PRIVACY ANALYSIS OF DECENTRALIZED FEDERATED LEARNING

PRIVACY ANALYSIS OF DECENTRALIZED FEDERATED LEARNING

Thesis

by

Wenrui YU

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended on Monday 17 July, 2023 at 13:00.

Student number:	5495040	
Faculty:	EEMCS	
Master program:	Electrical Engineering	
Specialization:	Signals and Systems	
Project Duration:	August, 2022 - July, 2023	
Thesis committee:	Prof. dr. ir. R. Heusdens,	TU Delft, supervisor
	Dr. K. Liang,	TU Delft
	Dr. Q. Li,	Tsinghua University

ABSTRACT

Privacy concerns in federated learning have attracted considerable attention recently. In centralized networks, it has been observed that even without directly exchanging raw training data, the exchange of other so-called intermediate parameters such as weights/gradients can still potentially reveal private information. However, there has been relatively less research conducted on privacy concerns in decentralized networks.

In this report, we analyze privacy leakage in optimization-based decentralized federated learning, which adopts generally distributed optimization schemes such as ADMM or PDMM in federated learning. By combining local updates with global aggregations, it was proved that optimization-based approaches are more advantageous compared to the traditional average consensus-based approaches, especially in scenarios where the data at the nodes are not independent and identically distributed (non-IID).

We further extend the privacy bound in distributed optimization to the decentralized learning framework. Different from the fact in the centralized learning framework the leaked information is the local gradients of each individual participant at all rounds, we find that in decentralized cases the leaked information is the difference of the local gradients within a certain time interval. Motivated by the gradient inversion in centralized networks, we then design a homogeneous attack to iteratively optimize dummy data whose gradient differences are close to the true revealed gradient differences. Though the gradient difference information still brings privacy concerns, we show that it is more challenging for adversaries to reconstruct private data using the difference of gradients than using the gradients themselves in the centralized case.

To deal with the privacy attack, we propose several potential defense strategies such as early stopping, inexact update and quantization etc. The main advantage of these approaches is that they introduce error/noise/distortion into decentralized federated learning for protecting private information from being revealed to others without affecting the training accuracy. In addition, we also show that the larger the batchsize is, the more difficult for the adversary to reconstruct the private information.

ACKNOWLEDGEMENTS

Time flies in TU Delft. I sincerely thank my supervisors, Richard Heusdens and Qiongxiu Li for their exhaustive guidance and assistance in finishing this thesis. Also, I would like to thank my parents and friends for their financial and mental support.

Wenrui Yu
Delft, June 2023

CONTENTS

LIST OF FIGURES

1

INTRODUCTION

With the rapid advancement of technology, the world is becoming increasingly interconnected. The advent of the Big Data era means better user behavior prediction and service improvement. But inevitably, people's private data are leaked and utilized intentionally or unintentionally. Consequently, how to protect privacy has become an important issue in this era.

Federated learning (FL) is a subject born out of this era. Its purpose is to enable collaborative training of multiple devices without the direct exchange of data [1]. For example, there is a limit to the number of cases available for a hospital, which makes it difficult to assist in diagnosis by means of data analysis. Therefore, hospitals from different areas prefer to seek collaboration with each other. However, patient privacy is supposed to be strictly confidential and should not be shared. This raises a concern. How can the information be utilized without sharing privacy?

Federated Averaging (FedAvg) [1] is the first federated learning algorithm that was proposed along with the concept of FL. It is suitable for such scenarios: there is one server that connects to all other clients/nodes, as shown in Figure 1.1. Data are distributed on all the nodes and strictly kept locally, so only other information can be exchanged between the server and nodes. We expect to train a global model which integrates information from all nodes. In such a setting, the main procedure of FedAvg can be summarized into three steps: 1) nodes first train local models based on their own private dataset and send the local model updates, i.e. gradients or weights, to the server; 2) with the received messages, the server do the weighted averaging aggregation to determine a global model and returns it to nodes; 3) nodes update the local models based on the updated global model and send the model updates back to the server, after which step 2 and 3 are repeated until convergence.

This type of approach, based on alternatively training local models and aggregating them with a server, has become the mainstream of federated learning. We call the corresponding topology with a central server as the centralized topology. This communication mode requires high communication bandwidth and is vulnerable. The server not only has to take high communication costs but also must be trusted by all nodes. Also, the

1

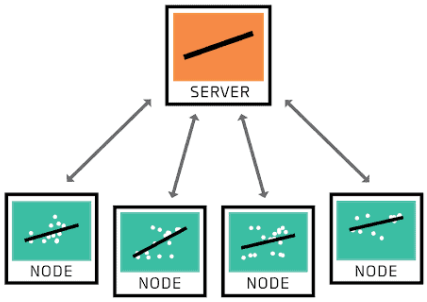

Figure 1.1: Federated Averaging [2]

entire network would go down as long as the server failed. Therefore, another topology, what we call decentralized topology, has recently gained attention. It removes the servers and constructs a connected network through direct communication channels between nodes.

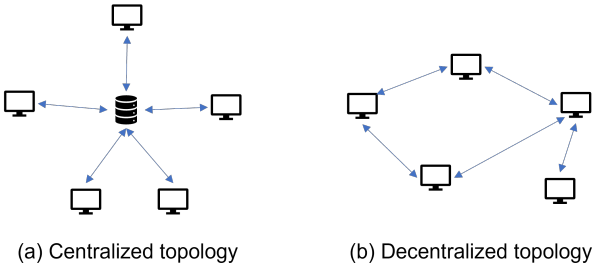

Figure 1.2: Two topologies in federated learning

Decentralized FL protocols fall into two main categories. The first one is based on average-consensus protocols. Similar to the training process in the centralized topology, nodes exchange data after training the model locally. However, the data exchange is limited to the surrounding neighbors. Examples where the data aggregation is done using average consensus techniques such as gossiping SGD [3], D-PSGD [4] and variations thereof [5], [6]. The second category contains protocols that are based on distributed optimization. These methods formulate the underlying problem as a constrained optimization problem and solve the optimization problem using distributed solvers like ADMM [7]–[9] or PDMM [10]–[12]. The constraints are formulated in such a way that, after convergence, the learned models at all nodes are identical. Hence, there is no explicit separation between updating local models and updating the global model.

Despite the fact that FL avoids direct data exchange, it does not necessarily imply that users can rest easily without any concerns. [13] first pointed out, adversaries can inverse private data from deep leaked gradients. Subsequently, a series of attacks [14]–[22] which attempt to reconstruct private data based on leaked gradients or weights have been continuously proposed and improved.

Most of the existing works on privacy leakage can apply to centralized topologies, considering the information leaked during transmission between nodes and the server. In addition, recently in [23], it is shown that average-consensus based decentralized FL protocols do not offer privacy advantages over centralized FL, as the traditional gradient inversion attack can still be applied to these protocols. And for those malicious nodes, they may have the chance to infer sensitive information more due to the different local generalizations. The privacy loss of optimization-based decentralized FL protocols, on the other hand, has, to the best of our knowledge, been rarely investigated.

The main contribution of this work is the extension of the upper bound of privacy leakage in distributed optimization to decentralized FL. The theory reveals that such privacy bound is essentially the leakage of gradient differences in a successive period. Therefore, a series of homogeneous gradient-based attacks still apply to optimized-based FL. However, in this case, carrying out attacks requires adversaries to possess more information and have stronger computational and storage resources compared to the scenario of gradient leakage. We also illustrate specific issues that exist under the general framework of decentralized FL, such as inexact updates. We find that in fact, some unavoidable perturbations in the training are helpful for preserving privacy. Thus we also propose some corresponding feasible defense strategies.

The report is structured as the following

- In Chapter 2 the preliminaries, including ADMM/PDMM framework and related privacy theory is introduced.
- In Chapter 3 the setup of subsequent experiments is given.
- In Chapter 4 two ways of inexact update of PDMM and related privacy bound are mentioned.
- In Chapter 5 the corresponding attack in decentralized FL and the comparison to the centralized FL is analyzed.
- In Chapter 6 possible perturbations and defense strategies are proposed.
- In Chapter 7 we give conclusions of this report and also discuss directions of the future works.

2

PRELIMINARIES

In this chapter, background information related to decentralized FL will be provided. Firstly in Section 2.1, decentralized learning is formulated as a general distributed optimization problem. Then in Section 2.2, the implementation of distributed optimization with the PDMM algorithm is given. Finally, the related privacy preservation approach and privacy leakage derivation are presented in Section 2.3.

2.1. PROBLEM STATEMENT

Let us define the decentralized topology with graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices which represents the nodes and $\mathcal{E}$ is the set of undirected edges which represents the connections between nodes. We use $\mathcal{N}_i = \{j \in \mathcal{V} \mid (i,j) \in \mathcal{E}\}$ to denote the neighbors of i-th node.

