} : i \in S_t\}\right)$. Finally, we update the model with a moving average over the current model and the aggregated local models.

The detailed algorithm is shown in Algorithm 1. $x^i_{t,h}$ is the model parameter updated in hth local iteration of the t^{th} epoch, on the ith device. $z^i_{t,h}$ is the data randomly drawn in hth local iteration of the t^{th} epoch, on the ith device. H^i_t is the number of local iterations in the t^{th} epoch, on the ith device. γ is the learning rate and T is the total number of epochs. Note that if we take Option I (or Option II with $b = 0$) with $\alpha = 1$, the algorithm is the same as the federated learning algorithm *FedAvg* [22]. Furthermore, if we take homogeneous local iterations $H^i_t = H, \forall i$, Option I with $\alpha = 1$ is the same as local SGD [26]. Thus, FedAvg and local SGD are both special cases of SLSGD.

Algorithm 1 SLSGD

```
1: Input: $k \in [n]$, $b$
2: Initialize $x_0$
3: for all epoch $t \in [T]$ do
4:    Randomly select a group of $k$ workers, denoted as $S_t \subseteq [n]$
5:    for all $i \in S_t$ in parallel do
6:       Receive the latest global model $x_{t-1}$ from the server
7:       $x^i_{t,0} \leftarrow x_{t-1}$
8:       for all local iteration $h \in [H^i_t]$ do
9:          Randomly sample $z^i_{t,h}$
10:         $x^i_{t,h} \leftarrow x^i_{t,h-1} - \gamma \nabla f(x^i_{t,h-1}; z^i_{t,h})$
11:      end for
12:      Push $x^i_{t,H^i_t}$ to the server
13:   end for
14:   Aggregate: $x'_t \leftarrow \begin{cases} \text{Option I:} & \frac{1}{k}\sum_{i \in S_t} x^i_{t,H^i_t} \\ \text{Option II:} & \texttt{Trmean}_b\left(\left\{x^i_{t,H^i_t} : i \in S_t\right\}\right) \end{cases}$
15:   Update the global model: $x_t \leftarrow (1-\alpha)x_{t-1} + \alpha x'_t$
16: end for
```

4.1 Threat Model and Defense Technique

First, we formally define the threat model.

Definition 1. *(Threat Model) In Line 12 of Algorithm 1, instead of the correct x^i_{t,H^i_t}, a worker, training on poisoned data or controlled by an abnormal/nefarious user, may push arbitrary values to the server.*

Remark 1. Note that the users/workers are anonymous to the servers, and the nefarious users can sometimes pretend to be well-behaved to fool the servers. Hence, it is impossible to surely identify the workers training on poisoned data, according to their historical behavior.

In Algorithm 1, Option II uses the trimmed mean as a robust aggregation which tolerates the proposed threat model. To define the trimmed mean, we first define the order statistics.

Definition 2. *(Order Statistics) By sorting the scalar sequence $\{u_i : i \in [k], u_i \in \mathbb{R}\}$, we get $u_{1:k} \leq u_{2:k} \leq \ldots \leq u_{k:k}$, where $u_{i:k}$ is the ith smallest element in $\{u_i : i \in [k]\}$.*

Then, we define the trimmed mean.

Definition 3. *(Trimmed Mean) For $b \in \{0, 1, \ldots, \lceil k/2 \rceil - 1\}$, the b-trimmed mean of the set of scalars $\{u_i : i \in [k]\}$ is defined as follows:*

$$\texttt{Trmean}_b(\{u_i : i \in [k]\}) = \frac{1}{k-2b} \sum_{i=b+1}^{k-b} u_{i:k},$$

where $u_{i:k}$ is the ith smallest element in $\{u_i : i \in [k]\}$ defined in Definition 2. The high-dimensional version ($u_i \in \mathbb{R}^d$) of $\texttt{Trmean}_b(\cdot)$ simply applies the trimmed mean in a coordinate-wise manner.

Note that the trimmed mean (Option II) is equivalent to the standard mean (Option I) if we take $b = 0$.

