Byzantine-Robust Decentralized Federated Learning with Trustworthiness

Gilbert Paul Neuner

February 28, 2025

1 Introduction

This written qualifying exam proposes a decentralized federated learning algorithm robust to poisoning attacks. In section 2, background on federated learning, poisoning attacks and defense strategies, and decentralized federated learning will be given. In section 3, two formulations of the main algorithm will be given (one centralized and one decentralized), as well as a multi-stage approach for using the main algorithm. In section 4, the main algorithm will be compared with other popular robust federated learning methods, on the task of adversarial sparse linear regression with synthetic data. In section 5, three future directions for this project will be outlined.

2 Background

2.1 Federated Learning

Federated learning refers to the setting in which multiple clients each have their own training dataset, and wish to work together to learn a model while keeping their training dataset private. Specifically:

- K denotes the number of clients, and $\mathcal{M}_k$ denotes the k-th client. The k-th client privately holds $D^{(k)}$, the k-th **local training dataset**.
- Superscript (k) indexes clients (e.g., $D^{(k)}$).
- $D = \cup_{i=1}^{K} D^{(k)}$ denotes the **joint training dataset**. Each element $d \in D$ is i.i.d. from an unknown distribution $\mathcal{X}$.
- Given a model $w \in \mathbb{R}^p$, $f(D, w)$ denotes the **empirical loss**. The goal is to find the **optimal model** $w^* = \arg\min_w \mathbb{E}_{D \sim \mathcal{X}} f(D, w)$. In practice, the goal is to learn $w^* = \arg\min_w f(D, w)$. The challenge is that the k-th client only has access to $D^{(k)}$, and can only evaluate **local empirical loss** $f(D^{(k)}, w)$.
- Federated learning may be **centralized**, in which case clients communicate with a **central server** (denoted by $\mathcal{C}$), or it may be **decentralized**, in which case clients communicate with each other directly. In centralized federated learning, clients learn a shared **global model** w, whereas in decentralized federated learning, clients each learn a **local model** $w^{(k)}$.

- Iteration number is indexed by superscript (s) outside of parentheses. For example, in centralized federated learning, $(w)^{(s)}$ denotes the global model at the s-th iteration, and in decentralized federated learning, $(w^{(k)})^{(s)}$ denotes the k-th local model at the s-th iteration. Sometimes for the sake of brevity the iteration index is omitted.

An iteration of centralized federated learning typically has the following steps:

- **Local Gradient:** Clients broadcast to the central server their **local gradient** $g^{(k)} = \nabla_w f(D^{(k)}, w)|_{w=w^{(s)}}$, $k = 1, \dots, K$, the gradient of the k-th local empirical loss evaluated at the current global model. (Sometimes, the empirical loss is not differentiable but can be split into a sum of a differentiable part and a non-differentiable part. In this case, the local gradient is of the differentiable part.)

- **Aggregation:** The central server aggregates local gradients $g^{(k)}$, $k = 1, \dots, K$ to form a **global update** g:

$$g = \text{Aggregate}(g^{(1)}, \dots, g^{(K)}).$$

  Aggregate is often some form of a weighted average.

- **Global Model Update:** The central server updates the global model

$$(w)^{(s+1)} = \text{ModelUpdate}((w)^{(s)}, g, (\alpha)^{(s)}),$$

  where $(\alpha)^{(s)}$, $s \in \mathbb{N}$ is the learning rate. Then, the central server broadcasts the new global model $(w)^{(s+1)}$ to each client. For example, in the case of gradient descent,

$$\text{ModelUpdate}((w)^{(s)}, g, (\alpha)^{(s)}) = (w)^{(s)} - (\alpha)^{(s)} g.$$

The main example of centralized federated learning in the non-adversarial setting is **FedAvg** [MMR16]. It follows the steps above, where the aggregation scheme is a weighted average of the local gradients and the weights are based on local sample sizes:

$$g = \sum_{k=1}^{K} \frac{|D_k|}{|D|} g^{(k)}.$$

2.2 Poisoning Attacks

Federated learning methods such as FedAvg are susceptible to **poisoning attacks**, in which each client is either **benign** or **adversarial**, but it is unknown which clients are benign and which are adversaries. Adversaries send arbitrary vectors denoted by $*$ instead of true local updates to prevent convergence. The framework is the same as before, except the client gradient is now

$$g^{(k)} = \begin{cases} \nabla_w f(D^{(k)}, w)|_{w=w^{(s)}}, & \mathcal{M}_k \text{ is benign;} \\ *, & \mathcal{M}_k \text{ is an adversary.} \end{cases}$$

A single adversary can prevent FedAvg from converging. If the K-th client is an adversary and wants the next global update to be the vector h, it can broadcast

$$g^{(K)} = \frac{|D|}{|D_K|}\left(h - \sum_{k=1}^{K-1} \frac{|D_k|}{|D|} g^{(k)}\right)$$

to the central server.

Two defense strategies are Byzantine-robust aggregation and similarity-based methods. **Byzantine-robust aggregation** (called such because of the Byzantine generals problem [LSP82]) is a defense strategy in which the aggregation scheme is designed to be robust to the corruption introduced by adversaries. Typically, the aggregation scheme involves removing extreme values.

Two examples are median and trimmed mean [YCR18]. In **median**, local gradients are aggregated

$$g = \operatorname{median}(g^{(1)}, \ldots, g^{(K)}),$$

where the median is componentwise, i.e., the i-th component of g is the median of the i-th components of $g^{(1)}$, …, $g^{(K)}$. Similarly, in **trimmed mean**, local gradients are aggregated

$$g = \operatorname{trmean}_f(g^{(1)}, \ldots, g^{(K)}),$$

where the trimmed mean is componentwise, and the f-trimmed mean of scalars x_1, …, x_K is defined to be the mean of x_1, …, x_K after its largest f and smallest f elements are removed.

Another example is **Krum** [BEG17], which is defined as follows. Denote by Γ_i the $K-f-2$ closest local updates to $g^{(i)}$ in Euclidean distance. Define the score $s(i) = \sum_{j \in \Gamma_i} \|g^{(i)} - g^{(j)}\|_2^2$. Local gradients are aggregated

$$g = \mathrm{Kr}(g^{(1)}, \ldots, g^{(K)}),$$

where Kr selects the $g^{(i)}$ with minimal score. The parameter f should be chosen to be greater than the number of adversaries.