With the connected network, We then consider the summation of the local cost functions as the objective of collaborative learning. Each local cost function $f_i(\boldsymbol{w}_i, (\boldsymbol{x}_i, \boldsymbol{\ell}_i))$ represents the sum of costs associated with individual samples, resulting in the equal weighting of all samples. $\{(\boldsymbol{x}_i, \boldsymbol{\ell}_i) : i \in \mathcal{V}\}$ is the local dataset and $\boldsymbol{w}_i \in \mathbb{R}^u$ is the model weights on node i. Here we use $f_i(\boldsymbol{w}_i)$ as the simplified written of $f_i(\boldsymbol{w}_i, (\boldsymbol{x}_i, \boldsymbol{\ell}_i))$. In such a setting, the objective is equivalent to the case of placing all data on a single node. And the optimization problem can be posed with the constraints to guarantee $\forall (i,j) \in \mathcal{E} : \boldsymbol{w}_i = \boldsymbol{w}_j$ as

$$\begin{aligned} \min_{\{\boldsymbol{w}_i : i \in \mathcal{V}\}} \quad & \sum_{i \in \mathcal{V}} f_i(\boldsymbol{w}_i), \\ \text{subject to} \quad & \forall (i,j) \in \mathcal{E} : \boldsymbol{B}_{i|j} \boldsymbol{w}_i + \boldsymbol{B}_{j|i} \boldsymbol{w}_j = \boldsymbol{0}, \end{aligned} \tag{2.1}$$

where $\boldsymbol{B}_{i|j} \in \mathbb{R}^{u \times u}$ is defined as the identity matrix $\boldsymbol{I}_u \in \mathbb{R}^{u \times u}$ with opposite signs as the following

$$\forall \left(i,j\right) \in \mathcal{E} : \ \boldsymbol{B}_{i|j} = \begin{cases} \boldsymbol{I}_u, & \text{if } i < j, \\ -\boldsymbol{I}_u, & \text{if } i > j. \end{cases} \tag{2.2}$$

2.2. ADMM/PDMM

We consider employing the ADMM/PDMM framework as an optimization-based approach for decentralized federated learning. From the monotone operator theory [24], [25], ADMM [26] is the $\frac{1}{2}$-averaged version of PDMM [10], [11] and thus can be expressed in one framework. For ADMM, it provides convergence guarantees for arbitrary convex, closed and proper (CCP) objective functions; while PDMM requires strong convexity.

Let $E = |\mathcal{E}|$ as the number of edges and $N = |\mathcal{V}|$ as the number of nodes in the graph. Define $\boldsymbol{C} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n] \in \mathbb{R}^{2uE \times uN}$, where $\boldsymbol{c}_i(l) = \boldsymbol{B}_{i|j}$ and $\boldsymbol{c}_j(l + uE) = \boldsymbol{B}_{j|i}$. From [25], by introducing auxiliary edge variables $\boldsymbol{z}$, the optimization problem with consensus constraints shown in Equation 2.1 can be solved by iteratively updating the following equation:

$$\boldsymbol{w}^{(t+1)} = \arg\min_{\boldsymbol{w}} \left(f(\boldsymbol{w}) + \boldsymbol{w}^\top \boldsymbol{C}^\top \boldsymbol{z}^{(t)} + \frac{\rho}{2} \|\boldsymbol{C}\boldsymbol{w}\|^2 \right), \tag{2.3}$$

$$\boldsymbol{y}^{(t+1)} = \boldsymbol{z}^{(t)} + 2\rho \boldsymbol{C}\boldsymbol{w}^{(t+1)}, \tag{2.4}$$

$$\boldsymbol{z}^{(t+1)} = (1-\theta)\boldsymbol{z}^{(t)} + \theta \boldsymbol{P}\boldsymbol{y}^{(t+1)}. \tag{2.5}$$

where $\boldsymbol{P} \in 2uE \times 2uE$ is the permutation matrix and ρ is a constant controlling the rate of convergence. $\theta \in (0, 1]$ controls the operator averaging, while $\theta = \frac{1}{2}$ is the case of Peaceman-Rachford splitting (ADMM) and $\theta = 1$ is Douglas-Rachford splitting (PDMM).

The corresponding individual update is written as [27]

$$\boldsymbol{w}_i^{(t+1)} = \arg\min_{\boldsymbol{w}_i} \left(f_i(\boldsymbol{w}_i) + \sum_{j \in \mathcal{N}_i} \boldsymbol{z}_{i|j}^{(t)\top} \boldsymbol{B}_{i|j} \boldsymbol{w}_i + \frac{\rho d_i}{2} \boldsymbol{w}_i^2 \right), \tag{2.6}$$

$$\forall j \in \mathcal{N}_i : \boldsymbol{z}_{j|i}^{(t+1)} = (1-\theta)\boldsymbol{z}_{j|i}^{(t)} + \theta \left(\boldsymbol{z}_{i|j}^{(t)} + 2\rho \boldsymbol{B}_{i|j} \boldsymbol{w}_i^{(t+1)} \right), \tag{2.7}$$

where $d_i = |\mathcal{N}_i|$ is the degree of node i.

It decomposes decentralized learning into a series of sub-problem solving and variable exchanges. The local model $\boldsymbol{w}_i$ is updated in Equation 2.6, while Equation 2.7 represents the information exchange in the decentralized network.

Algorithm 1 shows the implementation of the synchronous ADMM/PDMM algorithm in decentralized learning. For convenience, in the remainder of the report, we will do the analysis with the case of PDMM ($\theta = 1$).

2.3. PRIVACY

This section will first introduce the threat model we consider in this thesis in subsection 2.3.1. The privacy preservation method by inserting subspace perturbations will be presented in subsection 2.3.2. The bound of privacy leakage in decentralized learning will also be discussed in subsection 2.3.3.

2.3.1. THREAT MODEL

In this report, we consider two types of adversaries in decentralized learning. The first type is eavesdropping, where the eavesdropper can gain access to messages exchanged over unencrypted channels. The second type is passive (honest-but-curious) nodes, which

Algorithm 1 Synchronous ADMM/PDMM

Initialization of $\boldsymbol{z}^{(0)}$ ▷ Initialization
for $t = 0, 1, \ldots$ **do**
  for each node $i \in \mathcal{V}$ **in parallel do** ▷ Update nodes
    $\boldsymbol{w}_i^{(t+1)} = \arg\min_{\boldsymbol{w}_i} \left(f_i(\boldsymbol{w}_i) + \sum_{j\in\mathcal{N}_i} \boldsymbol{z}_{i|j}^{(t)\top} \boldsymbol{B}_{i|j} \boldsymbol{w}_i + \frac{\rho d_i}{2} \boldsymbol{w}_i^2 \right)$
    for each $j \in \mathcal{N}_i$ **do**
      $\boldsymbol{y}_{i|j}^{(t+1)} = (1-\theta)\boldsymbol{z}_{j|i}^{(t)} + \theta\left(\boldsymbol{z}_{i|j}^{(t)} + 2\rho \boldsymbol{B}_{i|j} \boldsymbol{w}_i^{(t+1)}\right)$
    end for
  end for
  for each $i \in \mathcal{V}$, $j \in \mathcal{N}_i$ **do** ▷ Data exchange (unicast)
    $\text{Node}_j \leftarrow \text{Node}_i(\boldsymbol{y}_{i|j}^{(t+1)})$
  end for
  for each $i \in \mathcal{V}$, $j \in \mathcal{N}_i$ **do** ▷ Secondary Update
    $\boldsymbol{z}_{j|i}^{(t+1)} = \boldsymbol{y}_{i|j}^{(t+1)}$
  end for
end for

we refer to as corrupted nodes. These nodes perform the same training steps as other honest nodes during the training process and do not initiate attacks actively. However, they collect the received information and try to infer the private data of honest nodes. Combining both types of adversaries, the maximum information leakage in the network is: a) All messages exchanged over unencrypted channels during the entire duration; b) All information possessed and collected by corrupted nodes.

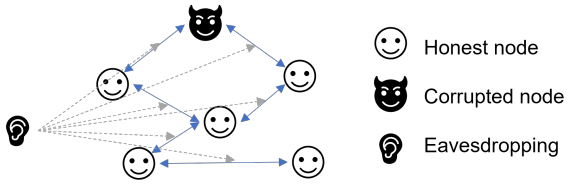

Figure 2.1: Threat Model

2.3.2. SUBSPACE-BASED PRIVACY PRESERVATION

In this section, we will explain a privacy-preserved subspace-based approach [28] proposed in distributed optimization.

The main idea of this method is to insert perturbation in the non-convergent subspace through the auxiliary variable. Let $\bar{H}_p = \text{span}(\boldsymbol{C}) + \text{span}(\boldsymbol{PC})$ and its orthogonal subspace $\bar{H}_p^{\perp} = \ker\left(\mathbf{C}^T\right) \cap \ker\left((\mathbf{PC})^T\right)$. So the auxiliary variable $\boldsymbol{z}^{(t)}$ can be separated into two components as $\boldsymbol{z}^{(t)} = \boldsymbol{z}_{\bar{H}_p}^{(t)} + \boldsymbol{z}_{\bar{H}_p^{\perp}}^{(t)}$, where $\boldsymbol{z}_{\bar{H}_p}^{(t)} = \Pi_{\bar{H}_p} \boldsymbol{z}^{(t)}$ and $\boldsymbol{z}_{\bar{H}_p^{\perp}}^{(t)} = \left(\boldsymbol{I} - \boldsymbol{\Pi}_{\bar{H}_p}\right)\boldsymbol{z}^{(t)}$. $\Pi_{\bar{H}_p}$ is the orthogonal projection onto $\bar{H}_p$. Since $\boldsymbol{C}^T \boldsymbol{z}_{\bar{H}_p^{\perp}}^{(t)} = \mathbf{0}$, the component in the orthogonal subspace does not act on primal variable updates. So we refer $\bar{H}_p$ and $\bar{H}_p^{\perp}$ as convergent