Remark 2. Algorithm 1 provides two levels of defense: robust aggregation (Line 14) and moving average (Line 15). The robust aggregation tries to filter out the models trained on poisoned data. The moving average mitigates not only the extra variance/error caused by robust aggregation and data poisoning, but also the accumulative error caused by infrequent synchronization of local updates.

Remark 3. We can also replace the coordinate-wise trimmed mean with other robust statistics such as geometric median [7]. We choose coordinate-wise median/trimmed mean in this paper because unlike geometric median, trimmed mean has a computationally efficient closed-form solution.

5 Convergence Analysis

In this section, we prove the convergence of Algorithm 1 with non-IID data, for a restricted family of non-convex functions. Furthermore, we show that the proposed algorithm tolerates the threat model introduced in Definition 1. We start with the assumptions required by the convergence guarantees.

5.1 Assumptions

For convenience, we denote $F^i(x) = \mathbb{E}_{z^i \sim \mathcal{D}^i} f(x; z^i)$.

Assumption 1. *(Existence of Global Optimum) We assume that there exists at least one (potentially non-unique) global minimum of the loss function $F(x)$, denoted by x^*.*

Assumption 2. *(Bounded Taylor's Approximation) We assume that for $\forall x, z$, $f(x; z)$ has L-smoothness and μ-weak convexity: $\langle \nabla f(x; z), y - x \rangle + \frac{\mu}{2}\|y - x\|^2 \leq f(y; z) - f(x; z) \leq \langle \nabla f(x; z), y - x \rangle + \frac{L}{2}\|y - x\|^2$, where $\mu \leq L$, and $L > 0$.*

Note that Assumption 2 covers the case of non-convexity by taking $\mu < 0$, non-strong convexity by taking $\mu = 0$, and strong convexity by taking $\mu > 0$.

Assumption 3. *(Bounded Gradient) We assume that for $\forall x \in \mathbb{R}^d, i \in [n]$, and $\forall z \sim \mathcal{D}^i$, we have $\|\nabla f(x; z)\|^2 \leq V_1$.*

Based on the assumptions above, we have the following convergence guarantees. All the detailed proofs can be found in the appendix.

5.2 Convergence without Data Poisoning

First, we analyze the convergence of Algorithm 1 with Option I, where there are no poisoned workers.

Theorem 1. *We take $\gamma \leq \min\left(\frac{1}{L}, 2\right)$. After T epochs, Algorithm 1 with Option I converges to a global optimum:*

$$\mathbb{E}\left[F(x_T) - F(x_*)\right] \leq \left(1 - \alpha + \alpha(1 - \frac{\gamma}{2})^{H_{min}}\right)^T \left[F(x_0) - F(x_*)\right] + \left[1 - \left(1 - \alpha + \alpha(1 - \frac{\gamma}{2})^{H_{min}}\right)^T\right] \mathcal{O}\left(V_1 + \left(1 + \frac{1}{k} - \frac{1}{n}\right) V_2\right),$$

where $V_2 = \max_{t \in \{0, T-1\}, h \in \{0, H_t^i - 1\}, i \in [n]} \|x_{t,h}^i - x_*\|^2$.

Remark 4. When $\alpha \to 1$, $\left(1 - \alpha + \alpha(1 - \frac{\gamma}{2})^{H_{min}}\right)^T \to (1 - \frac{\gamma}{2})^{TH_{min}}$, which results in nearly linear convergence to the global optimum, with error $\mathcal{O}(V_1 + V_2)$. When $\alpha \to 0$, the error is nearly reduced 0, but the convergence will slow down. We can tune α to trade-off between the convergence rate and the error. In practice, we can take diminishing α: $\alpha_t \propto \frac{1}{t^2}$, where α_t is the α in the t^{th} global epoch. Furthermore, taking $\alpha_T = \frac{1}{T^2}$, $\lim_{T \to +\infty} \left[1 - \left(1 - \alpha_T + \alpha_T(1 - \frac{\gamma}{2})^{H_{min}}\right)^T\right] = 0$.

5.3 Convergence with Data Poisoning

Under the threat model defined in Definition 1, in worst case, Algorithm 1 with Option I and $\alpha = 1$ (local SGD) suffers from unbounded error.