A **similarity measure** is a real-valued function that quantifies the similarity between two objects. In a **similarity-based method**, similarity measures such as Euclidean distance [CCL19], cosine similarity [CFL20], k-means [LWW21], or Pearson correlation coefficient [LLX21] are used to compare local gradients. The central server can use similarity between gradients to determine which clients it trusts and/or which clients to exclude from aggregation. In some methods similarity measures are used directly as coefficients during aggregation.

For example, in **Sniper** [CCL19], at every iteration, the server computes Euclidean distances between every pair of local gradients. Then, it constructs a graph, where two clients are connected if the distance between their local gradients is less than a threshold τ. If the maximum clique (denoted by $Honest$) of the resulting graph has size larger than $K/2$, then aggregation is FedAvg restricted to those clients belonging to $Honest$:

$$g = \sum_{k \in Honest} \frac{|D_k|}{|D|} g^{(k)}.$$

If the maximum clique $Honest$ is not larger than $K/2$, the central server reduces the threshold τ, repeatedly until $Honest$ is larger than $K/2$.

Similarity measures can also be used in conjunction with Byzantine-robust aggregation. For example, in **FLTrust**, [CFL20], it is assumed that the central server has its own training dataset $D^{(0)}$, on which it can compute its own local gradient, denoted by $g^{(0)}$. Then, the aggregation rule is given by

$$\frac{1}{\sum_{k=1}^{K} TS^{(k)}} \sum_{k=1}^{K} TS^{(k)} \bar{g}^{(k)}$$

where $TS^{(k)}$, called **trust score**, is defined by

$$TS^{(k)} = ReLU\left(\frac{\langle g^{(0)}, g^{(k)} \rangle}{\|g^{(0)}\|_2 \|g^{(k)}\|_2}\right)$$

and

$$\bar{g}^{(k)} = \frac{\|g^{(0)}\|_2}{\|g^{(k)}\|_2} g^{(k)}.$$

The term $\frac{\langle g^{(0)}, g^{(k)} \rangle}{\|g^{(0)}\|_2 \|g^{(k)}\|_2}$ is the cosine of the angle between the local updates of the central server $g^{(0)}$ and k-th client $g^{(k)}$, so FLTrust uses cosine similarity. Passing cosine similarity through *ReLU* sends the similarity measure between $g^{(0)}$ and $g^{(k)}$ to 0 whenever the angle between $g^{(0)}$ and $g^{(k)}$ exceeds 90 degrees. These similarity measures are applied to $g^{(k)}$, normalized to have magnitude $\|g^{(0)}\|_2$.

2.3 Decentralized Federated Learning

In decentralized federated learning, clients occupy a **communication graph**, and can only communicate with their neighbors. The neighborhood set of the i-th client is denoted by $\text{ne}(i)$. Let $\mathcal{M}_i$ be benign. An iteration of decentralized federated learning typically has the following steps:

- **Local Gradient:** The i-th client computes a **local gradient**

  $$g^{(i,i)} = \nabla_w f(D^{(i)}, w)|_{w=(w^{(i)})^{(s)}},$$

  the gradient of the i-th local empirical loss evaluated at the i-th current local model. Optionally, for $j \in \text{ne}(i)$, the j-th client computes a **local gradient** $g^{(i,j)}$,

  $$g^{(i,j)} = \begin{cases} \nabla_w f(D^{(j)}, w)|_{w=(w^{(i)})^{(s)}}, & \mathcal{M}_j \text{ is benign;} \\ *, & \mathcal{M}_j \text{ is an adversary.} \end{cases}$$

  the gradient of the j-th local empirical loss evaluated at the i-th current local model. The j-th client broadcasts $g^{(j,j)}$ and/or $g^{(i,j)}$ to the i-th client.

- **Aggregation:** The i-th client forms an **aggregated update** $g^{(i)}$. The aggregated update may depend on local gradients and/or local models:

  $$g^{(i)} = \text{Aggregate}(\{(w^{(j)})^{(s)}\}_{j \in \{i\} \cup \text{ne}(i)}, \{g^{(j,j)}\}_{j \in \{i\} \cup \text{ne}(i)}, \{g^{(i,j)}\}_{j \in \text{ne}(i)}).$$

- **Local Model Update:** The i-th client updates its local model

  $$(w^{(i)})^{(s+1)} = \text{ModelUpdate}((w^{(i)})^{(s)}, g^{(i)}, (\alpha^{(i)})^{(s)}),$$

  where $(\alpha^{(i)})^{(s)}$, $s \in \mathbb{N}$ is the learning rate of the i-th client. Then, the i-th client broadcasts its new local model to its neighbors.

Some disadvantages to decentralized federated learning is that it requires clients to perform more communications per iteration (the size of the neighborhood set in the decentralized setting, and 1 in the centralized setting), and computations that would have been performed by the central server are now outsourced to clients. However, there are some disadvantages to centralized federated learning in the adversarial setting. First, if the central server is itself an adversary, then it could easily prevent convergence of a centralized federated learning method. Second, if it is possible for the central server to be an adversary, clients might not be able to agree on an entity that should act as the central server.

Third, even if it is impossible for the central server to be an adversary, a centralized, similarity-based method requires two assumptions. The first assumption is that benign local gradients and adversarial local gradients are dissimilar in a way detectable to the central server. The first assumption is valid - if benign local gradients and adversarial local gradients are not dissimilar, then it would be difficult for adversaries to prevent global convergence. Therefore, it is generally possible for a central server to place clients into groups which are internally similar.

The second assumption is that having grouped clients using a similarity measure, it is possible for the central server to label the groups as benign or adversarial. The second assumption is necessary because the central server wants to place greater weight on local gradients it thinks belong to benign clients during aggregation. Unfortunately, the second assumption might not hold in the absence of additional information.

For example, Sniper requires that the adversarial local gradients do not form a maximum clique larger than that of the benign local gradients. This might not be the case if there are more adversaries than clients, or if adversaries purposefully broadcast local gradients with small Euclidean distances between each other. FLTrust addresses the second assumption by stipulating that the central server has its own local training dataset. Then, benign clients are likelier than adversaries to broadcast local gradients similar to the server's own. However, the central server might not have access to a local training dataset.

By contrast, in decentralized federated learning, clients perform aggregation rather than a central server. Since a benign client knows that itself is benign, it can label clients with local gradients similar to its own as benign, and place greater weight on those gradients during aggregation. As a result, it obtains a local model likely to be optimal from its own perspective.