Algorithm 2 Decentralized FL using differential ADMM/PDMM

Initialization of $\boldsymbol{z}^{(0)}$ ▷ Initialization
for $t = 0, 1, \ldots$ **do**
  for each node $i \in \mathcal{V}$ **in parallel do** ▷ Update nodes
    $\boldsymbol{w}_i^{(t+1)} = \operatorname{argmin}_{\boldsymbol{w}_i} \left(f_i(\boldsymbol{w}_i) + \sum_{j \in \mathcal{N}_i} \boldsymbol{z}_{i|j}^{(t)\top} \boldsymbol{B}_{i|j} \boldsymbol{w}_i + \frac{\rho d_i}{2} \boldsymbol{w}_i^2 \right)$
    for each $j \in \mathcal{N}_i$ **do**
      $\boldsymbol{z}_{j|i}^{(t+1)} = (1-\theta)\boldsymbol{z}_{j|i}^{(t)} + \theta \left(\boldsymbol{z}_{i|j}^{(t)} + 2\rho \boldsymbol{B}_{i|j} \boldsymbol{w}_i^{(t+1)} \right)$
      $\Delta \boldsymbol{z}_{j|i}^{(t+1)} = \boldsymbol{z}_{j|i}^{(t+1)} - \boldsymbol{z}_{j|i}^{(t)}$
    end for
  end for
  for each $i \in \mathcal{V}$, $j \in \mathcal{N}_i$ **do** ▷ Data exchange (unicast)
    $\text{Node}_j \leftarrow \text{Node}_i(\Delta \boldsymbol{z}_{j|i}^{(t+1)})$
  end for
  for each $i \in \mathcal{V}$, $j \in \mathcal{N}_i$ **do** ▷ Secondary Update
    $\boldsymbol{z}_{j|i}^{(t+1)} = \boldsymbol{z}_{j|i}^{(t)} + \Delta \boldsymbol{z}_{j|i}^{(t+1)}$
  end for
end for

subspace and non-convergent subspace respectively and there is no compromise in accuracy and convergence. To ensure $\tilde{H}_p^{\perp}$ is non-empty, we need the number of edges larger or equal to the number of nodes,i.e. $E \geq N$, as shown in [28].

The implementation of subspace perturbation is by randomly initializing the auxiliary variable $\boldsymbol{z}_{j|i}^{(0)}$. With this approach, we can use differential ADMM/PDMM to prevent the direct leakage of $\boldsymbol{z}_{j|i}^{(t+1)}$ as shown in Algorithm 2.

For each round $t+1$ node i only need to transmit $\Delta \boldsymbol{z}_{j|i}^{(t+1)} = \boldsymbol{z}_{j|i}^{(t+1)} - \boldsymbol{z}_{j|i}^{(t)}$ to its neighbor $j \in \mathcal{N}_i$ and the receiving node j can easily recover $\boldsymbol{z}_{j|i}^{(t+1)} = \boldsymbol{z}_{j|i}^{(t)} + \Delta \boldsymbol{z}_{j|i}^{(t+1)}$.

Since $\boldsymbol{z}_{j|i}^{(t+1)}$ can only be determined whenever $\boldsymbol{z}_{j|i}^{(0)}$ is known, we can use encrypted channels once at time $t = 0$ to transmit the initialization $\boldsymbol{z}_{j|i}^{(0)}$ to protect from eavesdropping. So that eavesdropping only reveals

$$\left\{ \Delta \boldsymbol{z}_{j|i}^{(t)} : t \geq 1, (i,j) \in \mathcal{E} \right\}. \tag{2.8}$$

2.3.3. PRIVACY BOUND IN DISTRIBUTED SETTING

In this section, we shortly describe the derivation of the upper bound in privacy loss with differential PDMM.

Theorem 1. *Assume there is at least one corrupt node in the network, then for each honest node i with at least one honest neighbor, the adversary can learn $\{\boldsymbol{w}_i^{(t)} : t \geq 1\}$ as well as*

$$\nabla f_i \left(\boldsymbol{w}_i^{(t)} \right) - \nabla f_i \left(\boldsymbol{w}_i^{(t+2)} \right), \quad t \geq 0. \tag{2.9}$$

Proof. Since Equation 2.8 is revealed, the adversary has the knowledge of

$$\begin{aligned}\Delta \boldsymbol{z}_{j|i}^{(t+1)} - \Delta \boldsymbol{z}_{i|j}^{(t)} &= \boldsymbol{z}_{j|i}^{(t+1)} - \boldsymbol{z}_{j|i}^{(t)} - \left(\boldsymbol{z}_{i|j}^{(t)} - \boldsymbol{z}_{i|j}^{(t-1)}\right) \\ &= 2\rho \boldsymbol{B}_{i|j} \boldsymbol{w}_i^{(t+1)} - 2\rho \boldsymbol{B}_{i|j} \boldsymbol{w}_i^{(t)} \\ &= 2\rho \boldsymbol{B}_{i|j} (\boldsymbol{w}_i^{(t+1)} - \boldsymbol{w}_i^{(t)}), t \geq 0,\end{aligned} \tag{2.10}$$

i.e. $\boldsymbol{w}_i^{(t+1)} - \boldsymbol{w}_i^{(t)}$ for $\forall\, t \geq 0$ is known. Since the global model gradually converges, $\boldsymbol{w}_i^{(t)} \to \boldsymbol{w}^*$ for $\forall\, i \in \mathcal{V}$, $\boldsymbol{w}_i^{(t)}$ at any round t can be backwards computed from the knowledge of $\boldsymbol{w}^*$ on corrupted nodes.

For sub-problem Equation 2.6, it has optimality condition as

$$0 = \nabla f_i(\boldsymbol{w}_i^{(t+1)}) + \sum_{j \in \mathcal{N}_i} \boldsymbol{B}_{i|j}^\top \boldsymbol{z}_{i|j}^{(t)} + \rho d_i \boldsymbol{w}_i^{(t+1)}. \tag{2.11}$$

And from the two successive $\boldsymbol{z}$-updates in Equation 2.7, we have

$$\boldsymbol{z}_{i|j}^{(t+1)} - \boldsymbol{z}_{i|j}^{(t-1)} = 2\rho \boldsymbol{B}_{i|j} \left(\boldsymbol{w}_i^{(t)} - \boldsymbol{w}_j^{(t+1)}\right). \tag{2.12}$$

So combine Equation 2.11 and Equation 2.12, we have the difference of gradients at time t and $t+2$ as

$$\begin{aligned}\nabla f_i\left(\boldsymbol{w}_i^{(t)}\right) - \nabla f_i\left(\boldsymbol{w}_i^{(t+2)}\right) &= \sum_{j \in \mathcal{N}_i} \boldsymbol{B}_{i|j}^\top \left(\boldsymbol{z}_{i|j}^{(t+1)} - \boldsymbol{z}_{i|j}^{(t-1)}\right) + \rho d_i \left(\boldsymbol{w}_i^{(t+2)} - \boldsymbol{w}_i^{(t)}\right) \\ &= \rho d_i \left(\boldsymbol{w}_i^{(t)} + \boldsymbol{w}_i^{(t+2)}\right) - 2\rho \sum_{j \in \mathcal{N}_i} \boldsymbol{w}_j^{(t+1)}.\end{aligned} \tag{2.13}$$

Since $\boldsymbol{w}_i^{(t)}$ for $\forall\, i \in \mathcal{V}$ is inferred by the adversary, all the terms on the RHS are known. Thus $\nabla f_i\left(\boldsymbol{w}_i^{(t)}\right) - \nabla f_i\left(\boldsymbol{w}_i^{(t+2)}\right)$ is revealed.

□

3

SETUP

In this chapter, we will introduce the setups for the relevant experiments in the subsequent chapters. Section 3.1 proposes the two topologies used and the optimization-based and average consensus-based algorithms that apply in collaborative learning. Section 3.2 presents two case studies, the convex and non-convex machine learning models we used.

3.1. TOPOLOGY

Decentralized Network

We use random geometric graph (RGG) [29] to simulate the decentralized network $\mathcal{G}$, as shown in Figure 3.1. In RGG, vertexes are randomly placed in a unit cube. An edge, i.e. the transmission channel, connects two vertexes, i.e. the number of nodes/clients when their distance is smaller than a certain radius r. We consider these nodes uniformly distributed in the two-dimensional region. For a 2-dimensional cube, as long as $r \geq \sqrt{\frac{log(N)}{N}}$, the graph is connected with probability $P \geq 1 - \frac{1}{N^2}$ [30], where N is the number of vertexes.

Then we consider the privacy-preserving distributed optimization with subspace perturbation and differential PDMM as shown in Algorithm 2. Encrypted channels are used at time $t = 0$ to transmitted $\boldsymbol{z}^{(0)}$, thus in the following steps we can limit the information leakage in the unencrypted channels into the variation of $\boldsymbol{z}^{(t)}$, as shown in Equation 2.8. The node construction is object-oriented, so each node has its own local variables and a list of neighbours.

Centralized Network

We set the network to have the same number of nodes as the decentralized network. In the centralized network, there is no direct transmission between nodes. All nodes connect and only connect with a central server. The server does not own private data and does not perform training, but only takes the role of aggregation.