Proposition 1. *(Informal) Algorithm 1 with Option I and $\alpha = 1$ can not tolerate the threat model defined in Definition 1.*

Proof. (Sketch) Without loss of generality, assume that in a specific epoch t, among all the k workers, the last q_1 of them are poisoned. For the poisoned workers, instead of pushing the correct value x^i_{t,H^i_t} to the server, they push $-\frac{k-q_1}{q_1} x^i_{t,H^i_t} + c$, where c is an arbitrary constant. For convenience, we assume IID (required by local SGD, but not our algorithm) local datasets for all the workers. Thus, the expectation of the aggregated global model becomes $\frac{1}{k}\left\{(k-q_1)\mathbb{E}\left[x^i_{t,H^i_t}\right] + q_1\mathbb{E}\left[-\frac{k-q_1}{q_1} x^i_{t,H^i_t} + c\right]\right\} = \frac{q_1}{k}c$, which means that in expectation, the aggregated global model can be manipulated to take arbitrary values, which results in unbounded error.

In the following theorems, we show that using Algorithm 1 with Option II, the error can be upper bounded.

Theorem 2. *Assume that additional to the n normal workers, there are q workers training on poisoned data, where $q \ll n$, and $2q \le 2b < k$. We take $\gamma \le \min\left(\frac{1}{L}, 2\right)$. After T epochs, Algorithm 1 with Option II converges to a global optimum:*

$$\begin{aligned}\mathbb{E}\left[F(x_T) - F(x_*)\right] &\le \left(1-\alpha+\alpha(1-\frac{\gamma}{2})^{H_{min}}\right)^T \left[F(x_0)-F(x_*)\right] \\ &+ \left[1-\left(1-\alpha+\alpha(1-\frac{\gamma}{2})^{H_{min}}\right)^T\right]\left[\mathcal{O}(V_1)+\mathcal{O}(\beta V_2)\right],\end{aligned}$$

where $V_2 = \max_{t\in\{0,T-1\},h\in\{0,H^i_t-1\},i\in[n]} \|x^i_{t,h} - x_*\|^2$, $\beta = 1 + \frac{1}{k-q} - \frac{1}{n} + \frac{k(k+b)}{(k-b-q)^2}$.

Remark 5. Note that the additional error caused by the q poisoned workers and b-trimmed mean is controlled by the factor $\frac{k(k+b)}{(k-b-q)^2}$, which decreases when q and b decreases, or k increases.

6 Experiments

In this section, we evaluate the proposed algorithm by testing its convergence and robustness. Note that zoomed figures of the empirical results can be found in the appendix.

6.1 Datasets and Evaluation Metrics

We conduct experiments on the benchmark CIFAR-10 image classification dataset [18], which is composed of 50k images for training and 10k images for testing. Each image is resized and cropped to the shape of $(24, 24, 3)$. We use a convolutional neural network (CNN) with 4 convolutional layers followed by 1 fully connected layer. We use a simple network architecture, so that it can be easily handled by edge devices. The detailed network architecture can be found in our submitted source code (will also be released upon publication). The experiments are conducted on CPU devices. We implement SLSGD using the MXNET [6] framework.

We also conduct experiments of LSTM-based language models on WikiText-2 dataset [23]. The model architecture was taken from the MXNET and Gluon-NLP tutorial [11]. The results can be found in the appendix.

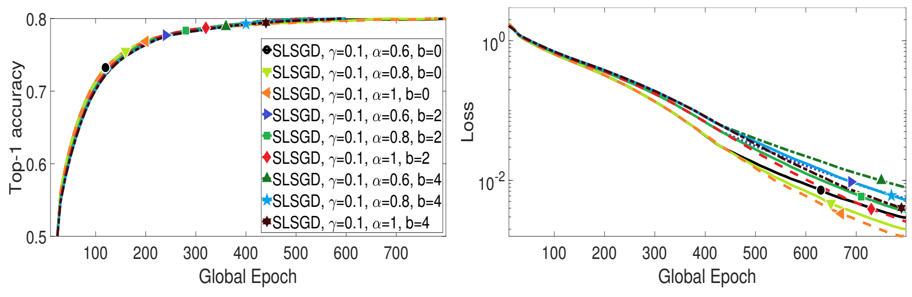
In each experiment, the training set is partitioned onto $n = 100$ devices. We test the preformance of SLSGD on both balanced and unbalanced partitions:

– **Balanced Partition.** Each of the $n = 100$ partitions has 500 images.
– **Unbalanced Partition.** To make the setting more realistic, we partition the training set into unbalanced sizes. The sizes of the 100 partitions are $104, 112, \ldots, 896$ (an arithmetic sequence with step 8, starting with 104). Furthermore, to enlarge the variance, we make sure that in each partition, there are at most 5 different labels out of all the 10 labels. Note that some partitions only have one label.