Trusted Decentralized FL [GTB22] is a decentralized defense strategy against poisoning attacks. Two interesting things about Trusted Decentralized FL are the way that it computes "trust" and its aggregation scheme (more on the aggregation scheme in subsection 5.3). While Sniper and FLTrust use similarity measures (Euclidean distance and cosine similarity) as direct proxies for "trust", Trusted Decentralized FL computes trust through a comparatively sophisticated process. Let $\mathcal{M}_i$ be benign, and let $j \in \mathrm{ne}(i)$ index the neighbors of client i. In an iteration of Trusted Decentralized FL, client i first computes $I^{(i,j)} \in \{0, 1\}$, which indicates whether client j did not act suspiciously. The formula for $I^{(i,j)}$ is based on both a clustering-based method and a distance-based method. Denote by

$$\tau^{(i,j)} = \frac{r^{(i,j)} + 1}{r^{(i,j)} + s^{(i,j)} + 2} \in [0, 1]$$

the "local trust" client i has for client j, which is based only on similarity between the models of client i and client j. The parametrization of $\tau^{(i,j)}$ lends itself to a nice interpretation: $\tau^{(i,j)}$ is the expected value of a random variable

$$X^{(i,j)} \sim \mathrm{Beta}(r^{(i,j)} + 1, s^{(i,j)} + 1).$$

Client i increments $r^{(i,j)} \leftarrow \rho_1 r^{(i,j)} + I^{(i,j)}$ and $s^{(i,j)} \leftarrow \rho_2 s^{(i,j)} + 1 - I^{(i,j)}$ where $0 < \rho_1 < \rho_2$ are "forgetting factors". Denote by $t^{(i,j)}$ the "global trust" client i has for client j, which is based on the opinions the other neighbors of client i have for client j. Update

$$t^{(i,j)} = \frac{1}{\sum_{l \in \mathrm{ne}(i), l \neq j} \tau^{(i,l)}} \sum_{l \in \mathrm{ne}(i), l \neq j} \tau^{(i,l)} t^{(l,j)}.$$

That is, update global trust by a weighted average of neighbors' global trust, where the weights are local trust. Finally, client j broadcasts $(w^{(j)})' = \mathrm{ModelUpdate}(w^{(j)}, g^{(j,j)}, \alpha)$ to client i. The model $(w^{(j)})'$ is the local model the j-th client would obtain using only its local gradient $g^{(j,j)}$. The aggregation scheme is

$$g = g^{(i,i)} + \epsilon \frac{1}{\sum_{j \in \mathrm{ne}(i)} |D^{(j)}| t^{(i,j)}} \sum_{j \in \mathrm{ne}(i)} |D^{(j)}| t^{(i,j)} ((w^{(j)})' - (w^{(i)})').$$

3 Main Algorithm

3.1 Statement of Main Algorithm

The main algorithm presented here is a decentralized defense strategy against poisoning attacks. Let $\mathcal{M}_i$ be benign. First, client i computes $g^{(i,i)}$. Client i also wants to hear what its neighbors think about its model, so it broadcasts $(w^{(i)})^{(s)}$ to its neighbors and receives $g^{(i,j)}$, $j \in \mathrm{ne}(i)$ back. Client i will aggregate the local gradients $g^{(i,j)}$, $j \in \mathrm{ne}(i)$ during aggregation. A major component in determining the weight $g^{(i,j)}$ gets during aggregation is the cosine similarity measure between $g^{(i,i)}$ and $g^{(i,j)}$. This similarity measure is rescaled to fall in $[0,1]$ and denoted by a_{ij}:

$$a_{ij} \leftarrow \frac{1}{2}\left(\frac{\langle g^{(i,i)}, g^{(i,j)} \rangle}{\|g^{(i,i)}\|_2 \|g^{(i,j)}\|_2} + 1\right) \in [0,1]. \qquad \text{(cosine similarity step)}$$

The major assumption made here is that benign clients have a belief regarding the number of clients which are adversaries (for more on the validity of this assumption, see subsections 3.2 and 5.2). If clients believe that the number of adversaries is small, then they would want to inflate similarity measures to take greater advantage of the information propagated by their neighbors. Conversely, if benign clients believe that the number of adversaries is large, they would want to shrink similarity measures to reduce the influence of corrupted updates. The hyperparameter $\gamma \in \mathbb{R}$, called **trustworthiness**, represents the amount that benign clients trust the other clients collectively, and transforms the similarity measure:

$$a_{ij} \leftarrow a_{ij}^{\exp(-\gamma)} \in [0,1]. \qquad \text{(trustworthiness step)}$$

As Figure 1 shows, values of γ less than 0 shrink a_{ij}, and values of γ greater than 0 inflate a_{ij}:

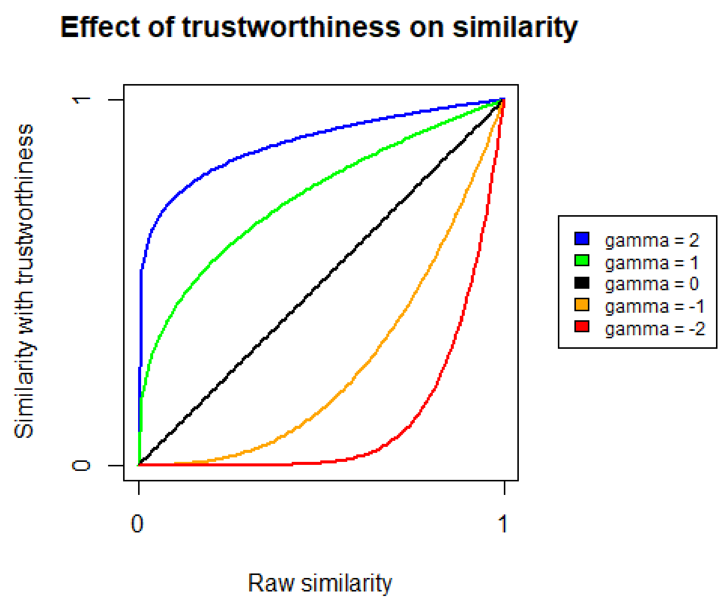

Figure 1: Similarity measure transformed by various values of γ

A common tactic for adversaries is to boost the magnitude of their update to have an outsized impact during aggregation. To protect against this, $g^{(i,j)}$ is shrunk to have magnitude at most $\|g^{(i,i)}\|_2$:

$$a_{ij} \leftarrow a_{ij} \min(1, \|g^{(i,i)}\|_2 / \|g^{(i,j)}\|_2) \in [0,1]. \qquad \text{(normalization step)}$$