We consider the Federated Averaging (FedAvg) [1] as a representative averaging consensus-based FL algorithm in such a topology. In each communication round, activated nodes first train a local model and transmit the weights to the server; then the server does the

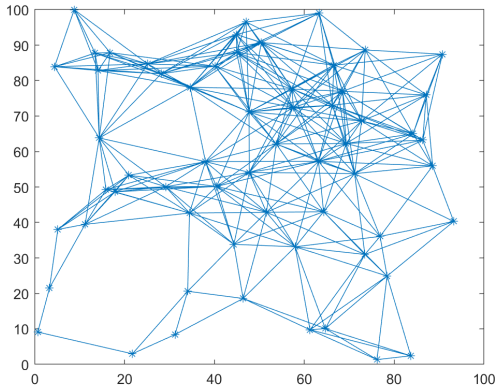

Figure 3.1: Example of RGG with $N = 60$

weighted averaging to get a global model; the global model is sent back to those nodes and nodes repeat the steps in the next round. n_i is the local dataset size and $n = \sum_{i=1}^{N} n_i$ is the total number of samples in the systems. So the weight of each local model can be expressed as $\frac{n_i}{n}$. The pseudo-code is shown in Algorithm 3.

Empirically, for non-convex models, the model will tend to converge to the same local minimum with common initialization [1].

A special case of FedAvg is FedSgd as shown in Algorithm 4. Consider the FedSgd with $C = 1, E = 1$ and $B = \infty$ (batchsize equals local dataset size). In this case, the exchange of weights is equivalent to the exchange of gradients. Since $\nabla f_i(\boldsymbol{w}_i^{(t)}) = \frac{\boldsymbol{w}^{(t)} - \boldsymbol{w}^{(t+1)}}{\alpha}$, so that $\boldsymbol{w}^{(t+1)} = \sum_{i=1}^{N} \frac{n_i}{n} \boldsymbol{w}_i^{(t+1)} = \boldsymbol{w}^{(t)} - \alpha \sum_{i=1}^{N} \frac{n_i}{n} \nabla f_i(\boldsymbol{w}_i^{(t)})$.

3.2. CASE STUDY

3.2.1. DISTRIBUTED LOGISTIC REGRESSION

We first consider a classical convex problem in machine learning. Logistic regression can be used in classification problems. With the nonlinear mapping, we restrict the binary dependent variable into the range $(0, 1)$, so we can choose the cutoff value (0.5) to indicate the 0/1 label.

For each node i, consider the number of samples as n_i and the number of features as u. There are n_i pairs of $(\boldsymbol{x}_{ik}, \ell_{ik})$ where $\boldsymbol{x}_{ik} \in \mathbb{R}^u$ is the feature vector and $\ell_{ik} \in \{0, 1\}$ is the label. These are the local information node i holds.

For binary classification problems with logistic regression, we construct the model with parameter $(\boldsymbol{w}_i, b_i)$, where $\boldsymbol{w}_i \in \mathbb{R}^u$ and $b \in \mathbb{R}$. We map $\boldsymbol{w}_i^\top \boldsymbol{x}_{ik} + b_i$ into range $(0, 1)$ with sigmoid activation, i.e output $y_{ik} = \sigma(\boldsymbol{w}_i^\top \boldsymbol{x}_{ik} + b_i) = \frac{1}{1+e^{-(\boldsymbol{w}_i^\top \boldsymbol{x}_{ik} + b_i)}}$.

So the local cost function, which defines as the cross entropy, can be presented as

$$f_i(\boldsymbol{w}_i, b_i) = -\sum_{k=1}^{n_i} \{\ell_{ik} \log y_{ik} + (1 - \ell_{ik}) \log(1 - y_{ik})\}, \tag{3.1}$$

Algorithm 3 Federated Averaging

B is the local minibatch size, E is the number of local epochs, α is the learning rate, and C is the node fraction.

for each round $t = 0, 1, \ldots$ **do** ▷ Server executation
  $m \leftarrow \max(C \cdot N, 1)$
  $S_t \leftarrow$ (random set of m nodes)
  for each node $i \in S_t$ **in parallel do**
    $\boldsymbol{w}_i^{(t+1)} \leftarrow \text{NodeUpdate}\,(i, \boldsymbol{w}^{(t)})$
  end for
  $\boldsymbol{w}^{(t+1)} \leftarrow \sum_{i=1}^{N} \frac{n_i}{n} \boldsymbol{w}_i^{(t+1)}$
end for

NodeUpdate$(i, \boldsymbol{w})$:
$\mathscr{B} \leftarrow$ (split local dataset $\mathscr{D}_i$ into batches of size B)
for each epoch e from 1 to E **do**
  for batch $b \in \mathscr{B}$ **do**
    $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \nabla f(\boldsymbol{w}; b)$
  end for
end for
return $\boldsymbol{w}$ to server

where $\log(\cdot)$ denotes the natural logarithm.

Its gradient can be expressed as

$$\frac{\partial f_i}{\partial \boldsymbol{w}_i} = \sum_{k=1}^{n_i} \left(y_{ik} - \ell_{ik}\right) \boldsymbol{x}_{ik}, \tag{3.2}$$

$$\frac{\partial f_i}{\partial b_i} = \sum_{k=1}^{n_i} \left(y_{ik} - \ell_{ik}\right). \tag{3.3}$$

For multinomial classification, the activation is modified to softmax, since we need the outputs to be the confidences of each class, thus the outputs satisfy $\sum_{c=1}^{C} y_{ik,c} = 1$, where we have C classes. we construct the model with parameter $(\boldsymbol{w}_i, \boldsymbol{b}_i)$, where $\boldsymbol{w}_i = [\boldsymbol{w}_{i,1}, \ldots, \boldsymbol{w}_{i,C}] \in \mathbb{R}^{u \times C}$ and $\boldsymbol{b}_i = [b_{i,1}, \ldots, b_{i,C}] \in \mathbb{R}^C$. The output for each class c is $y_{ik,c} = \sigma_c(\boldsymbol{w}_i^\top \boldsymbol{x}_{ik} + b_{i,c}) = \frac{e^{\boldsymbol{w}_{i,c}^\top \boldsymbol{x}_{ik} + b_{i,c}}}{\sum_{r=1}^{C} e^{\boldsymbol{w}_{i,r}^\top \boldsymbol{x}_{ik} + b_{i,r}}}$.

The cost function is correspondingly modified as

$$f_i(\boldsymbol{w}_i, b_i) = -\sum_{k=1}^{n_i} \log\left(y_{ik,\ell_{ik}}\right), \tag{3.4}$$

as well as the gradient

$$\frac{\partial f_i}{\partial \boldsymbol{w}_i} = \sum_{k=1}^{n_i} \left(y_{ik,c} - \delta_{c,\ell_{ik}}\right) \boldsymbol{x}_{ik} \tag{3.5}$$

Algorithm 4 Federated SGD

for each round $t = 0, 1, \ldots$ **do** ▷ Server executation
  for each node $i \in \mathcal{V}$ **in parallel do**
    $\nabla f_i(\boldsymbol{w}_i^{(t+1)}) \leftarrow \text{NodeUpdate}\ (i, \nabla f(\boldsymbol{w}^{(t)}))$
  end for
  $\nabla f(\boldsymbol{w}^{(t+1)}) \leftarrow \sum_{i=1}^{N} \frac{n_i}{n} \nabla f_i(\boldsymbol{w}_i^{(t+1)})$
end for

NodeUpdate$(i, \nabla f(\boldsymbol{w}))$:
$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \nabla f(\boldsymbol{w})$
return $\nabla f(\boldsymbol{w})$ to server

$$\frac{\partial f_i}{\partial b_i} = \sum_{k=1}^{n_i} y_{ik,c} - \delta_{c,\ell_{ik}} \tag{3.6}$$

where $\delta_{c,\ell_{ik}}$ is the Kronecker-delta.

3.2.2. DISTRIBUTED MULTI-LAYER PERCEPTRON

Then we extend the case into a non-convex problem with a 2-layer perceptron. Multi-layer perceptron is a simple model structure for the classification algorithm which is combined with several fully connected layers and non-linear activations. In each layer, each neuron is connected with all neurons from the last layer and uses their weighted sum as the input of the activation function. It is a classical structure which can solve linearly non-separable problems like XOR.

For the multi-layer perceptron, the gradient of f_i can be calculated by the chain rule. We set the activation function in the hidden layer as Rectified Linear Unit (ReLU). ReLU is a nonlinear activation which defines as the positive part of its argument, i.e. $ReLU(z) = \max(0, z) = \frac{z+|z|}{2}$. Its gradient is the step function, i.e. $ReLU'(z) = \begin{cases} 1 & \text{If} \quad z > 0 \\ 0 & \text{If} \quad z < 0 \end{cases}$. The output layer also uses sigmoid for binary classification and softmax for multi-classification.

For binary classification, Assuming the hidden layer has h neurons, then for the model parameters $(\boldsymbol{w}_{i1}, \boldsymbol{b}_{i1}, \boldsymbol{w}_{i2}, b_{i2})$, $\boldsymbol{w}_{i1} \in \mathbb{R}^{u \times h}$, $\boldsymbol{b}_{i1} \in \mathbb{R}^h$, $\boldsymbol{w}_{i2} \in \mathbb{R}^h$ and $b_{i2} \in \mathbb{R}$.

