Secure Federated Learning

Neil Gong
Department of Electrical and Computer Engineering
Department of Computer Science (secondary appointment)
Duke University

This talk is available on YouTube: https://www.youtube.com/watch?v=LP4uqW18yA0
Conventional Paradigm: Centralized Learning

Google, Apple, Facebook

Machine learning model

Clients

Smartphone, IoT devices, self-driving cars
Challenges of Centralized Learning

• Server data breaches

• High communications cost
  • Intolerable for resource-constrained clients
    • Smartphone
    • IoT
Federated Learning

• Data stay locally on clients
• Clients train models locally
• Clients send models or updates to server
• Real-world deployment

Artificial intelligence / Machine learning

How Apple personalizes Siri without hoovering up your data

The tech giant is using privacy-preserving machine learning to improve its voice assistant while keeping your data on your phone.

by Karen Hao

December 11, 2019
This Talk

What are the security issues of federated learning

How to build secure federated learning
Road Map

• Part I: Local model poisoning attacks to federated learning

• Part II: Secure federated learning via trust bootstrapping

• Part III: Provably secure federated learning
Road Map

• Part I: Local model poisoning attacks to federated learning

• Part II: Secure federated learning via trust bootstrapping

• Part III: Provably secure federated learning
Federated Learning Background

Google’s FedAvg: \( w = \frac{1}{n} \sum_{i=1}^{n} w_i \)

Step I. Send global model to clients

Step II. Train local models and send them to server

Step III. Aggregate local models

Equivalent to send local model updates \( w_i - w \) to server
Federated Learning is Vulnerable to Poisoning Attacks
Federated Learning is Vulnerable to Poisoning Attacks

Fake or compromised genuine clients
Fake clients can be many
Federated Learning is Vulnerable to Poisoning Attacks

Data poisoning attack

Malicious client

Fake or compromised genuine clients
Fake clients can be many
Federated Learning is Vulnerable to Poisoning Attacks

Local model poisoning attack

Data poisoning attack

Malicious client

Fake or compromised genuine clients
Fake clients can be many
Federated Learning is Vulnerable to Poisoning Attacks

Fake or compromised genuine clients
Fake clients can be many
Byzantine-robust Federated Learning as Defense

• Byzantine-robust aggregation rule
  • Krum
  • Trimmed mean
  • Median

• Key idea
  • Remove “outlier” local models

• Theoretical guarantee
  • Various assumptions
    • IID data, smooth loss function, etc.
  • Bound change of global model parameters caused by malicious clients
An Example: Median

Client 1

\[ w_{11} \quad w_{12} \quad \ldots \quad w_{1m} \]

Client 2

\[ w_{21} \quad w_{22} \quad \ldots \quad w_{2m} \]

\vdots

Client n

\[ w_{n1} \quad w_{n2} \quad \ldots \quad w_{nm} \]

Server

\[ w_1 \quad w_2 \quad \ldots \quad w_m \]
Our Work

Byzantine-robust federated learning is vulnerable to local model poisoning attacks

Increase testing error rate of global model

Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. “Local Model Poisoning Attacks to Byzantine-Robust Federated Learning”. In USENIX Security Symposium, 2020
Threat Model

• Attacker’s goal
  • High testing error rate

• Attacker’s capability:
  • Access to malicious clients
    •Fake clients
    • Compromised genuine clients
  • Send arbitrary local models

• Attacker’s knowledge:
  • Full vs. Partial knowledge
    • Data on all vs. malicious clients
  • Aggregation rule
    • Yes or no
Our Idea

No attack: global model changes along some direction
Our Idea

Deviate global model the most towards inverse of the direction

$w'_{1}$

$w'$

$w_{prev}$

$w$

$w_{2}$
Formulate Optimization Problem

\[
\max_{w'_1, \ldots, w'_c} s^T (w - w')
\]

Subject to

\[
w = \mathcal{A}(w_1, \ldots, w_c, w_{c+1}, \ldots, w_n)
\]

\[
w' = \mathcal{A}(w'_1, \ldots, w'_c, w_{c+1}, \ldots, w_n)
\]

Used in all or multiple iterations

Applicable to any aggregation rule

Poisoned local models on malicious clients

Global model before attack

Update direction

Global model after attack

Maximize deviation of global model

Global model aggregation before attack

Global model aggregation after attack
Solving the Optimization Problem

• Full knowledge
  • $w_1, \ldots, w_c, w_{c+1}, \ldots, w_n$ are known
  • Solve the optimization problem using them

