Paper ID: 8424

Byzantine-Robust Decentralized Federated Learning via Local Performance Checking

Kaichuang Zhang[1], Alina Basharat[1] and Ping Xu[1]

[1]The University of Texas Rio Grande Valley

1.Introduction

- Performance evaluation has been proven successful in Federated Learning (FL), but it utilizes a performance test dataset at the central server which is unreasonable.
- The client in **Decentralized Federated Learning (DFL)** has its own local dataset that can be used as a performance test dataset; performance evaluation has not been proven in DFL. Here is an example, as Figure 1 shows.

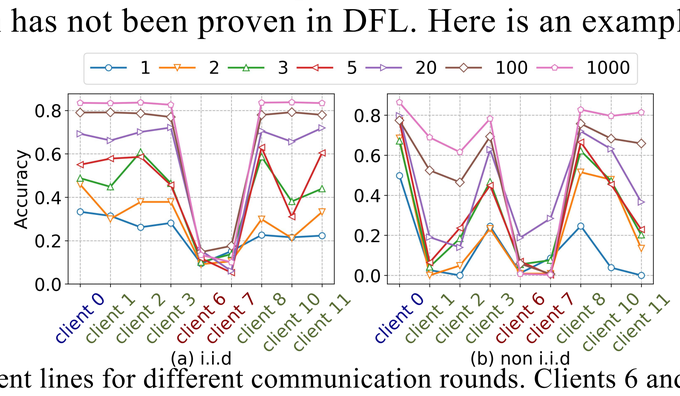


Figure 1. Different lines for different communication rounds. Clients 6 and 7 are Byzantine clients, other clients are benign.

2.Problem Statement

The learning task of DFL is to minimize the average of local clients' cost under a fixed undirected graph, given by

$$\min_{x\in\mathbb{R}^d} \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{\xi_n\sim\mathcal{D}_n}\ f_n(x;\xi_n),$$

$x\in\mathbb{R}^d$ is the model parameter to be learned; $f_n(x;\xi_n)$ is the local objective function of client n based on the mini-batch dataset ξ_n sampled from local training dataset $\mathcal{D}_n$.

Each client will perform bellow three stages at each round:

- Stage 1 (local training): client n takes a local SGD step to obtain an intermediate update

$$x_n^{t+\frac{1}{2}} = x_n^t - \eta^t \nabla f_n(x_n^t;\xi_n^t)$$

- Stage 2 (communication): send updates and receive updates from neighbors

$$X_n = \{x_n^{t+\frac{1}{2}}\} \cup \{x_m^{t+\frac{1}{2}} | m\in\mathcal{N}_n\}$$

- Stage 3 (aggregation): get a new local model

$$x_n^{t+1} = \sum_{x_i\in X_n} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} x_i^{t+\frac{1}{2}}$$

However, in practice, not all clients will follow the prescribed stages. The unexpected behaviors performed by the so-called Byzantine clients (or malicious clients) are usually assumed to be omniscient and arbitrarily malicious.

3.Local Performance Checking

We propose the Local Performance Checking (LPC) under the DFL, leveraging the fact that each client in DFL functions as a parameter server with a local dataset.

Algorithm 1 Local Performance Checking (LPC) performed at client n

Input: $X_n = \{x_n^{t+\frac{1}{2}}, x_m^{t+\frac{1}{2}} (\forall m\in\mathcal{N}_n)\}$ consists of client n's local model $x_n^{t+\frac{1}{2}}$, updates from neighbors $x_m^{t+\frac{1}{2}}$ $(\forall m\in\mathcal{N}_n)$, sampling fraction α; the communication round t.

Output: new local model x_n^{t+1}.

1. sample α of the local training data to form the local checking dataset $\mathcal{TD}_n$;
2. for all $x_i^{t+\frac{1}{2}} \in X_n$ do
   get the accuracy a_i use $\mathcal{TD}_n$;
   end for
3. calculate the filtering threshold $\beta = \frac{1}{|X_n|}\sum_{i=1}^{|X_n|} a_i$.
4. filter out $x_i^{t+\frac{1}{2}}$ if $a_i \le \beta$ to get a new update set.
5. update local model $x_n^{t+1} = \frac{1}{|\bar{X}_n|}\sum_{j\in\bar{X}_n} x_j^{t+\frac{1}{2}}$.
6. return x_n^{t+1}.