The forward propagation is

$$\begin{aligned} \boldsymbol{z}_{i1} &= \boldsymbol{w}_{i1}^{\mathsf{T}} \boldsymbol{x}_{ik} + \boldsymbol{b}_{i1}, \\ \boldsymbol{z}_{i2} &= \boldsymbol{w}_{i2}^{\mathsf{T}} \text{ReLU}(\boldsymbol{z}_{i1}) + b_{i2}, \\ y_{ik} &= \sigma(\boldsymbol{z}_{i2}). \end{aligned} \tag{3.7}$$

So the gradients are known as

$$\begin{aligned}
\frac{\partial f_i}{\partial \boldsymbol{w}_{i2}} &= \sum_{k=1}^{n_i} \left(y_{ik} - \ell_{ik}\right) \mathrm{ReLU}\left(\boldsymbol{z}_{i1}\right) \\
\frac{\partial f_i}{\partial b_{i2}} &= \sum_{k=1}^{n_i} \left(y_{ik} - \ell_{ik}\right) \\
\frac{\partial f_i}{\partial \boldsymbol{w}_{i1}} &= \sum_{k=1}^{n_i} \left(y_{ik} - \ell_{ik}\right) \boldsymbol{w}_{i2} \boldsymbol{x}_i^{\mathsf{T}} \odot \mathrm{ReLU}'\left(\boldsymbol{z}_{i1}\right) \\
\frac{\partial f_i}{\partial \boldsymbol{b}_{i1}} &= \sum_{k=1}^{n_i} \left(y_{ik} - \ell_{ik}\right) \boldsymbol{w}_{i2} \odot \mathrm{ReLU}'\left(\boldsymbol{z}_{i1}\right),
\end{aligned} \tag{3.8}$$

The similar derivation is also applied to the case of multi-classification.

4

DISTRIBUTED OPTIMIZATION VIA INEXACT PDMM

In decentralized learning, solving sub-problems in Equation 2.6 exactly is often challenging. Sub-problems can only be solved exactly in a few cases, such as average consensus, linear regression, etc. In this section, we mention two common approaches for updating sub-problems in an approximate manner. One approach is to use iterative methods in Section 4.1, where the approximate value is obtained through a finite number of iterations. Another approach is to use the quadratic approximation in Section 4.2. The original privacy bound in Equation 2.9 is derived based on the optimality condition in Equation 2.11, so when using approximate updates, the boundary of privacy leakage also changes. We also discuss the changes in the privacy bound. Finally, the utility of inexact updates will be shown in experiments in Section 4.3.

4.1. ITERATION METHOD

The first approach is to use any learning-based iterative method with several steps in optimization to get an approximated solution. For the nodes, only a sufficiently close solution needs to be obtained, not caring about its iterative process. Therefore, we can use any method provided that convergence is satisfied.

Convergence Analysis

The inexact solution can be seen as introducing additive noise on the weights. From the quantization scheme in distributed optimization [31], it is known that the optimization is sure to converge if the sequence $\left\|\boldsymbol{n}^{(t)}\right\|$ is finitely summable, where $\boldsymbol{n}^{(t)}$ is the noise term in the auxiliary variable $\boldsymbol{z}$.

Let $\hat{\boldsymbol{w}}^{(t)} = \boldsymbol{w}^{(t)} + \boldsymbol{e}^{(t)}$ is the inexact solution of the subproblem in round t, $\boldsymbol{e}^{(t)}$ is the error term. We can obtain $\hat{\boldsymbol{z}}^{(t)} = \boldsymbol{z}^{(t)} + 2\theta\rho\boldsymbol{C}\boldsymbol{e}^{(t)}$.

Since $\left\|\boldsymbol{n}^{(t)}\right\| = \left\|2\theta\rho\boldsymbol{C}\boldsymbol{e}^{(t)}\right\| = 2\theta\rho\left\|\boldsymbol{C}\boldsymbol{e}^{(t)}\right\| \leq 2\theta\rho\|\boldsymbol{C}\|\left\|\boldsymbol{e}^{(t)}\right\|$, thus as long as the sequence of error $\left\|\boldsymbol{e}^{(t)}\right\|$ satisfies the finite summable condition, there has a same convergence guarantee as the case of quantization.

Privacy Bound

Assuming we use the gradient descent with $k_{\max}$ steps in each w-update. Let k denote the inner iteration and α denote the fixed learning rate. From Equation 2.6, the inner iteration at round t can be expressed with the gradient of the new sub-problem as

$$\text{For} k = 1,\dots,k_{\max}: \boldsymbol{w}_i^{(t,k)} = \boldsymbol{w}_i^{(t,k-1)} - \alpha(\nabla f_i(\boldsymbol{w}_i^{(t,k-1)}) + \sum_{j\in\mathcal{N}_i} \boldsymbol{B}_{i|j}^{\top}\boldsymbol{z}_{i|j}^{(t)} + \rho d_i \boldsymbol{w}_i^{(t,k-1)}), \quad (4.1)$$

where the initial weight $\boldsymbol{w}_i^{(t,0)} = \boldsymbol{w}_i^{(t)} = \boldsymbol{w}_i^{(t-1,k_{\max})}$ and the transmitted weight is $\boldsymbol{w}_i^{(t+1)} = \boldsymbol{w}_i^{(t,k_{\max})}$.

In terms of the adversary, these intermediate states are kept at local and not revealed, i.e, only $\boldsymbol{w}_i^{(t,0)}$, as well as $\boldsymbol{w}_i^{(t,k_{\max})} (or \boldsymbol{w}_i^{(t+1,0)})$ for $t \geq 1$ is exposed, which brings the difficulty for the attack. Moreover, the derivation only applies to the single batch update. It means for the case where nodes contain a large-scale dataset, the more commonly used approach, stochastic gradient descent (SGD), would also blur the upper bound that the adversary can obtain.

We consider two special cases in which the privacy boundary is still clear. The first one is the case when $k_{\max} = 1$, the inexact w-update can be simplified as

$$\boldsymbol{w}_i^{(t+1)} = \boldsymbol{w}_i^{(t)} - \alpha(\nabla f_i(\boldsymbol{w}_i^{(t)}) + \sum_{j\in\mathcal{N}_i} \boldsymbol{B}_{i|j}^{\top}\boldsymbol{z}_{i|j}^{(t)} + \rho d_i \boldsymbol{w}_i^{(t)}). \quad (4.2)$$

Correspondingly, in this case, the bound should be modified as

$$\nabla f_i\left(\boldsymbol{w}_i^{(t)}\right) - \nabla f_i\left(\boldsymbol{w}_i^{(t+2)}\right) = (\rho d_i - \frac{1}{\alpha})(\boldsymbol{w}_i^{(t+2)} - \boldsymbol{w}_i^{(t)}) + \frac{1}{\alpha}\boldsymbol{w}_i^{(t+3)} + (2\rho d_i - \frac{1}{\alpha})\boldsymbol{w}_i^{(t+1)} - 2\rho \sum_{j\in\mathcal{N}_i} \boldsymbol{w}_j^{(t+2)}, \quad (4.3)$$

Another case is $k_{\max} \to \infty$. For convex cost function $f_i(\boldsymbol{w}_i^{(t)})$, the exact solution can be reached with infinite iterations. Thus we can use the optimality condition in Equation 2.11 to derive the privacy bound as

$$\nabla f_i\left(\boldsymbol{w}_i^{(t)}\right) - \nabla f_i\left(\boldsymbol{w}_i^{(t+2)}\right) = \rho d_i\left(\boldsymbol{w}_i^{(t)} + \boldsymbol{w}_i^{(t+2)}\right) - 2\rho \sum_{j\in\mathcal{N}_i} \boldsymbol{w}_j^{(t+1)}. \quad (4.4)$$

4.2. QUADRATIC APPROXIMATION

[32] provides the quadratic approximation of PDMM (QA-PDMM) with λ-update version. Here we give a derivation of z-update.

First we quadratically approximate the cost function with a positive constant μ as

$$f(\boldsymbol{w}) \approx f\left(\boldsymbol{w}^{(t)}\right) + \nabla f\left(\boldsymbol{w}^{(t)}\right)^{\top}\left(\boldsymbol{w} - \boldsymbol{w}^{(t)}\right) + \frac{\mu}{2}\left\|\boldsymbol{w} - \boldsymbol{w}^{(t)}\right\|^2. \quad (4.5)$$

So the minimization has the quadratically approximated form as

$$\boldsymbol{w}^{(t+1)} \approx \arg\min_{\boldsymbol{w}} f\left(\boldsymbol{w}^{(t)}\right) + \nabla f\left(\boldsymbol{w}^{(t)}\right)^{\top}\left(\boldsymbol{w} - \boldsymbol{w}^{(t)}\right) + \frac{\mu}{2}\left\|\boldsymbol{w} - \boldsymbol{w}^{(t)}\right\|^2 + \boldsymbol{w}^{\top}\boldsymbol{C}^{\top}\boldsymbol{z}^{(t)} + \frac{\rho}{2}\|\boldsymbol{C}\boldsymbol{w}\|^2, \quad (4.6)$$

which implies its optimal condition satisfies

$$\nabla f(\boldsymbol{w}^{(t)}) + \mu(\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)}) + \rho\boldsymbol{C}^{\top}\boldsymbol{C}\boldsymbol{w}^{(t+1)} + \boldsymbol{C}^{\top}\boldsymbol{z}^{(t)} = 0. \quad (4.7)$$

So the $\boldsymbol{w}$ update can be written as

$$\boldsymbol{w}^{(t+1)} = (\mu \boldsymbol{I} + \rho \boldsymbol{C}^\intercal \boldsymbol{C})^{-1} \left(\mu \boldsymbol{w}^{(t)} - \nabla f\left(\boldsymbol{w}^{(t)}\right) - \boldsymbol{C}^\intercal \boldsymbol{z}^{(t)}\right). \tag{4.8}$$