In each epoch, $k = 10$ devices are randomly selected to launch local updates, with the minibatch size of 50. We repeat each experiment 10 times and take the average. We use top-1 accuracy on the testing set, and cross entropy loss function on the training set as the evaluation metrics.

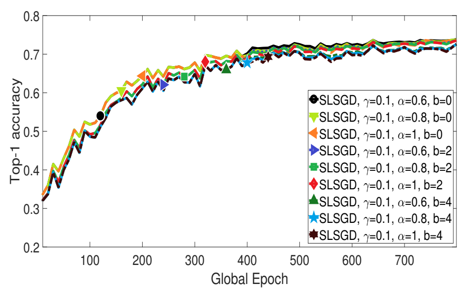
The baseline algorithm is *FedAvg* introduced by [22], which is a special case of our proposed Algorithm 1 with Option I and $\alpha = 1$. To make the comparison clearer, we refer to *FedAvg* as "SLSGD, $\alpha = 1, b = 0$".

We test SLSGD with different hyperparameters γ, α, and b (definitions can be found in Table 1).

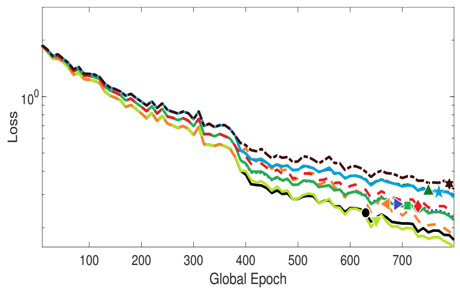


(a) Top-1 accuracy on testing set (b) Cross entropy on training set

Fig. 1. Convergence on training data with balanced partition, without attack. Each epoch is a full pass of the local training data. Legend "SLSGD, $\gamma = 0.1, \alpha = 0.8, b = 2$" means that SLSGD takes the learning rate 0.1 and $\texttt{Trmean}_2$ for aggregation, and the initial $\alpha = 1$ decays by the factor of 0.8 at the 400th epoch. SLSGD with $\alpha = 1$ and $b = 0$ is the baseline *FedAvg*. Note that we fix the random seeds. Thus, before α decays at the 400th epoch, results with the same γ and b are the same.



(a) Top-1 accuracy on testing set



(b) Cross entropy on training set

Fig. 2. Convergence on training data with unbalanced partition, without attack. Each epoch is a full pass of the local training data. Legend "SLSGD, $\gamma = 0.1, \alpha = 0.8, b = 2$" means that SLSGD takes the learning rate 0.1 and $\texttt{Trmean}_2$ for aggregation, and the initial $\alpha = 1$ decays by the factor of 0.8 at the 400th epoch. SLSGD with $\alpha = 1$ and $b = 0$ is the baseline *FedAvg*. Note that we fix the random seeds. Thus, before α decays at the 400th epoch, results with the same γ and b are the same.

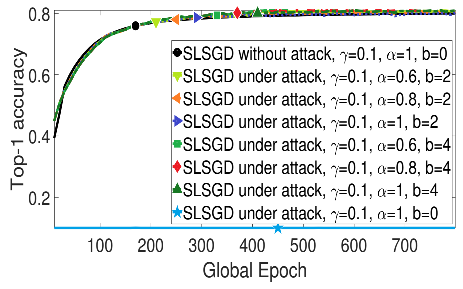
6.2 SLSGD without Attack

We first test the performance of SLSGD on the training data with balanced partition, without data poisoning. The result is shown in Fig. 1. When there are no poisoned workers, using trimmed mean results in extra variance. Although larger b and smaller α makes the convergence slower, the gap is tiny. In general, SLSGD is insensitive to hyperparameters.