• Partial knowledge
  • Only $w_1, \ldots, w_c$ are known
  • Use them to estimate $w$

• Unknown aggregation rule
  • Attacker assumes one
Experimental Setup

• 100 clients
  • 20% malicious

• Datasets:
  • MNIST
  • Fashion-MNIST
  • CH-MNIST
  • Breast Cancer Wisconsin (Diagnostic)

• Non-IID data on clients
  • Non-IID: not Independently and Identically Distributed
Experimental Setup

• 100 clients
  • 20% malicious

• Datasets:
  • MNIST
  • Fashion-MNIST
  • CH-MNIST
  • Breast Cancer Wisconsin (Diagnostic)

• Non-IID data on clients
  • Non-IID: not Independently and Identically Distributed
Our Attack is Effective

<table>
<thead>
<tr>
<th>Method</th>
<th>NoAttack</th>
<th>Gaussian</th>
<th>LabelFlip</th>
<th>Partial</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>Krum</td>
<td>0.11</td>
<td>0.10</td>
<td>0.10</td>
<td>0.75</td>
<td>0.77</td>
</tr>
<tr>
<td>Trimmed Mean</td>
<td>0.06</td>
<td>0.07</td>
<td>0.07</td>
<td>0.14</td>
<td>0.23</td>
</tr>
<tr>
<td>Median</td>
<td>0.06</td>
<td>0.06</td>
<td>0.16</td>
<td>0.28</td>
<td>0.32</td>
</tr>
</tbody>
</table>

Byzantine-robust methods
Our Attack is Effective

No attack

<table>
<thead>
<tr>
<th></th>
<th>NoAttack</th>
<th>Gaussian</th>
<th>LabelFlip</th>
<th>Partial</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>Krum</td>
<td>0.11</td>
<td>0.10</td>
<td>0.10</td>
<td>0.75</td>
<td>0.77</td>
</tr>
<tr>
<td>Trimmed Mean</td>
<td>0.06</td>
<td>0.07</td>
<td>0.07</td>
<td>0.14</td>
<td>0.23</td>
</tr>
<tr>
<td>Median</td>
<td>0.06</td>
<td>0.06</td>
<td>0.16</td>
<td>0.28</td>
<td>0.32</td>
</tr>
</tbody>
</table>
**Our Attack is Effective**

Add Gaussian noise to local models

<table>
<thead>
<tr>
<th>Method</th>
<th>NoAttack</th>
<th>Gaussian</th>
<th>LabelFlip</th>
<th>Partial</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>Krum</td>
<td>0.11</td>
<td>0.10</td>
<td>0.10</td>
<td>0.75</td>
<td>0.77</td>
</tr>
<tr>
<td>Trimmed Mean</td>
<td>0.06</td>
<td>0.07</td>
<td>0.07</td>
<td>0.14</td>
<td>0.23</td>
</tr>
<tr>
<td>Median</td>
<td>0.06</td>
<td>0.06</td>
<td>0.16</td>
<td>0.28</td>
<td>0.32</td>
</tr>
</tbody>
</table>
## Our Attack is Effective

Flip labels of local training data

<table>
<thead>
<tr>
<th>Method</th>
<th>NoAttack</th>
<th>Gaussian</th>
<th>LabelFlip</th>
<th>Partial</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>Krum</td>
<td>0.11</td>
<td>0.10</td>
<td>0.10</td>
<td>0.75</td>
<td>0.77</td>
</tr>
<tr>
<td>Trimmed Mean</td>
<td>0.06</td>
<td>0.07</td>
<td>0.07</td>
<td>0.14</td>
<td>0.23</td>
</tr>
<tr>
<td>Median</td>
<td>0.06</td>
<td>0.06</td>
<td>0.16</td>
<td>0.28</td>
<td>0.32</td>
</tr>
</tbody>
</table>
Our Attack is Effective