Thus the individual update for each node i in Equation 2.6 finally has the following form

$$\boldsymbol{w}_i^{(t+1)} = \frac{\mu \boldsymbol{w}_i^{(t)} - \nabla f_i(\boldsymbol{w}_i^{(t)}) - \sum_{j \in \mathcal{N}_i} \boldsymbol{B}_{i|j}^\intercal \boldsymbol{z}_{i|j}^{(t)}}{\mu + \rho d_i}. \tag{4.9}$$

Convergence Analysis

Theorem 2. *Assume $f(\boldsymbol{w})$ is m-strongly convex and β-smooth, then as long as $\mu > \frac{\beta^2}{2m}$, we have*

$$\boldsymbol{w}^{(t)} \to \boldsymbol{w}^*.$$

Proof. With QA-PDMM, the iterates in Equation 2.3-Equation 2.5 are rewritten as

$$\boldsymbol{w}^{(t+1)} = \arg\min_{\boldsymbol{w}} f\left(\boldsymbol{w}^{(t)}\right) + \nabla f\left(\boldsymbol{w}^{(t)}\right)^\intercal \left(\boldsymbol{w} - \boldsymbol{w}^{(t)}\right) + \frac{\mu}{2}\left\|\boldsymbol{w} - \boldsymbol{w}^{(t)}\right\|^2 + \boldsymbol{w}^\intercal \boldsymbol{C}^\intercal \boldsymbol{z}^{(t)} + \frac{\rho}{2}\|\boldsymbol{C}\boldsymbol{w}\|^2, \tag{4.10}$$

$$\boldsymbol{y}^{(t+1)} = \boldsymbol{z}^{(t)} + 2\rho \boldsymbol{C}\boldsymbol{w}^{(t+1)}, \tag{4.11}$$

$$\boldsymbol{z}^{(t+1)} = \boldsymbol{P}\boldsymbol{y}^{(t+1)}. \tag{4.12}$$

Thus we have

$$\begin{aligned} \|\boldsymbol{z}^{(t+1)} - \boldsymbol{z}^*\|_2^2 &= \|\boldsymbol{y}^{(t+1)} - \boldsymbol{y}^*\|_2^2 \\ &= \|\boldsymbol{z}^{(t)} - \boldsymbol{z}^* + 2\rho C(\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^*)\|_2^2 \\ &= \|\boldsymbol{z}^{(t)} - \boldsymbol{z}^*\|_2^2 + 4\rho(\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^*)^\intercal C^\intercal \left(\boldsymbol{z}^{(t)} - \boldsymbol{z}^* + \rho C(\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^*)\right) \end{aligned} \tag{4.13}$$

Combined with Equation 4.7, Equation 4.13 can be written as

$$\|\boldsymbol{z}^{(t+1)} - \boldsymbol{z}^*\|_2^2 = \|\boldsymbol{z}^{(t)} - \boldsymbol{z}^*\|_2^2 - 4\rho(\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^*)^\intercal \left(\nabla f(\boldsymbol{w}^{(t)}) - \nabla f(\boldsymbol{w}^*) + \mu(\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)})\right). \tag{4.14}$$

We have the different terms on the RHS of Equation 4.14 as

$$\mu(\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^*)^\intercal \left(\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)}\right) = \frac{\mu}{2}\|\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^*\|_2^2 + \frac{\mu}{2}\|\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)}\|_2^2 - \frac{\mu}{2}\|\boldsymbol{w}^{(t)} - \boldsymbol{w}^*\|_2^2, \tag{4.15}$$

since $(\boldsymbol{a}-\boldsymbol{b})^T(\boldsymbol{a}-\boldsymbol{c}) = \frac{1}{2}\left(\|\boldsymbol{a}-\boldsymbol{c}\|_2^2 - \|\boldsymbol{b}-\boldsymbol{c}\|_2^2 + \|\boldsymbol{a}-\boldsymbol{b}\|_2^2\right)$, and

$$\begin{aligned} & (\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^*)^\intercal \left(\nabla f(\boldsymbol{w}^{(t)}) - \nabla f(\boldsymbol{w}^*)\right) \\ = \; & (\boldsymbol{w}^{(t)} - \boldsymbol{w}^*)^\intercal \left(\nabla f(\boldsymbol{w}^{(t)}) - \nabla f(\boldsymbol{w}^*)\right) + (\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)})^\intercal \left(\nabla f(\boldsymbol{w}^{(t)}) - \nabla f(\boldsymbol{w}^*)\right) \\ \geq \; & (\boldsymbol{w}^{(t)} - \boldsymbol{w}^*)^\intercal \left(\nabla f(\boldsymbol{w}^{(t)}) - \nabla f(\boldsymbol{w}^*)\right) - \tfrac{\mu}{2}\|\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)}\|_2^2 - \tfrac{1}{2\mu}\|\nabla f(\boldsymbol{w}^{(t)}) - \nabla f(\boldsymbol{w}^*)\|_2^2, \end{aligned} \tag{4.16}$$

since $2\boldsymbol{a}^T\boldsymbol{b} \leq \|\boldsymbol{a}\|_2^2 + \|\boldsymbol{b}\|_2^2$.

Since $f(\boldsymbol{w})$ is assumed m-strongly convex and β-smooth, we have

$$(\boldsymbol{w}^{(t)} - \boldsymbol{w}^*)^\intercal \left(\nabla f(\boldsymbol{w}^{(t)}) - \nabla f(\boldsymbol{w}^*)\right) \geq m\|\boldsymbol{w}^{(t)} - \boldsymbol{w}^*\|_2^2, \tag{4.17}$$

and

$$\|\nabla f(\boldsymbol{w}^{(t)}) - \nabla f(\boldsymbol{w}^*)\|_2 \leq \beta \|\boldsymbol{w}^{(t)} - \boldsymbol{w}^*\|_2. \tag{4.18}$$

So the inequality in Equation 4.16 becomes

$$\begin{aligned} & (\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^*)^\intercal \left(\nabla f(\boldsymbol{w}^{(t)}) - \nabla f(\boldsymbol{w}^*)\right) \\ \geq\ & m\|\boldsymbol{w}^{(t)} - \boldsymbol{w}^*\|_2^2 - \tfrac{\mu}{2}\|\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)}\|_2^2 - \tfrac{\beta^2}{2\mu}\|\boldsymbol{w}^{(t)} - \boldsymbol{w}^*\|_2^2 \\ =\ & (m - \tfrac{\beta^2}{2\mu})\|\boldsymbol{w}^{(t)} - \boldsymbol{w}^*\|_2^2 - \tfrac{\mu}{2}\|\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)}\|_2^2, \end{aligned} \tag{4.19}$$

With Equation 4.15 and Equation 4.19, Equation 4.14 becomes

$$\|\boldsymbol{z}^{(t+1)} - \boldsymbol{z}^*\|_2^2 - \|\boldsymbol{z}^{(t)} - \boldsymbol{z}^*\|_2^2 \leq -4\rho(m - \frac{\beta^2}{2\mu})\|\boldsymbol{w}^{(t)} - \boldsymbol{w}^*\|_2^2 - 4\rho\frac{\mu}{2}\left(\|\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^*\|_2^2 - \|\boldsymbol{w}^{(t)} - \boldsymbol{w}^*\|_2^2\right). \tag{4.20}$$

Iterating over t yields

$$\begin{aligned} & 4\rho(m - \tfrac{\beta^2}{2\mu})\textstyle\sum_{\ell=1}^{t}\|\boldsymbol{w}^{(\ell)} - \boldsymbol{w}^*\|_2^2 \\ \leq\ & 2\rho\mu\left(\|\boldsymbol{w}^{(0)} - \boldsymbol{w}^*\|_2^2 - \|\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^*\|_2^2\right) + \|\boldsymbol{z}^{(0)} - \boldsymbol{z}^*\|_2^2 - \|\boldsymbol{z}^{(t+1)} - \boldsymbol{z}^*\|_2^2 \\ \leq\ & 2\rho\mu\|\boldsymbol{w}^{(0)} - \boldsymbol{w}^*\|_2^2 + \|\boldsymbol{z}^{(0)} - \boldsymbol{z}^*\|_2^2. \end{aligned} \tag{4.21}$$

So the RHS of Equation 4.21 is bounded and thus we can say that, as long as $m - \frac{\beta^2}{2\mu} > 0$, i.e. $\mu > \frac{\beta^2}{2m}$, we have

$$\lim_{t\to\infty}\left\|\boldsymbol{w}^{(t)} - \boldsymbol{w}^*\right\|_2^2 = 0,$$

thus

$$\boldsymbol{w}^{(t)} \to \boldsymbol{w}^*.$$

□

Privacy Bound

From Equation 4.9, the revised privacy bound can be expressed as

$$\begin{aligned} & \nabla f_i\left(\boldsymbol{w}_i^{(t)}\right) - \nabla f_i\left(\boldsymbol{w}_i^{(t+2)}\right) \\ =\ & -(\mu + \rho d_i)(\boldsymbol{w}_i^{(t+1)} - \boldsymbol{w}_i^{(t+3)}) + \mu(\boldsymbol{w}_i^{(t)} - \boldsymbol{w}_i^{(t+2)}) - \textstyle\sum_{j\in\mathcal{N}_i}\boldsymbol{B}_{i|j}^\intercal(\boldsymbol{z}_{i|j}^{(t)} - \boldsymbol{z}_{i|j}^{(t+2)}) \\ =\ & -(\mu + \rho d_i)(\boldsymbol{w}_i^{(t+1)} - \boldsymbol{w}_i^{(t+3)}) + \mu(\boldsymbol{w}_i^{(t)} - \boldsymbol{w}_i^{(t+2)}) + 2\rho d_i\boldsymbol{w}_i^{(t+1)} - 2\rho\textstyle\sum_{j\in\mathcal{N}_i}\boldsymbol{w}_j^{(t+2)} \\ =\ & (\rho d_i - \mu)\boldsymbol{w}_i^{(t+1)} + (\mu + \rho d_i)\boldsymbol{w}_i^{(t+3)} + \mu(\boldsymbol{w}_i^{(t)} - \boldsymbol{w}_i^{(t+2)}) - 2\rho\textstyle\sum_{j\in\mathcal{N}_i}\boldsymbol{w}_j^{(t+2)}. \end{aligned} \tag{4.22}$$

So for the QA-PDMM, the adversary can still derive the difference of gradients when knowing the value of μ. The case is similar with $k_{\max} = 1$, since the w-update only has a one-time calculation and the gradient difference can eventually be written as an expression related to $\boldsymbol{w}$.