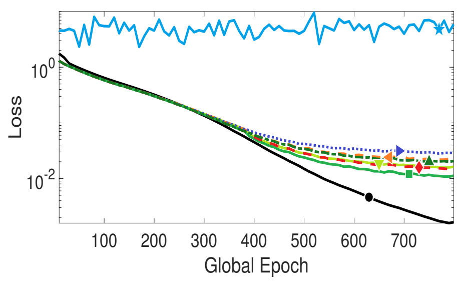
Then, we test the performance with unbalanced partition, without data poisoning. The result is shown in Fig. 2. Note that the convergence with unbalanced partition is generally slower compared to balanced partition due to the larger variance. Using appropriate α ($\alpha = 0.8$) can mitigate such extra variance.

6.3 SLSGD under Data Poisoning Attack

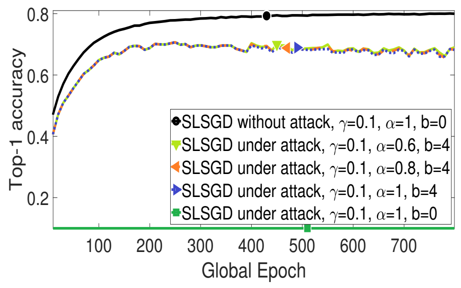
To test the tolerance to poisoned workers, we simulate data poisoning which "flips" the labels of the local training data. The poisoned data have "flipped" labels, i.e., each $label \in \{0, \ldots, 9\}$ in the local training data will be replaced by $(9 - label)$. The experiment is set up so that in each epoch, in all the $k = 10$ randomly selected workers, q workers are compromised and subjected to data poisoning. The results are shown in Fig. 3 and Fig. 4. We use FedAvg/SLSGD without data poisoning (Option I) as the ideal benchmark. As expected, SLSGD without trimmed mean can not tolerate data poisoning, which causes catastrophic failure. SLSGD with Option II tolerates the poisoned worker, though converges slower compared to SLSGD without data poisoning. Furthermore, larger b and smaller α improves the robustness and stabilizes the convergence.



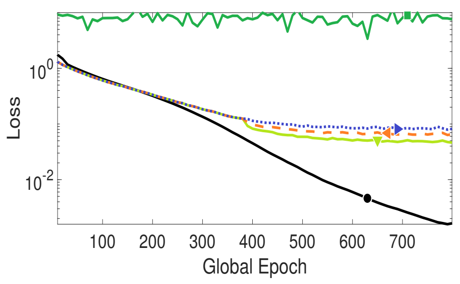
(a) Top-1 accuracy on testing set, $q = 2$



(b) Cross entropy on training set, $q = 2$



(c) Top-1 accuracy on testing set, $q = 4$



(d) Cross entropy on training set, $q = 4$

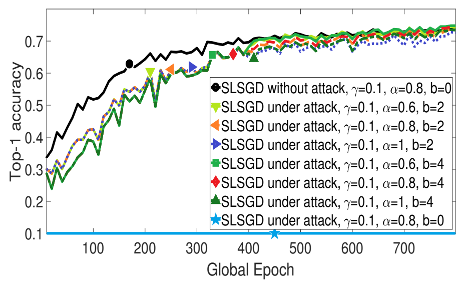
Fig. 3. Convergence on training data with balanced partition, with "label-flipping" attack. In each epoch, we guarantee that $q \in \{2, 4\}$ of the $k = 10$ selected workers are poisoned. Each epoch is a full pass of the local training data. Legend "SLSGD, $\gamma = 0.1, \alpha = 0.8, b = 2$" means that SLSGD takes the learning rate 0.1 and $\texttt{Trmean}_2$ for aggregation, and the initial $\alpha = 1$ decays by the factor of 0.8 at the 400th epoch. SLSGD with $\alpha = 1$ and $b = 0$ is the baseline *FedAvg*. Note that we fix the random seeds. Thus, before α decays at the 400th epoch, results with the same γ and b are the same.