Note that $a_{ii} = 1$, $i = 1, \ldots, K$. The aggregation scheme is simply

$$g^{(i)} \leftarrow \frac{1}{1 + \sum_{j \in \mathrm{ne}(i)} a_{ij}} \left(g^{(i,i)} + \sum_{j \in \mathrm{ne}(i)} a_{ij} g^{(i,j)} \right). \qquad \text{(aggregation step)}$$

The main algorithm:

Algorithm 1 Main Algorithm, Decentralized Version

```
INPUT: communication graph $G$, rule ModelUpdate, trustworthiness $\gamma \in \mathbb{R}$; $\mathcal{M}_k$ ($k = 1, \dots, K$) holds $C_k \subseteq \mathrm{ne}(k)$, local training dataset $D^{(k)}$, and learning rate $(\alpha^{(k)})^{(s)}$, $s \in \mathbb{N}$
INITIALIZE: $\mathcal{M}_k$ initializes $(w^{(k)})^{(0)}$, $k = 1, \dots, K$
REPEAT UNTIL CONVERGENCE:
 1: for $i = 1, \dots, K$ do
 2:     $\mathcal{M}_i$ sets $g^{(i,i)} \leftarrow \nabla_w f(D^{(i)}, w)|_{w=(w^{(i)})^{(s)}}$
 3:     for $j \in C_i \subseteq \mathrm{ne}(i)$ do
 4:         $\mathcal{M}_i$ broadcasts $(w^{(i)})^{(s)}$ to $\mathcal{M}_j$
 5:         $\mathcal{M}_j$ broadcasts $g^{(i,j)} \leftarrow \nabla_w f(D^{(j)}, w)|_{w=(w^{(i)})^{(s)}}$ to $\mathcal{M}_i$
 6:         $\mathcal{M}_i$ sets $a_{ij} \leftarrow \frac{1}{2}\left(\frac{\langle g^{(i,i)}, g^{(i,j)} \rangle}{\|g^{(i,i)}\|_2 \|g^{(i,j)}\|_2} + 1\right) \in [0,1]$
 7:         $\mathcal{M}_i$ sets $a_{ij} \leftarrow a_{ij}^{\exp(-\gamma)} \in [0,1]$
 8:         $\mathcal{M}_i$ sets $a_{ij} \leftarrow a_{ij} \min(1, \|g^{(i,i)}\|_2 / \|g^{(i,j)}\|_2) \in [0,1]$
 9:     end for
10:     $\mathcal{M}_i$ sets $g^{(i)} \leftarrow \frac{1}{1+\sum_{j \in \mathrm{ne}(i)} a_{ij}} \left(g^{(i,i)} + \sum_{j \in \mathrm{ne}(i)} a_{ij} g^{(i,j)}\right)$
11:     $\mathcal{M}_i$ sets $(w^{(i)})^{(s+1)} \leftarrow \mathrm{ModelUpdate}((w^{(i)})^{(s)}, g^{(i)}, (\alpha^{(i)})^{(s)})$
12: end for
RETURN $\hat{w}^{(k)}$, $k = 1, \dots, K$
```

If the communication graph is the complete graph with K edges (i.e., $\mathrm{ne}(i) = \{1, \dots, K\} \setminus \{i\}$, $i = 1, \dots, K$) and $C_k = \mathrm{ne}(k)$, $k = 1, \dots, K$, then Algorithm 1 is equivalent from an input-output perspective to Algorithm 2, given below. Algorithm 2 is centralized in the sense that it involves a central server, but clients still learn local models $w^{(k)}$ rather than a global model w.

Algorithm 2 Main Algorithm, Centralized Version

```
INPUT: trustworthiness $\gamma \in \mathbb{R}$, rule ModelUpdate; $\mathcal{M}_k$ holds local training dataset $D^{(k)}$ and learning rate $(\alpha^{(k)})^{(s)}$, $k = 1, \ldots, K$, $s \in \mathbb{N}$
INITIALIZE: $\mathcal{M}_k$ initializes $(w^{(k)})^{(0)}$, $k = 1, \ldots, K$
REPEAT UNTIL CONVERGENCE:
 1: for $i = 1, \ldots, K$ do
 2:     for $j = 1, \ldots, K$ do
 3:         $\mathcal{C}$ broadcasts $(w^{(i)})^{(s)}$ to $\mathcal{M}_j$
 4:         $\mathcal{M}_j$ broadcasts $g^{(i,j)} \leftarrow \nabla_w f(D^{(j)}, w)|_{w=(w^{(i)})^{(s)}}$ to $\mathcal{C}$
 5:     end for
 6: end for
 7: for $i = 1, \ldots, K$ do
 8:     for $j = 1, \ldots, K$ do
 9:         $\mathcal{C}$ sets $a_{ij} \leftarrow \frac{1}{2}\left(\frac{\langle g^{(i,i)}, g^{(i,j)}\rangle}{\|g^{(i,i)}\|_2 \|g^{(i,j)}\|_2} + 1\right) \in [0,1]$
10:         $\mathcal{C}$ sets $a_{ij} \leftarrow a_{ij}^{\exp(-\gamma)} \in [0,1]$
11:         $\mathcal{C}$ sets $a_{ij} \leftarrow a_{ij} \min(1, \|g^{(i,i)}\|_2 / \|g^{(i,j)}\|_2) \in [0,1]$
12:     end for
13:     $\mathcal{C}$ broadcasts $g^{(i)} \leftarrow \frac{1}{\sum_j a_{ij}} \sum_j a_{ij} g^{(i,j)}$ to $\mathcal{M}_i$
14:     $\mathcal{M}_i$ broadcasts $(w^{(i)})^{(s+1)} \leftarrow \mathrm{ModelUpdate}((w^{(i)})^{(s)}, g^{(i)}, (\alpha^{(i)})^{(s)})$ to $\mathcal{C}$
15: end for
RETURN $\hat{w}^{(k)}$, $k = 1, \ldots, K$
```

3.2 Multi-Stage Approach

Let client i be benign. After a run-through of the main algorithm, client i holds the similarity measures a_{ij}, $j \in \mathrm{ne}(i)$ in addition to $\hat{w}^{(i)}$. From the perspective of client i, clients j with relatively large a_{ij} are likelier to be benign, and clients j with relatively small a_{ij} are likelier to be adversaries. This motivates an approach in which the main algorithm is repeated, and at each subsequent stage client i restricts communication to the neighbors j from the previous stage with large a_{ij}. Since γ represents the amount that a benign client trusts other clients, γ is incremented between stages.