<table>
<thead>
<tr>
<th></th>
<th>NoAttack</th>
<th>Gaussian</th>
<th>LabelFlip</th>
<th>Partial</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>Krum</td>
<td>0.11</td>
<td>0.10</td>
<td>0.10</td>
<td>0.75</td>
<td>0.77</td>
</tr>
<tr>
<td>Trimmed Mean</td>
<td>0.06</td>
<td>0.07</td>
<td>0.07</td>
<td>0.14</td>
<td>0.23</td>
</tr>
<tr>
<td>Median</td>
<td>0.06</td>
<td>0.06</td>
<td>0.16</td>
<td>0.28</td>
<td>0.32</td>
</tr>
</tbody>
</table>
Our Attack is Effective

<table>
<thead>
<tr>
<th>Method</th>
<th>NoAttack</th>
<th>Gaussian</th>
<th>LabelFlip</th>
<th>Partial</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>Krum</td>
<td>0.11</td>
<td>0.10</td>
<td>0.10</td>
<td>0.75</td>
<td>0.77</td>
</tr>
<tr>
<td>Trimmed Mean</td>
<td>0.06</td>
<td>0.07</td>
<td>0.07</td>
<td>0.14</td>
<td>0.23</td>
</tr>
<tr>
<td>Median</td>
<td>0.06</td>
<td>0.06</td>
<td>0.16</td>
<td>0.28</td>
<td>0.32</td>
</tr>
</tbody>
</table>

Our attacks can effectively increase testing error rates
Impact of #Malicious Clients

Our attacks are more effective with more malicious clients.
Impact of Degree of Non-IID

Our attacks are more effective when clients’ data are more Non-IID
Our Attacks Transfer between Aggregation Rules

<table>
<thead>
<tr>
<th></th>
<th>Krum</th>
<th>Trimmed mean</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>No attack</td>
<td>0.14</td>
<td>0.12</td>
<td>0.13</td>
</tr>
<tr>
<td>Krum attack</td>
<td>0.70</td>
<td>0.15</td>
<td>0.18</td>
</tr>
<tr>
<td>Trimmed mean attack</td>
<td>0.14</td>
<td>0.25</td>
<td>0.20</td>
</tr>
</tbody>
</table>
Comparing with Data Poisoning Attacks

<table>
<thead>
<tr>
<th></th>
<th>NoAttack</th>
<th>DataPoisoning</th>
<th>Partial</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>Krum</td>
<td>0.23</td>
<td>0.24</td>
<td>0.85</td>
<td>0.89</td>
</tr>
<tr>
<td>Trimmed Mean</td>
<td>0.12</td>
<td>0.12</td>
<td>0.27</td>
<td>0.32</td>
</tr>
<tr>
<td>Median</td>
<td>0.13</td>
<td>0.13</td>
<td>0.19</td>
<td>0.21</td>
</tr>
</tbody>
</table>

Data poisoning attacks are ineffective for Byzantine-robust methods

Our attacks are effective
Summary

• Proposed a general framework to attack federated learning

• Existing Byzantine-robust federated learning is vulnerable to local model poisoning attacks
Road Map

• Part I: Local model poisoning attacks to federated learning

• Part II: Secure federated learning via trust bootstrapping

• Part III: Provably secure federated learning
Root Cause of Insecurity

No root trust

Every client could be malicious
Our FLTrust: Bootstrapping Trust

• Server collects a small, clean training dataset

• Server maintains a server model
  • Like how a client maintains a local model

• Use server model to bootstrap trust
  • Assign trust scores to clients

Revisiting Federated Learning Background

Step I. Send global model to clients

Step II. Train local models and send their updates to server

Step III. Aggregate local model updates and update global model

Server model update

Global model $w$

Local model $w_i$
Local model updates $g_1, g_2$

server model update $g_0$

Our Aggregation Rule

Normalizing the magnitudes of local model updates

ReLU-clipped cosine similarity based trust score (TS)

$T S_1 = \text{ReLU}(c_1) = c_1$

$T S_2 = \text{ReLU}(c_2) = 0$

Aggregation

$g = \frac{1}{T S_1 + T S_2} (T S_1 \cdot \overline{g}_1 + T S_2 \cdot \overline{g}_2)$

$w = w - \beta \cdot g$
Theoretical Analysis

Under certain assumptions, for an arbitrary number of malicious clients, the difference between the global model learnt by FLTrust and the optimal global model under no attacks is bounded.
Empirical Results

MNIST
100 clients, 20 malicious
Server’s training dataset: 100 examples sampled from MNIST