4.3. EXPERIMENTS

We first validate it on a simple case of binary logistic regression in Chapter 3. A connected RGG network with $N = 60$ nodes is generated. Each node i holds $n_i = 1$ data samples $\boldsymbol{x}_{ik} \in \mathbb{R}^2$ and binary labels $\ell_i \in \{0, 1\}$. The two datasets were generated from random samples drawn from a unit variance Gaussian distribution with mean $\boldsymbol{\mu}_0 = (-1, -1)^\mathsf{T}$ ($\ell_{ik} = 0$) and mean $\boldsymbol{\mu}_1 = (1, 1)^\mathsf{T}$ ($\ell_{ik} = 1$). PDMM was used for decentralized learning with constant $\rho = 0.4$ and the two weight updates mentioned in this chapter were implemented. In the iterative method, a fixed learning rate of $\alpha = 0.1$ is used for gradient descent. We use the FedAvg in centralized FL as the baseline with the same dataset and single gradient descent iteration having the same learning rate as the rate used in decentralized FL.

Figure 4.1 shows the whole dataset in the system and a bunch of well-trained local models after global convergence. These local models converge consistently (overlapped) and demonstrate good performance in classifying the entire dataset.

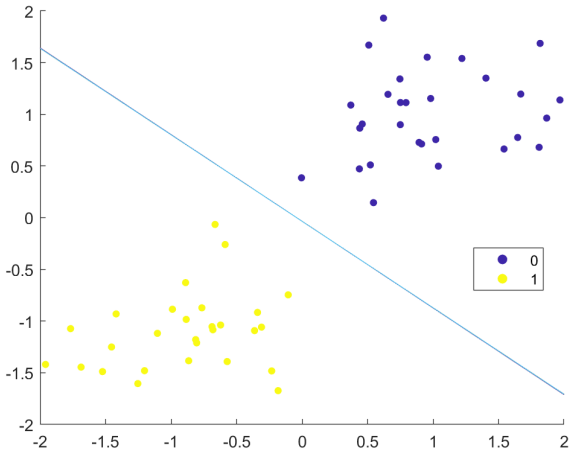

Figure 4.1: Dataset and well-trained models

We verify the effectiveness of both inexact updates, as shown in Figure 4.2. For the iterative method, only a small $k_{\max}$ is needed in each round to approximate the solution of the subproblem. It can be observed that the increase in $k_{\max}$ does not bring an obvious faster convergence (curves of $k_{\max} \geq 10$ almost overlapped). This implies that nodes do not need to allocate too many computational resources to approximate the exact solutions of the subproblems. And for quadratic approximation, finding an appropriate value for μ is important. A smaller value of μ could accelerate convergence but must satisfy $\mu > \frac{\beta^2}{2m}$.

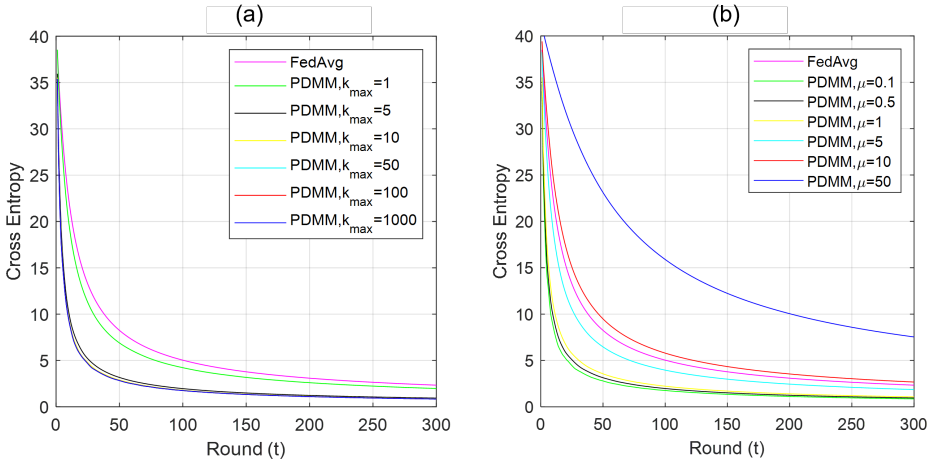

Figure 4.2: PDMM with the inexact update. (a) Iterative method. (b) Quadratic Approximation

5

Gradient Information-based Attack in Federated Learning

In this chapter, we analyze the information obtained by eavesdropping and passive nodes in FL, along with the associated attacks. Section 5.1 describes the gradient information leakage in communication channels. Section 5.2 discusses the attacks adversaries can employ to exploit the leaked information. We highlight the differences in privacy-preserving aspects of average consensus-based and optimization-based FL in Section 5.3. Finally, Section 5.4 provides corresponding experimental results.

5.1. Information Leakage

In this section, we describe the information leakage in both average consensus-based and optimization-based approaches under the assumed threat model in subsection 2.3.1.

5.1.1. Gradient Leakage

Considering the threat model discussed in subsection 2.3.1, direct leakage of gradients only occurs in a few cases. In the average consensus-based approach in both centralized and decentralized topology, the exchange of gradients/weights in each update can lead to gradient leakage, such as FedSGD. In the former case, the gradients are obtained directly from eavesdropping; while in the latter case, the gradient information can be derived by taking the difference between weights from two consecutive communication rounds.

Another possible case is the original PDMM algorithm. Since the original PDMM algorithm transmits the variable $\boldsymbol{z}$ directly, eavesdropping can reveal $\left\{\boldsymbol{z}_{j|i}^{(t)} : t \geq 1, (i,j) \in \mathcal{E}\right\}$. Also as shown in subsection 2.3.3, if the adversary continuously eavesdrops on the messages in a sufficient number of communication rounds, the local models $\boldsymbol{w}_i^{(t)}$ can be approximated. Thus in the optimal condition in Equation 2.11 we can see that the last two terms on the RHS, $\sum_{j\in\mathcal{N}_i} \boldsymbol{B}_{i|j}^{\top}\boldsymbol{z}_{i|j}^{(t)}$ and $\rho d_i \boldsymbol{w}_i^{(t+1)}$ are revealed, gradients $\nabla f_i(\boldsymbol{w}_i^{(t)})$ can be calculated as $-\sum_{j\in\mathcal{N}_i} \boldsymbol{B}_{i|j}^{\top}\boldsymbol{z}_{i|j}^{(t)} - \rho d_i \boldsymbol{w}_i^{(t+1)}$. In conclusion, we can say that differential PDMM with subspace perturbation can avoid direct gradient leakage.

But subspace perturbation is only suitable for the decentralized topology as we mentioned in subsection 2.3.2 that the number of edges should be larger or equal to the number of nodes to ensure subspace $\tilde{H}_p^{\perp}$ is non-empty. Recently research [33] shows that optimization-based algorithms in centralized topology, like FedSplit [34] and SCAFFOLD [35] are actually the special cases of PDMM. We have the PDMM form in the centralized network as [33]

$$\text{nodes} \begin{cases} \boldsymbol{w}_i^{(t+1)} = \arg\min \left(f_i(\boldsymbol{w}_i) + \frac{\rho}{2} \left\| \boldsymbol{w}_i - \boldsymbol{z}_{s|i}^{(t)} \right\|^2 \right) \\ \boldsymbol{z}_{i|s}^{(t+1)} = 2\boldsymbol{w}_i^{(t+1))} - \boldsymbol{z}_{s|i}^{(t)} \end{cases} \tag{5.1}$$

$$\text{server} \begin{cases} \boldsymbol{w}_s^{(t+1)} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{z}_{i|s}^{(t+1)} \\ \boldsymbol{z}_{s|i}^{(t+1)} = 2\boldsymbol{w}_s^{(t+1)} - \boldsymbol{z}_{i|s}^{(t+1)} \end{cases} \tag{5.2}$$

In each round, the nodes send $\{z_{i|s}\}$ to the server and the server updates the parameter and then sends $z_{s|i}$ to node $i \in \mathcal{V}$.