1. Fix $\gamma \in \mathbb{R}$ and $(w^{(k)})^{(0)}$. Run through the decentralized main algorithm with $C_k = \mathrm{ne}(k)$, $k = 1, \ldots, K$. When the algorithm terminates, each

client i records raw cosine similarity

$$\frac{1}{2}\left(\frac{\langle g^{(i,i)}, g^{(i,j)}\rangle}{\|g^{(i,i)}\|_2\|g^{(i,j)}\|_2}+1\right), \quad j \in \{i\} \cup \mathrm{ne}(i).$$

2. Run through the decentralized main algorithm, again starting from $(w^{(k)})^{(0)}$, incrementing $\gamma \leftarrow \gamma + 1$, and restricting

$$C_i = \left\{ j \in \mathrm{ne}(i) \colon \frac{1}{2}\left(\frac{\langle g^{(i,i)}, g^{(i,j)}\rangle}{\|g^{(i,i)}\|_2\|g^{(i,j)}\|_2}+1\right) > \tau \right\}$$

for some threshold $\tau \in (0, 1)$.

Note that raw cosine similarity is used during thresholding rather than a_{ij} after trustworthiness or normalization steps. While the purpose of cosine similarity is to quantify the similarity between clients' updates, the purpose of trustworthiness and normalization is merely to make aggregation robust. Therefore, a_{ij} prior to the application of trustworthiness and normalization best captures the chance that client j is benign or adversarial from the perspective of client i.

4 Experiments on Synthetic Data

4.1 Adversarial Sparse Linear Regression

The main algorithm is tested on the task of **adversarial sparse linear regression**, in which

- The k-th client holds $D^{(k)} = (X^{(k)}, Y^{(k)})$, where $X^{(k)} \in \mathbb{R}^{n^{(k)} \times p}$ and $Y^{(k)} \in \mathbb{R}^{n^{(k)}}$, $k = 1, \ldots, K$.
- The joint training dataset is

$$X = \begin{bmatrix} X^{(1)} \\ \vdots \\ X^{(K)} \end{bmatrix} \in \mathbb{R}^{n \times p}$$

$$Y = \begin{bmatrix} Y^{(1)} \\ \vdots \\ Y^{(K)} \end{bmatrix} \in \mathbb{R}^{n}$$

where $n = n^{(1)} + \cdots + n^{(k)}$.

- If client k is an adversary, it randomly selects coordinates $i \in \{1, \dots, n^{(k)}\}$ at which to corrupt $Y^{(k)} \in \mathbb{R}^{n^{(k)}}$. At these coordinates, client k puts $[Y^{(k)}]_i \leftarrow -[Y^{(k)}]_i$. Client k uses the corrupted $Y^{(k)}$ for the computation of local gradients. The number of indices at which $Y^{(k)}$ is corrupted will be prespecified as an experimental variable.

- The empirical loss is
$$f(X, Y, \beta) = \frac{1}{n}\|Y - X\beta\|_2^2 + \lambda\|\beta\|_1.$$

- The k-th local empirical loss is
$$f(X^{(k)}, Y^{(k)}, \beta) = \frac{1}{n^{(k)}}\|Y^{(k)} - X\beta\|_2^2 + \lambda\|\beta\|_1.$$

In sparse linear regression, two common inferential targets are β^* and the support of β^*. The former is measured by relative norm $\|\beta^* - \hat{\beta}\|_2/\|\beta^*\|_2$. The latter is measured by F1 score. Ordinarily, F1 score is computed by comparing the support of $\hat{\beta}$ to the support of β^* (e.g., a true positive occurs precisely when a component of $\hat{\beta}$ and its corresponding component of β^* are both nonzero). However, since adversaries corrupt Y by flipping the sign of Y_i for some coordinates i, it was common in the case of high corruption for benign clients to obtain $\hat{\beta}$ with the correct support, but every nonzero entry has the incorrect sign. Therefore, F1 score is computed by comparing the positive entries of $\hat{\beta}$ to the positive entries of β^* (e.g., a true positive occurs precisely when a component of $\hat{\beta}$ and its corresponding component of β^* are both greater than zero).

4.2 Data Generation

A total of 3 experiments were conducted, each repeated over 10 runs. All results in subsection 4.5 are reported as averages over 10 runs. The parameters $K = 10$, $n^{(k)} = 50$, $k = 1, \dots, 10$, and $p = 100$ were fixed across all experiments. At each new experimental run, a new dataset was generated in the following way. The optimal model $\beta^* \in \mathbb{R}^{100}$ is generated as a vector of 95 zeros and 5 ones, the positions of the ones drawn uniformly from $\{1, \dots, 100\}$. I.e., the sparsity of β^* is 0.05. Each row of $X^{(k)} \in \mathbb{R}^{50\times 100}$ is drawn i.i.d. from a multivariate normal
$$X^{(k)} \sim N_{100}(0_{100}, I_{100\times 100}).$$

Each component of the error $\epsilon \in \mathbb{R}^{100}$ is drawn i.i.d. from a univariate normal

$$\epsilon \sim N(0, \|\beta^*\|_2^2).$$

The variance is chosen to be $\|\beta^*\|_2^2$ so that the signal to noise ratio equals 1. Finally, $Y^{(k)} = X^{(k)}\beta^* + \epsilon$.

4.3 Federated Proximal Gradient Descent

The approach to minimizing the empirical loss is to adapt proximal gradient descent to the federated setting:

- The local empirical loss is the sum of a differentiable part and a non-differentiable part. The gradient of the differentiable part is

  $$\frac{1}{n^{(k)}}(X^{(k)})^\top(X^{(k)}\beta - Y^{(k)}),$$

  and this formula is used in the computation of local gradients.

- The model update is given by

  $$\text{ModelUpdate}(\beta, g, \alpha) = S_{\lambda\alpha}(\beta - \alpha g)$$

  where $S_{\lambda\alpha}$ is the **soft thresholding operator** defined componentwise by

  $$[S_{\lambda\alpha}(\beta)]_i = \begin{cases} \beta_i - \lambda\alpha, & \beta_i > \lambda\alpha \\ 0, & |\beta_i| \leq \lambda\alpha \\ \beta_i + \lambda\alpha, & \beta_i < -\lambda\alpha. \end{cases}$$

- At the s-th iteration, the learning rate is given by

  $$\arg\min_{(\alpha)^{(s)} \in A} \sum_{k=1}^{K} f(X^{(k)}, Y^{(k)}, (\beta)^{(s+1)})$$

  for the centralized approaches and

  $$\arg\min_{(\alpha^{(k)})^{(s)} \in A} f(X^{(k)}, Y^{(k)}, (\beta^{(k)})^{(s+1)})$$

  for the decentralized approaches, where $A = \{10^1, 10^{0.5}, 10^0, 10^{-0.5}, 10^{-1}\}$.