State-of-the-art method in non-adversarial settings

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>FedAvg</th>
<th>Krum</th>
<th>Trim-mean</th>
<th>Median</th>
<th>FLTrust</th>
</tr>
</thead>
<tbody>
<tr>
<td>No attack</td>
<td>0.04</td>
<td>0.10</td>
<td>0.06</td>
<td>0.06</td>
<td>0.05</td>
</tr>
<tr>
<td>Label flipping attack</td>
<td>0.06</td>
<td>0.10</td>
<td>0.06</td>
<td>0.06</td>
<td>0.05</td>
</tr>
<tr>
<td>Krum attack</td>
<td>0.10</td>
<td>0.91</td>
<td>0.14</td>
<td>0.15</td>
<td>0.05</td>
</tr>
<tr>
<td>Trim attack</td>
<td>0.28</td>
<td>0.10</td>
<td>0.23</td>
<td>0.43</td>
<td>0.06</td>
</tr>
</tbody>
</table>

Our FLTrust is robust against poisoning attacks
Adaptive Attack

\[
\begin{align*}
\max_{w'_1, \ldots, w'_c} & \quad s^T (w - w') \\
\text{Subject to} & \quad w = A(w_1, \ldots, w_c, w_{c+1}, \ldots, w_n) \\
& \quad w' = A(w'_1, \ldots, w'_c, w_{c+1}, \ldots, w_n)
\end{align*}
\]

Applicable to any aggregation rule
Our FLTrust is Robust against Adaptive Attack
Summary

• The server can enhance security of federated learning via collecting a small, clean training dataset to bootstrap trust
Road Map

• Part I: Local model poisoning attacks to federated learning

• Part II: Secure federated learning via trust bootstrapping

• Part III: Provably secure federated learning
Limitations of Byzantine-robust Federated Learning

• Bound change in global model parameters caused by malicious clients
  • Under assumptions
    • IID data on clients
    • Smooth loss function
    • ...

• Limitations
  • Assumptions do not hold
  • Not bound testing error rate or accuracy
Our Provably Secure Federated Learning

• Guarantee a lower bound of testing accuracy

• Only assumption
  • Bounded #malicious clients

Defining Provable Security

Label predicted for $x$ when the global model is trained on $C$

$$h(C', x) = h(C, x) \quad \text{for any } C', \ #\text{malicious clients} \leq m^*$$

A set of benign clients  \quad \text{Testing input}

A federated learning algorithm is provably secure if its predicted label for a testing input is not affected by a bounded number of malicious clients

$m^*: \text{certified security level for } x$
Our Ensemble Federated Learning: the First Provably Secure Method

• Training
  • $n$ clients
  
  • Select $k$ clients randomly and train a global model
    • Use any federated learning method, e.g., FedAvg
  
  • Repeat to train $N$ global models

• Testing
  • Majority vote of the $N$ global models to predict label of $x$
Provable Security: Intuition

Training Phase

Testing Phase

$w_1 \quad w_2 \quad w_3$

$C_1 \quad C_2 \quad C_3 \quad C_4 \quad C_5$

$\text{Majority Vote}$

$x \quad y \quad y'$
Given $C$ and $x$, we can derive the certified security level $m^*$ for $x$

Our derived certified security level is tight
Evaluation Metric: Certified Accuracy @ m

• Fraction of testing inputs whose
  • Labels are correctly predicted
  • Certified security levels are at least m

• A lower bound of testing accuracy
  • #malicious clients ≤ m
  • No matter what attacks are used!
FedAvg vs. Ensemble FedAvg

MNIST dataset, 1,000 clients
A moderate number of global models are enough.
Summary

• Ensemble federated learning is provably secure against bounded number of malicious clients

• Achieve certified accuracy
  • A lower bound of testing accuracy
    • No matter what attacks are used
Conclusion

• Part I: Local model poisoning attacks to federated learning
  • “Local Model Poisoning Attacks to Byzantine-Robust Federated Learning”. In Usenix Security Symposium, 2020.

• Part II: Secure federated learning via trust bootstrapping
  • “FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping”. In NDSS, 2021.

• Part III: Provably secure federated learning
  • “Provably Secure Federated Learning against Malicious Clients”. In AAAI, 2021.

Acknowledgements

Xiaoyu Cao         Minghong Fang
Jinyuan Jia        Jia Liu