Note that taking $q = 4$ in every epoch pushes to the limit of SLSGD since the algorithm requires $2q < k$. In practice, if there are totally $q = 4$ poisoned workers in the entire $n = 100$ workers, there is no guarantee that the poisoned workers will always be activated in each epoch. Poisoning 40% of the sampled data in each epoch incurs huge noise, while SLSGD can still prevent the global model from divergence.

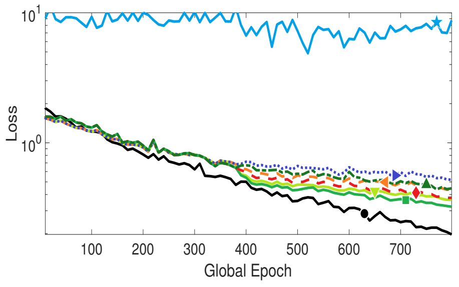
In Fig. 5, we show how α and b affect the convergence when data poisoning and unbalanced partition cause extra error and variance. In such scenario, larger b and smaller α makes SLSGD more robust and converge faster.

6.4 Acceleration by Local Updates

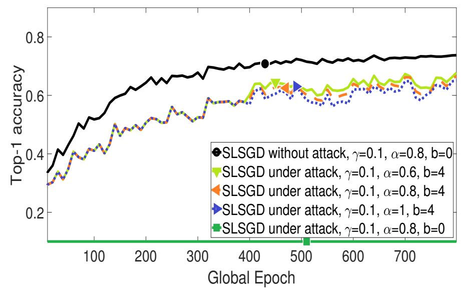
According to our theoretical analysis, more local updates in each epoch accelerate the convergence. We test this theory in Fig. 6 with balanced partition, without data poisoning. In the legend, "pass=3" means each epoch is 3 full passes of the



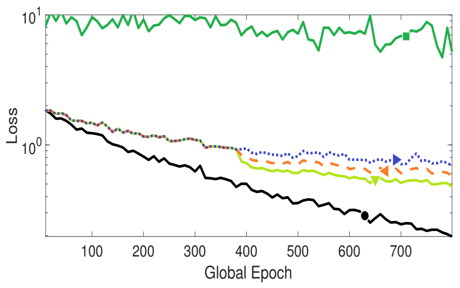
(a) Top-1 accuracy on testing set, $q = 2$



(b) Cross entropy on training set, $q = 2$



(c) Top-1 accuracy on testing set, $q = 4$



(d) Cross entropy on training set, $q = 4$

Fig. 4. Convergence on training data with unbalanced partition, with "label-flipping" attack. In each epoch, we guarantee that q of the $k = 10$ selected workers are poisoned. Each epoch is a full pass of the local training data. Legend "SLSGD, $\gamma = 0.1, \alpha = 0.8, b = 2$" means that SLSGD takes the learning rate 0.1 and $\texttt{Trmean}_2$ for aggregation, and the initial $\alpha = 1$ decays by the factor of 0.8 at the 400th epoch. SLSGD with $\alpha = 1$ and $b = 0$ is the baseline *FedAvg*. Note that we fix the random seeds. Thus, before α decays at the 400th epoch, results with the same γ and b are the same.

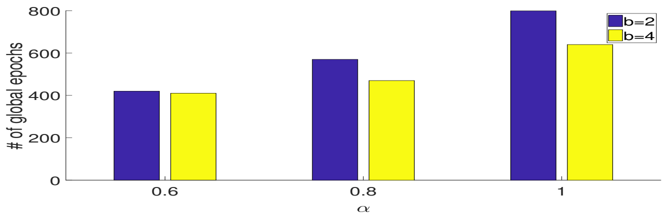


Fig. 5. Number of global epochs to reach training loss value 0.5, with unbalanced partition and $q = 2$ poisoned workers. $\gamma = 0.1$. α and b varies. "α" on the x-axis is the initial value of α, which does not decay during training.

local datasets ($H = 3 \times 500/50 = 30$ local iterations) on the selected workers. We show that with more local iterations, SLSGD converges faster.