4.4 Experimental Setup

In each experiment, if there are K' adversaries, then the first $10 - K'$ clients are benign and the last K' clients are adversaries. In particular, the first client is always benign (K' is always less than 10).

In experiments 1 and 2, the following algorithms are pitted against each other:

- FedAvg;
- median;
- trimmed mean, trimming the 3 largest entries and 3 smallest entries;
- Krum, including the 5 closest local gradients in the computation of score;
- The main algorithm with a complete communication graph, each client communicating with all its neighbors at every iteration, and trustworthiness $\gamma = -0.5$;
- A second stage following the first run-through of the main algorithm, where a client j is included if $\frac{1}{2}\left(\frac{\langle g^{(1,1)}, g^{(1,j)}\rangle}{\|g^{(1,1)}\|_2 \|g^{(1,j)}\|_2} + 1\right) > 0.55$ (i.e., a client is included if client 1 trusts it). Trustworthiness is incremented from $\gamma = -0.5$ to $\gamma = 0.5$.

Also, results for LASSO on $(X^{(1)}, Y^{(1)})$ are recorded. This is done to illustrate that a single client does not possess a large enough local dataset to recover β^*, necessitating federated learning.

Each time an algorithm was carried out, the regularization parameter λ was selected via grid search over $\{10^{-2.5}, 10^{-2}, 10^{-1.5}, 10^{-1}, 10^{-0.5}, 10^{0}, 10^{0.5}, 10^{1}\}$. The parameter λ was selected to maximize F1 score, and minimize relative norm if F1 scores are tied.

Each method was carried out for 100 iterations.

4.5 Results

Two experimental variables are considered: the number of adversaries, and the percent of coordinates that each adversary corrupts. In experiment 1, the number of adversaries is varied from 0 to 9 and the percent of corrupted indices is fixed at 75%. For LASSO, FedAvg, median, trimmed mean, and Krum, F1 score and relative norm are based on the global $\hat{\beta}$ and β^*. For the main algorithm and the 2-stage main algorithm, F1 score and relative norm are based on $\hat{\beta}^{(1)}$ and β^*.

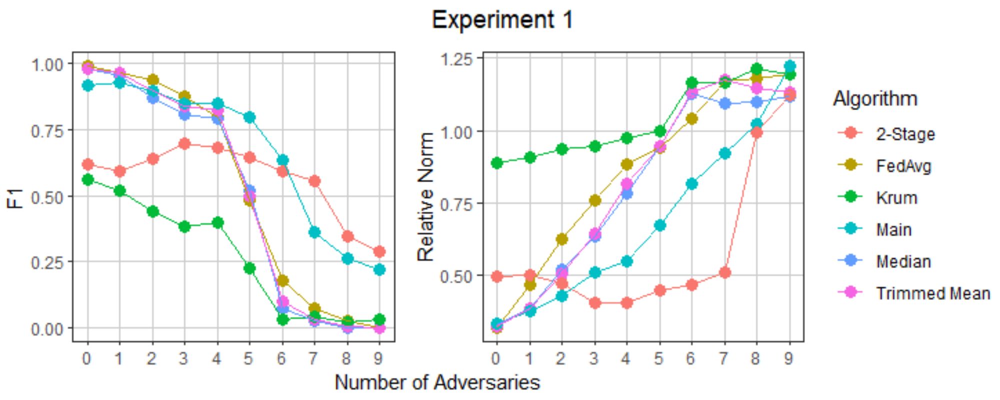

Figure 2: Experiment 1

In terms of F1 score, the performance of FedAvg, median, and trimmed mean steeply drops off with a fifth adversary. The main algorithm performs comparably to FedAvg, median, and trimmed mean for 0 to 4 adversaries and outperforms them for 5 to 9 adversaries. The 2-stage main algorithm performs worse than FedAvg, median, and trimmed mean for 0 to 4 adversaries and better for 5 to 9 adversaries. Intuitively, with a large enough number of adversaries, the aggregation methods of median and trimmed mean will fail to exclude corrupted local gradients. Therefore, the main algorithm should outperform median and trimmed mean with a sufficient number of adversaries.

In experiment 2, the number of adversaries is fixed at 6 and the percent of coordinates corrupted varies in $\{0\%, 10\%, 20\%, \ldots, 90\%, 100\%\}$. Again, for LASSO, FedAvg, median, trimmed mean, and Krum, F1 score and relative norm are based on the global $\hat{\beta}$ and β^*. This time, for the main algorithm, F1 score is based on the combined confusion matrix of clients 1 through 4 (the benign clients), and relative norm is reported as the average of the relative norms of clients 1 through 4. For the 2-stage main algorithm, F1 score is based on the combined confusion matrix of the benign clients present for the second stage (clients 2, 3, and 4 are not necessarily part of the second stage) and relative norm is reported as the average relative norm of the benign clients present for the second stage.

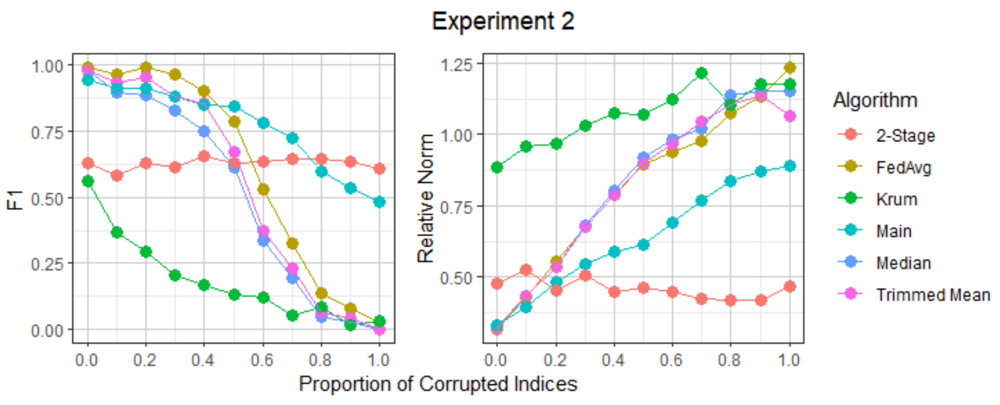

Figure 3: Experiment 2

The main algorithm performs comparably to FedAvg, median, and trimmed mean for 0% to 40% corruption added and outperforms them for 50% to 100% corruption added. The 2-stage main algorithm performs worse than FedAvg, median, and trimmed mean for 0% to 50% corruption added and better for 60% to 100% corruption added. Intuitively, having fixed 6 adversaries, it should be difficult for median or trimmed mean to exclude corrupted local gradients from aggregation. However, the effect of corrupted local gradients is not pronounced unless the percent corruption added is sufficiently high, at which point the main algorithm will outperform median and trimmed mean.

