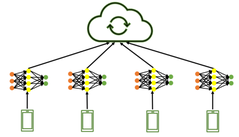
Poisoning Attack to Byzantine-Robust Federated Learning

Fang et al. and Cao et al.



Abstract

[1]: (1) Systematic study on attacking Byzantine-robust Federated Learning(FL); (2) Local model poisoning attacks(LMP attack); (3) Two defenses against LMP attack.
[2]: (1) FLTrust to achieve Byzantine robustness against attackers; (2) Evaluate FLTrust against existing attacks; (3) Evaluatate FLTrust against adaptive attacks.

Keywords: FL, Byzantine-robust, Poisioning Attack.

Introduction

Byzantine-robust federated learning works well and securely when some clients are malicious. It applies the byzantine-robust aggregation Rules(Krum,Bulyan,trimmed mean,and median).
Two categories of poisoning attacks:

- Data poisoning attacks aims to pollute the training data.
- Local model poisoning attacks aims to poison the local models or their updates.

Model Poisoning Attacks[1]

Optimazation Problem: Attackers aim to deviates from the most opposite direction

$$\begin{aligned} &\max_{\mathbf{w}'_1,\cdots,\mathbf{w}'_c} \mathbf{s}^T(\mathbf{w}-\mathbf{w}') \\ \text{subject to } &\mathbf{w} = \mathcal{A}(\mathbf{w}_1,\cdots,\mathbf{w}_c,\mathbf{w}_{c+1},\cdots,\mathbf{w}_m) \\ &\mathbf{w}' = \mathcal{A}(\mathbf{w}'_1,\cdots,\mathbf{w}'_c,\mathbf{w}_{c+1},\cdots,\mathbf{w}_m) \end{aligned}$$

A New LMP Attack Technique: Attackers aim to compromise the integrity of the learning process in the training phase.

- Challenge: how to craft the local models sent from the compromised worker to the master.
- Solution: formulate crafting local models as solving an optimization problem in each iteration of FL.

Two New Defenses: Generalize two types of defense (RONI and TRIM) to so-called ERR and LFR defend against our local model poisoning attacks.

- Error Rate based Rejection (ERR): removes the local models that have large negative impact on the error rate of the global model
- Loss Function based Rejection (LFR): removes the local models that result in large loss

Data Poisoning Attacks[2]

FLTrust: the first FL method that bootstraps trust to achieve Byzantine robustness against malicious clients

- The server maintains a training model based on a self-collected clean small training dataset to bootstrap trust and detect the deviation of the malicious gradient.
- ReLU-clipped cosine similarity (of gradient vector) based trust score.
- Normalizing the magnitudes of local model updates to mitigate the attack of scaling the magnitudes of the local model.

Rule of thumb: The direction of attacked gradient vector deviates from the true ones. The server can maintain a true and trusted model and update it using a small clean dataset(called root dataset) collected by manual labeling

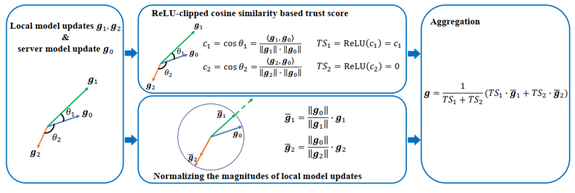


Figure 2: FLTrust Overview

Secure policy in LMP[1]

Paper [1] fufills the following **Five Security Policies**.

Threat Model: Assume the attacker has control of some worker devices and manipulates the local model parameters sent from these devices to the master device during the learning process.
Cost First Principle: Althrough we note that our defenses make the global model slower to learn and adapt to new data as that data may be identified as from potentially malicious local models, but the FL system is more secure due to the defense.
Detection and Response: Design two defense methods(Error Rate based Rejection(ERR) and Loss Function based Rejection (LFR)) to detect the poisoning attacked gradient.
Defense in Depth: Combine the proposed ERR and LFR methods together to achieve two layers of defense to remove the attacked local models in two depth.
Open Design: Assume that all attackers fully understand the mechanism of federated learning and the design of these two local data posioning attack defenses.

	No attack	Krum	Trimmed mean
Krum	0.14	0.72	0.13
Krum + ERR	0.14	0.62	0.13
Krum + LFR	0.14	0.58	0.14
Krum + Union	0.14	0.48	0.14
Trimmed mean	0.12	0.15	0.23
Trimmed mean + ERR	0.12	0.17	0.21
Trimmed mean + LFR	0.12	0.18	0.12
Trimmed mean + Union	0.12	0.18	0.12
Median	0.13	0.17	0.19
Median + ERR	0.13	0.21	0.25
Median + LFR	0.13	0.20	0.13
Median + Union	0.13	0.19	0.14

Figure 1: Defence, in Depth

Secure policy in FLTrust[2]

Paper [2] fulfills the following **Six Security Policies**.

Threat Model: Attackers control some malicious clients, which can be fake clients injected by the attacker or genuine ones compromised by the attacker, but they does not compromise the server.
Cost First Principle: A small root dataset for server to maintain a trusted model with gradient updating direction can bootstrap the trust of the whole FL system achieving the byzantine robustness.
Detection and Response: ReLU-clipped cosine similarity based trust score. If the direction of a local model update is more similar to that of the server model update, then the direction of the local model update may be more "promising".
Defense in Depth: The attacker can also scale the magnitudes of the updated gradient on the malicious clients by a large factor such that they dominate the global model update. The method of normalizing the magnitudes of local model updates are proposed to mitigate that attack.
Open Design: Assuming that all attackers fully understand the design of the FLtrust, but the attack still can compromise the system.
Secure by Design: Design a new Byzantine-robust aggregation rule in FLTrust to incorporate the root of trust.

References

[1] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to {Byzantine-Robust} federated learning. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1605–1622, 2020.

[2] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. The Internet Society, 2021.

Future Research

(1): Scale the limition of [1]'s current work of untargeted attacks to targeted attacks.

(2): Designing stronger local model poisoning attacks to FLTrust framework.[2]

Contact and ID

Name vio1etus
Blog vio1etus's blog
Link https://vio1etus.github.io/blog/

Made by LaTex 2022.6.30