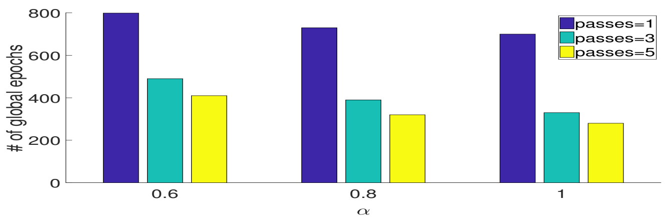


Fig. 6. Number of global epochs to reach training loss value 0.003, with balanced partition, without poisoned workers. $\gamma = 0.1$. α and number of local iterations varies. "pass=3" means each epoch is 3 full passes of the local datasets on the selected workers. "α" on the x-axis is the initial value of α, which does not decay during training.

6.5 Discussion

The hyperparameters of SLSGD affects the convergence differently in different scenarios:

– *Balanced partition, no attacks.* In this case, the overall variance is relatively small. Thus, it is not necessary to use smaller α to mitigate the variance. The extra variance caused by trimmed mean slows down the convergence. Since the variance does not dominate, smaller α and larger b potentially slow down the convergence, but the gap is tiny.
– *Unbalanced partition, no attacks.* In this case, the overall variance is larger than the balanced case. Note that not only the size of local datasets, but also the label distribution are unbalanced among the devices. Some partitions only contains one label, which enlarges the accumulative error caused by infrequent synchronization and overfitting the local training data. Thus, using appropriate α can mitigate the variance. However, it is not necessary to use the trimmed mean, since the variance caused by unbalanced partition is not too bad compared to data poisoning.
– *Balanced partition, under attacks.* In this case, the error caused by poisoned workers dominates. We must use trimmed mean to prevent divergence. Larger b improves the robustness and convergence. Furthermore, using smaller α also mitigates the error and improves the convergence.
– *Unbalanced partition, under attacks.* In this case, the error caused by poisoned workers still dominates. In general, the usage of hyperparameters is similar to the case of balanced partition under attacks. However, the unbalanced partition makes it more difficult to distinguish poisoned workers from normal workers. As a result, the convergence gets much slower. Smaller α obtain more improvement and better stabilization.

In general, there is a trade-off between convergence rate and variance/error reduction. In the ideal case, if the variance is very small, SLSGD with $\alpha = 1$ and $b = 0$, i.e., *FedAvg*, has fastest convergence. Using other hyperparameters slightly slows down the convergence, but the gap is tiny. When variance gets larger, users can try smaller α. When the variance/error gets catastrophically large, the users can use the trimmed mean to prevent divergence.

7 Conclusion

We propose a novel distributed optimization algorithm on non-IID training data, which has limited communication and tolerates poisoned workers. The algorithm has provable convergence. Our empirical results show good performance in practice. In future work, we are going to analyze our algorithm on other threat models, such as hardware or software failures.

Acknowledgements

This work was funded in part by NSF CNS 1409416, by a gift from Microsoft, and by computational resources donated by Intel, AWS, and Microsoft Azure.

References

1. Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J.L.: A public domain dataset for human activity recognition using smartphones. In: ESANN (2013)
2. Bae, H., Jang, J., Jung, D., Jang, H., Ha, H., Yoon, S.: Security and privacy issues in deep learning. arXiv preprint arXiv:1807.11655 (2018)
3. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V.: How to backdoor federated learning. arXiv preprint arXiv:1807.00459 (2018)
4. Bhagoji, A.N., Chakraborty, S., Mittal, P., Calo, S.: Analyzing federated learning through an adversarial lens. arXiv preprint arXiv:1811.12470 (2018)
5. Cao, Y., Hou, P., Brown, D., Wang, J., Chen, S.: Distributed analytics and edge intelligence: Pervasive health monitoring at the era of fog computing. In: Proceedings of the 2015 Workshop on Mobile Big Data. pp. 43–48. ACM (2015)
6. Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., Zhang, Z.: MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv preprint arXiv:1512.01274 (2015)
7. Chen, Y., Su, L., Xu, J.: Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. ACM SIGMETRICS Performance Evaluation Review **46**(1), 96–96 (2019)
8. EU: European Union's General Data Protection Regulation (GDPR) (2018), `https://eugdpr.org/`, Last visited: Nov. 2018
9. Fung, C., Yoon, C.J., Beschastnikh, I.: Mitigating Sybils in federated learning poisoning. arXiv preprint arXiv:1808.04866 (2018)
10. Garcia Lopez, P., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M., Felber, P., Riviere, E.: Edge-centric computing: Vision and challenges. ACM SIGCOMM Computer Communication Review **45**(5), 37–42 (2015)
11. gluon-nlp.mxnet.io: LSTM-based Language Models (2019), `https://gluon-nlp.mxnet.io/master/examples/language_model/language_model.html`, Last visited: Mar. 2019
12. HealthInsurance.org, S.A.: Health insurance portability and accountability act of 1996. Public law **104**, 191 (1996)
13. Ho, Q., Cipar, J., Cui, H., Lee, S., Kim, J.K., Gibbons, P.B., Gibson, G.A., Ganger, G., Xing, E.P.: More effective distributed ml via a stale synchronous parallel parameter server. In: Advances in neural information processing systems. pp. 1223–1231 (2013)