For reference, in experiments 1 and 2, LASSO on $(X^{(1)}, Y^{(1)})$ achieves an F1 score of 0.59 and relative norm of 0.90, averaged over the 10 runs. In terms of F1 score, FedAvg performs better than LASSO with as many as 4 adversaries or 50% corruption added, and the main algorithm performs better with as many as 6 adversaries or 80% corruption added, demonstrating the need for federated learning.

The setup for experiment 3 is the same as experiment 1 (varying the number of adversaries at 75% corruption added), but this time the main algorithm is compared with itself for multiple values of trustworthiness. This experiment is meant to verify the assertion that the trustworthiness parameter can be used to leverage knowledge about the number of adversaries.

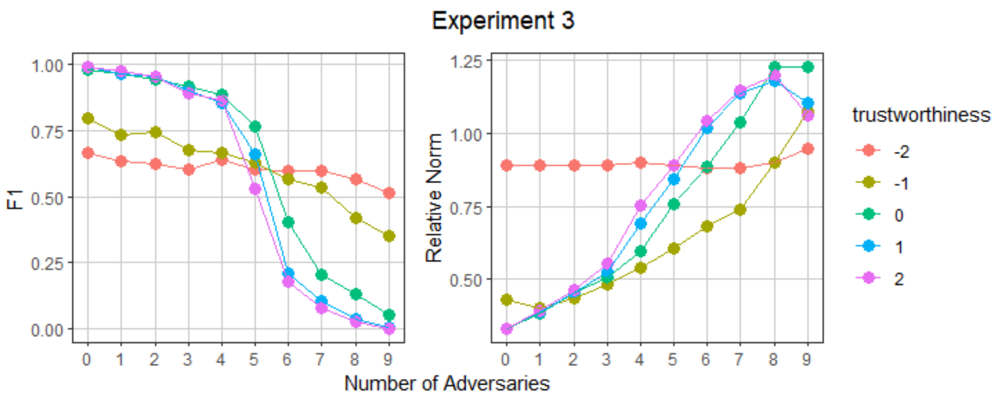

Figure 4: Experiment 3

In terms of F1 score, $\gamma = 0, 1, 2$ perform best with 0 to 4 adversaries, all 5 methods perform comparably with 5 adversaries, and $\gamma = -1, -2$ perform best with 6 to 9 adversaries. While $\gamma = 0, 1, 2$ steeply drop off between the low and high adversary settings, the decline in performance for $\gamma = -1$ and -2 is gradual. Intuitively, when benign clients are the majority, it is a good strategy to inflate similarity measures, and when adversaries are the majority, it is a good strategy to shrink similarity measures. Because the methods with high trustworthiness give weight to all local gradients, their performance depends greatly on the number of adversaries. Since the methods with low trustworthiness downweigh dissimilar local gradients, their performance is inelastic with respect to the number of adversaries.

5 Discussion

There are many further improvements and future directions for this research project. Three of the most interesting improvements and directions are listed below.

5.1 Adversaries

The poisoning attacks considered here can be categorized as **data poisoning attacks** since adversaries poison their local training data. Another kind of poisoning attack is **local model poisoning attacks**, in which adversaries directly poison the local gradient that they send to the server or to their neighbor. One framework for formulating a local model poisoning attack is given in [FCJ20], in which adversaries intend to maximally deviate the global model from the direction it would head without attacks. To do so, adversaries send local gradients which solve a constrained optimization problem based on the current global model, current local models of all clients, and the aggregation rule used by the central server. With this framework, [FCJ20] derive **Krum attack**, **trimmed mean attack**, and **median attack**. In [CFL20], this same framework is used to tailor a local model poisoning attack to FLTrust. It is worthwhile to investigate the robustness of the main algorithm to Krum attack, trimmed mean attack, and median attack, and also to use the framework of [FCJ20] to design an attack optimal with respect to the main algorithm.

5.2 Trustworthiness

One of the main ideas behind the algorithm is to base the coefficient of $g^{(i,j)}$ on some similarity measure in $[0, 1]$, and then to either inflate or shrink that coefficient based on trustworthiness. Trustworthiness as currently formulated can be improved in at least two ways.

First, trustworthiness is currently a global parameter. This fails to represent the highly plausible scenario in which client i has unequal amounts of trust for two of its neighbors. Therefore, each client i should track trustworthiness locally for each of its neighbors j. Then, trustworthiness can be denoted $\gamma^{(i,j)}$.

Second, trustworthiness currently remains static for the duration of the main algorithm. This fails to represent the highly plausible scenario in which the reputation of client j in the eyes of client i changes over the duration of the main algorithm. The multi-stage approach is an initial attempt to address this issue, but suffers from multiple flaws. For one, requiring an entire run-through of the main algorithm to complete before updating trustworthiness is slow. For two, if adversaries know when a round will end, they can broadcast uncorrupted gradients at the final iteration to ensure they will be included in the next round.

Therefore, it makes sense to dispense with the multistage approach in favor of an approach where trustworthiness is updated at every iteration, as in Trusted Decentralized FL. Adaptive trustworthiness should depend on the history of similarity scores, but the main idea of basing the aggregation coefficient on similarity at the current iteration and inflating it or shrinking it by trustworthiness should remain.