The LPC algorithm is presented in **Algorithm 1**, which consists the following four steps:

- Step 1: Build the performance test dataset $\mathcal{TD}_n$;
- Step 2: Perform performance checking;
- Step 3: Determine the filtering threshold and filter malicious updates;
- Step 4: Aggregation;

4.Experiment

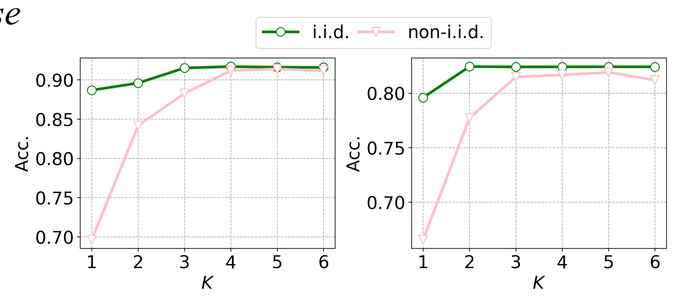
We evaluated the effectiveness of LPC under different scenarios which is the combination of **Network Topologies, Datasets, Attacks, Robust Aggregation Rules**, etc.

- Comparison (test accurcy) with other robust aggregation algorithm

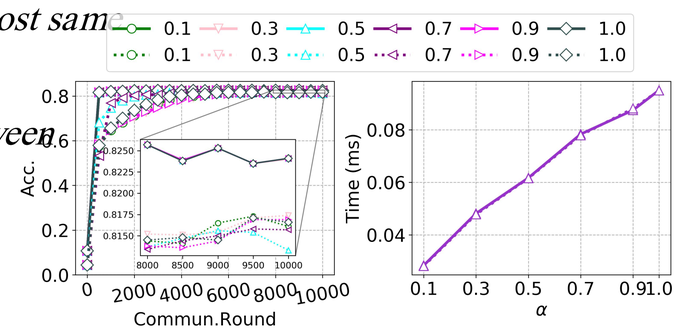
Attack	Data	No-AT	Median	TM	Krum	SCC	*LPC*
Gaussian Attack	M(i.i.d.)	0.91609	0.91572	0.91600	0.91360	0.91601	0.91589
	M(non-i.i.d.)	0.91563	0.90968	0.91494	0.85008	0.91681	0.91732
	FM(i.i.d.)	0.82350	0.82200	0.82289	0.82295	0.82281	0.82398
	FM(non-i.i.d.)	0.82378	0.73589	0.81770	0.70413	0.81570	0.79805
Sign-Flipping Attack	M(i.i.d.)	0.91609	0.89076	0.88843	0.91360	0.78295	0.91589
	M(non-i.i.d.)	0.91563	0.83079	0.83887	0.85008	0.40238	0.91732
	FM(i.i.d.)	0.82350	0.79705	0.79215	0.82295	0.43322	0.82397
	FM(non-i.i.d.)	0.82378	0.73402	0.69974	0.70413	0.25796	0.76371
A Little Is Enough Attack	M(i.i.d.)	0.91609	0.91640	0.91627	0.91616	0.91629	0.91635
	M(non-i.i.d.)	0.91563	0.89249	0.91091	0.89714	0.91485	0.90358
	FM(i.i.d.)	0.82350	0.82225	0.82331	0.82182	0.82366	0.82369
	FM(non-i.i.d.)	0.82378	0.76063	0.81211	0.79946	0.81752	0.78445
Same-Value Attack	M(i.i.d.)	0.91609	0.89115	0.88772	0.91260	0.78748	0.91586
	M(non-i.i.d.)	0.91563	0.82740	0.83974	0.09907	0.63989	0.91733
	FM(i.i.d.)	0.82350	0.79763	0.79222	0.82190	0.63132	0.82388
	FM(non-i.i.d.)	0.82378	0.73398	0.70110	0.09205	0.53360	0.61643

* M is MNIST, FM is Fashin MNIST, i.i.d. is independent and identically distributed, non-i.i.d. is non-iindependent and identically distributed, No-AT is no attack with mean, TM is Trimmed Mean, SCC is Self-Centered Clipping, LPC is Local Performance Checking.

- Impact of local training round K.
  - ✓ *Test accuracy increase with K increase.*



- Impact of test dataset sample ratio α.
  - ✓ *Different α have almost same Test accuracy.*
  - ✓ *The relationship between time and α is linear.*



We also conducted several group experiments to check the impact of network density ρ and the impact of the number of malicious nodes M. The code-level details are available at https://github.com/KaichuangZhang/Local-Performance-Checking.

5.Conclusion

- In this paper, we proposed local performance checking to achieve Byzantine robustness for DFL. The proposed algorithm leveraged the unique capability of DFL to directly obtain performance test datasets for local performance checking. A performance metric is employed at each client to determine whether the update is malicious or benign. Numerous experiments show that the proposed algorithm is effective.
- At the same time, we identified some future directions for future improvements, including reducing time complexity, enhancing resistance to knowledge isolation attacks, and developing more robust selection methods.
  - ✓ Special Attacks. Like Isolation Attacks and Backdoor Attacks;
  - ✓ Computation Cost.
  - ✓ Better Threshold.

ICONIP 2024

31st International Conference on Neural Information Processing

December 2–6, 2024 · Auckland, New Zealand iconip2024.org

Presenting Author

Ping Xu
ping.t.xu@utrgv.edu