14. Hong, K., Lillethun, D., Ramachandran, U., Ottenwälder, B., Koldehofe, B.: Mobile fog: A programming model for large-scale applications on the internet of things. In: Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing. pp. 15–20. ACM (2013)
15. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
16. Konečnỳ, J., McMahan, B., Ramage, D.: Federated optimization: Distributed optimization beyond the datacenter. arXiv preprint arXiv:1511.03575 (2015)
17. Konečnỳ, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016)
18. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
19. Li, M., Andersen, D.G., Park, J.W., Smola, A.J., Ahmed, A., Josifovski, V., Long, J., Shekita, E.J., Su, B.Y.: Scaling distributed machine learning with the parameter server. In: OSDI. vol. 14, pp. 583–598 (2014)
20. Li, M., Andersen, D.G., Smola, A.J., Yu, K.: Communication efficient distributed machine learning with the parameter server. In: Advances in Neural Information Processing Systems. pp. 19–27 (2014)
21. Mahdavinejad, M.S., Rezvan, M., Barekatain, M., Adibi, P., Barnaghi, P., Sheth, A.P.: Machine learning for internet of things data analysis: A survey. Digital Communications and Networks **4**(3), 161–175 (2018)
22. McMahan, H.B., Moore, E., Ramage, D., Hampson, S., et al.: Communication-efficient learning of deep networks from decentralized data. arXiv preprint arXiv:1602.05629 (2016)
23. Merity, S., Xiong, C., Bradbury, J., Socher, R.: Pointer sentinel mixture models. arXiv preprint arXiv:1609.07843 (2016)
24. Pantelopoulos, A., Bourbakis, N.G.: A survey on wearable sensor-based systems for health monitoring and prognosis. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **40**(1), 1–12 (2010)
25. source.android.com: Key and ID Attestation (2019), `https://source.android.com/security/keystore/attestation`, Last visited: Mar. 2019
26. Stich, S.U.: Local SGD Converges Fast and Communicates Little. arXiv preprint arXiv:1805.09767 (2018)
27. Xie, C., Koyejo, O., Gupta, I.: Phocas: dimensional Byzantine-resilient stochastic gradient descent. arXiv preprint arXiv:1805.09682 (2018)
28. Xie, C., Koyejo, S., Gupta, I.: Fall of Empires: Breaking Byzantine-tolerant SGD by Inner Product Manipulation. In: Proceedings of the 35th conference on Uncertainty in artificial intelligence. AUAI Press (2019)
29. Xie, C., Koyejo, S., Gupta, I.: Zeno: Distributed Stochastic Gradient Descent with Suspicion-based Fault-tolerance. In: International Conference on Machine Learning. pp. 6893–6901 (2019)
30. Yin, D., Chen, Y., Ramchandran, K., Bartlett, P.: Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. arXiv preprint arXiv:1803.01498 (2018)
31. Yu, H., Yang, S., Zhu, S.: Parallel restarted SGD for non-convex optimization with faster convergence and less communication. arXiv preprint arXiv:1807.06629 (2018)
32. Zeydan, E., Bastug, E., Bennis, M., Kader, M.A., Karatepe, I.A., Er, A.S., Debbah, M.: Big data caching for networking: Moving from cloud to edge. IEEE Communications Magazine **54**(9), 36–42 (2016)