One idea for adaptive trustworthiness which draws some inspiration from Trusted Decentralized FL is the following. First, let trustworthiness take values in $(0, 1)$ instead of $\mathbb{R}$. Noting that the function from $(0, 1)$ to $(0, \infty)$ defined by $x \mapsto -1+1/x$ is bijective and decreasing, the trustworthiness step is rewritten

$$a_{ij} \leftarrow a_{ij}^{-1+1/\gamma^{(i,j)}}. \tag{trustworthiness step}$$

Parameterize trustworthiness by

$$(\gamma^{(i,j)})^{(s)} = \frac{(A_{ij})^{(s)}}{(A_{ij})^{(s)} + (B_{ij})^{(s)}}.$$

Every iteration, update

$$\begin{aligned}(A_{ij})^{(s+1)} &= (A_{ij})^{(s)} + ReLU\left(\frac{\langle g^{(i,i)}, g^{(i,j)}\rangle}{\|g^{(i,i)}\|_2\|g^{(i,j)}\|_2}\right)\\ (B_{ij})^{(s+1)} &= (B_{ij})^{(s)} + ReLU\left(-\frac{\langle g^{(i,i)}, g^{(i,j)}\rangle}{\|g^{(i,i)}\|_2\|g^{(i,j)}\|_2}\right).\end{aligned}$$

For an intuitive explanation, denote by θ the angle between $g^{(i,i)}$ and $g^{(i,j)}$. At every iteration, if θ is less than 90 degrees, increment A_{ij} by $\cos(\theta)$ (which increases $\gamma^{(i,j)}$ by an amount which scales with the acuteness of θ) and if θ is greater than 90 degrees, increment B_{ij} by $-\cos(\theta)$ (which decreases $\gamma^{(i,j)}$ by an amount which scales with the obtuseness of θ).

5.3 Decentralized Aggregation

During experimentation, only a complete communication graph was considered. It would be worthwhile to investigate the performance of the main algorithm on communication graphs that are not complete. Actually, the current aggregation scheme might not be totally adequate for a non-complete communication graph.

Consider decentralized federated learning in the non-adversarial setting. As before, let $(w^{(j)})' = \text{ModelUpdate}(w^{(j)}, g^{(j,j)}, \alpha)$. While the aggregation scheme of Trusted Decentralized FL takes the form

$$g^{(i)} = \text{Aggregate}(g^{(i,i)}, \{(w^{(j)})' - (w^{(i)})'\}_{j\in\text{ne}(i)}), \tag{1}$$

the aggregation scheme of the main algorithm takes the form

$$g^{(i)} = \text{Aggregate}(\{g^{(i,j)}\}_{j \in \{i\} \cup \text{ne}(i)}). \tag{2}$$

Because of the dependence of equation 1 on $(w^{(j)})'$, equation 1 can be called **model-based aggregation**. Since equation 2 requires clients to compute more gradients, equation 2 can be called **gradient-based aggregation**.

In gradient-based aggregation, client i considers $g^{(i,j)}$, $j \in \{i\} \cup \text{ne}(i)$ in aggregation. Each $g^{(i,j)}$ is the gradient of the j-th local empirical loss evaluated at the i-th current local model. Thus, gradient-based aggregation yields an update which is highly personalized to i-th current local model.

In model-based aggregation, client i compares $(w^{(j)})'$, $j \in \text{ne}(i)$ to $(w^{(i)})'$ during aggregation. The advantage of model-based aggregation pertains to the propagation of information over a communication graph which is not complete. For the simplest possible case, consider the following communication graph:

$$\begin{tikzcd} \textcircled{1} \arrow[r, dash] & \textcircled{2} \arrow[r, dash] & \textcircled{3} \end{tikzcd} .$$

At iteration s, client 2 aggregates

$$g^{(2)} = \text{Aggregate}(g^{(2,2)}, (w^{(1)})' - (w^{(2)})', (w^{(3)})' - (w^{(2)})').$$

In iteration s, information from client 3 propagates to the new local model of client 2 (via $(w^{(3)})'$ during aggregation).

At iteration $s+1$, client 1 aggregates

$$g^{(2)} = \text{Aggregate}(g^{(1,1)}, (w^{(2)})' - (w^{(1)})').$$

In iteration $s+1$, information from client 2 propagates to the new local model of client 1 (via $(w^{(2)})'$ during aggregation). In iteration $s+1$, information from client 3 also propagates to the new local model of client 1, despite the fact that client 1 cannot communicate with client 3. This is because $(w^{(2)})'$ contains information propagated to client 2 from client 3 in iteration s.

By contrast, with gradient-based aggregation, there is no mechanism by which information propagates from client 3 to client 1. Client 1 receives only the gradient of the 2nd local empirical loss evaluated at the 1st current local model, which does not depend in any way on client 3.

One further direction is to formulate an aggregation scheme

$$g^{(i)} = \text{Aggregate}(\{(w^{(j)})' - (w^{(i)})'\}_{j \in \text{ne}(i)}, \{g^{(i,j)}\}_{j \in \{i\} \cup \text{ne}(i)}).$$

This aggregation scheme can be a combination of a "personalized" component depending on $g^{(i,j)}$, $j \in \{i\} \cup \mathrm{ne}(i)$ as well as a "global" component depending on $(w^{(j)})' - (w^{(i)})'$, $j \in \mathrm{ne}(i)$. By appropriately weighting these components, a balance can be struck between personalized updates based on the current local model of client i, and global updates that incorporate information propagated along the entirety of the communication graph.

References

[BEG17] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. “Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent.” In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[CCL19] Di Cao, Shan Chang, Zhijian Lin, Guohua Liu, and Donghong Sun. “Understanding Distributed Poisoning Attack in Federated Learning.” In *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 233–239, 2019.

[CFL20] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. “FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping.” *CoRR*, **abs/2012.13995**, 2020.

[FCJ20] 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)*, pp. 1605–1622. USENIX Association, August 2020.

[GTB22] Anousheh Gholami, Nariman Torkzaban, and John S. Baras. “Trusted Decentralized Federated Learning.” In *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–6, 2022.

[LLX21] Xiaoyuan Liu, Hongwei Li, Guowen Xu, Zongqi Chen, Xiaoming Huang, and Rongxing Lu. “Privacy-Enhanced Federated Learning Against Poisoning Adversaries.” *IEEE Transactions on Information Forensics and Security*, **16**:4574–4588, 2021.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. “The Byzantine generals problem.”, Jul 1982.

[LWW21] Dongcheng Li, W. Eric Wong, Wei Wang, Yao Yao, and Matthew Chau. “Detection and Mitigation of Label-Flipping Attacks in Federated Learning Systems with KPCA and K-Means.” In *2021 8th International Conference on Dependable Systems and Their Applications (DSA)*, pp. 551–559, 2021.

[MMR16] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. "Federated Learning of Deep Networks using Model Averaging." *CoRR*, **abs/1602.05629**, 2016.

[YCR18] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter L. Bartlett. "Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates." *CoRR*, **abs/1803.01498**, 2018.