Assuming we have eavesdropping and passive nodes (not the server) as cooperative adversaries. Since $\{z_{i|s}, z_{s|i}\}$ are exposed to the adversary, from Equation 5.1 we can obtain the local model $\boldsymbol{w}_i^{(t+1)} = \frac{z_{i|s}^{(t+1)} + z_{s|i}^{(t)}}{2}$. For the subproblems, we have the optimality condition as

$$0 = \nabla f_i \left(\boldsymbol{w}_i^{(t+1)} \right) + \rho(\boldsymbol{w}_i^{(t+1)} - \boldsymbol{z}_{s|i}^{(t)}).$$

Thus gradients $\nabla f_i \left(\boldsymbol{w}_i^{(t+1)} \right)$ of the original problem are revealed.

Approximated Gradient Leakage

In many practical average consensus-based applications, due to the factors like communication overhead, it is not feasible to exchange updates at every round. For example, in FedAvg, $E > 1$ or $B < \infty$ (more than one epoch or using mini-batch SGD). In this case, when weights are exchanged over non-encrypted channels, the leaked information essentially contains multiple updates and updates from multiple batches.

This approach in FL indeed does increase the difficulty for adversaries to launch attacks. However, it is not invulnerable to attacks designed on gradients. For instance, [36] uses a simple strategy to approximate the gradient of each epoch with multiple updates. They divide the leaked information by the number of epochs, thus getting an approximation of the gradient.

5.1.2. DIFFERENCE OF GRADIENTS LEAKAGE

When considering the strategy of inserting subspace perturbation to differential ADMM/PDMM, we assume that the adversary has access to the eavesdropped information $\left\{ \Delta \boldsymbol{z}_{j|i}^{(t)} : t \geq 1, (i,j) \in \mathcal{E} \right\}$. In the ideal case, by observing an infinite number of rounds of messages transmitted channel and exact update in each sub-problem, the upper bound of information leakage for any victim node i is given by the difference of gradients at time t and $t+2$, which is shown in Equation 2.9. In specific inexact update mentioned in Chapter 4, with the knowledge of hyperparameters, such as learning rate α or quadratic approximation factor μ, the difference of gradients can still be inferred with different expressions.

In other cases, inevitable noises are introduced in the estimation of privacy bound, thus only an approximation of the difference of gradients can be obtained.

5.2. ATTACK

The common attack based on gradient information leakage is described in this section, including label inference in subsection 5.2.1 and gradient inversion attack in subsection 5.2.2.

5.2.1. LABEL INFERENCE ATTACK

The private information $(\boldsymbol{x}_i, \boldsymbol{\ell}_i)$ for $i \in \mathcal{V}$ that nodes hold and expect to preserve contains data $\boldsymbol{x}_i$ and label $\boldsymbol{\ell}_i$. The label information is not only sensitive but also can help with the recovery of private data.

Label inference is available with the gradient

It has been shown that the label information $\boldsymbol{\ell}_i$ can be analytically determined from the leaked gradients of the output layer in the early rounds [14], [37]. Consider the local model for multi-classification has L layers and is trained with cross-entropy loss.

Firstly consider the case where the gradient only contains one sample update, i.e. $B = 1$ [14]. Let $\boldsymbol{z}_{i,L} = (z_{i,1}, \ldots, z_{i,C})$ denote the logits in the output layer, the loss function is given by

$$f_i(\boldsymbol{w}_i) = -\log\Big(\frac{e^{z_{i,\ell_i}}}{\sum_j e^{z_{i,j}}}\Big) = \log\big(\textstyle\sum_j e^{z_{i,j}}\big) - z_{i,\ell_i}. \tag{5.3}$$

Let $\boldsymbol{w}_{i,L,c}$ denote the weights in the output layer L corresponding to $z_{i,c}$, i.e. $z_{i,c} = \boldsymbol{w}_{i,L,c}^{\mathsf{T}} \boldsymbol{a}_{i,L-1} + b_{i,L,c}$, where $\boldsymbol{a}_{i,L-1}$ is the activation at layer $L-1$. The gradient of $f_i(\boldsymbol{w}_i)$ with respect to $\boldsymbol{w}_{i,L,c}$ can then be expressed as [14]:

$$\nabla' f_i(\boldsymbol{w}_{i,L,c}) \triangleq \frac{\partial f_i(\boldsymbol{w}_i)}{\partial \boldsymbol{w}_{i,L,c}} = \frac{\partial f_i(\boldsymbol{w}_i)}{\partial z_{i,c}} \frac{\partial z_{i,c}}{\partial \boldsymbol{w}_{i,L,c}} = g_c \boldsymbol{a}_{L-1}, \tag{5.4}$$

and g_c is the gradient of the cross entropy in Equation 5.3 with respect to logit c given by

$$g_c = \frac{e^{z_{i,c}}}{\sum_j e^{z_{i,j}}} - \delta_{c,\ell_i}. \tag{5.5}$$

Hence, $g_c < 0$ for $c = \ell_i$ and $g_c > 0$ otherwise. Since the activation $\boldsymbol{a}_{L-1}$ is independent of the class index c, the ground-truth label ℓ_i can be inferred from the shared gradients since $\nabla'^{\mathsf{T}} f_i(\boldsymbol{w}_{i,L,\ell_i}) \nabla' f_i(\boldsymbol{w}_{i,L,c}) = g_{\ell_i} g_c \|\boldsymbol{a}_{L-1}\|^2 < 0$ for $c \neq \ell_i$ and positive only for $c = \ell_i$.

Similar results hold for the case where $B > 1$ [37]. For $B > 1$, we have the gradient as the average of each sample, thus we modified g_c for each label c as

$$g_c = \frac{1}{B} \sum_{k=1}^{B} \frac{e^{z_{ik,c}}}{\sum_j e^{z_{ik,j}}} - \frac{\lambda_c}{B}, \tag{5.6}$$

where λ_c is the number of occurrences of label c in the batch. In the early rounds, the untrained model has poor performance in predictions, so $\frac{e^{z_{ik,c}}}{\sum_j e^{z_{ik,j}}}$ gets close to zero. Thus in the batch where $B > 1$, the magnitude of the gradient is almost proportional to the number of occurrences of label c, i.e.

$$\nabla' f_i(\boldsymbol{w}_{i,L,c}) \approx -\frac{\lambda_c}{B} \boldsymbol{a}_{L-1} \tag{5.7}$$

Label inference is not available with the difference of gradients

We can analyze the applicability of label inference attacks with the difference of gradients in a similar way. For $B = 1$, we have the representation of the last layer as

$$\begin{aligned}\nabla f(\boldsymbol{w}_{i,L,c}^{(t)}) - \nabla f(\boldsymbol{w}_{i,L,c}^{(t+2)}) &= g_c^{(t)} \boldsymbol{a}_{L-1}^{(t)} - g_c^{(t+2)} \boldsymbol{a}_{L-1}^{(t+2)} \\ &= \Big(\frac{e^{z_{i,c}^{(t)}}}{\sum_j e^{z_{i,j}^{(t)}}} - \delta_{c,\ell_i}\Big)\boldsymbol{a}_{L-1}^{(t)} - \Big(\frac{e^{z_{i,c}^{(t+2)}}}{\sum_j e^{z^{(t+2)} z_{i,j}}} - \delta_{c,\ell_i}\Big)\boldsymbol{a}_{L-1}^{(t+2)} \\ &= \Big(\frac{e^{z_{i,c}^{(t)}}}{\sum_j e^{z_{i,j}^{(t)}}}\boldsymbol{a}_{L-1}^{(t)} - \frac{e^{z_{i,c}^{(t+2)}}}{\sum_j e^{z_{i,j}^{(t+2)}}}\boldsymbol{a}_{L-1}^{(t+2)}\Big) - \delta_{c,\ell_i}\big(\boldsymbol{a}_{L-1}^{(t)} - \boldsymbol{a}_{L-1}^{(t+2)}\big)\end{aligned} \tag{5.8}$$

It can be observed that the difference of gradients does not contain sign information that can be used to distinguish the label. Moreover, for specific cases like logistic regression, the model only contains one layer. In those one-layer model structures, the information from $L-1$ layer is actually the input data $\boldsymbol{x}$, i.e. $\boldsymbol{a}_{L-1}^{(t)} = \boldsymbol{a}_{L-1}^{(t+2)} = \boldsymbol{x}$. Thus we have

$$\begin{aligned}\nabla f(\boldsymbol{w}_{i,L,c}^{(t)}) - \nabla f(\boldsymbol{w}_{i,L,c}^{(t+2)}) &= g_c^{(t)}\boldsymbol{x} - g_c^{(t+2)}\boldsymbol{x} \\ &= \left(\frac{e^{z_{i,c}^{(t)}}}{\sum_j e^{z_{i,j}^{(t)}}} - \frac{e^{z_{i,c}^{(t+2)}}}{\sum_j e^{z_{i,j}^{(t+2)}}}\right)\boldsymbol{x}\end{aligned} \tag{5.9}$$

where the Kronecker-delta is cancelled out and the expression is the same for both $c = \ell_i$ or $c \neq \ell_i$. In an extreme sense, it can be stated in special cases that the difference of gradients does not contain any label information.

5.2.2. GRADIENT INFORMATION INVERSION ATTACK

Gradient Inversion Attack

The gradients of a model often contain some degree of redundancy with respect to the original data. This also brings the possibility to reconstruct the data by solving the inverse problem of computing the gradient. Indeed, non-linear inverse mapping typically does not have an analytical solution in most cases. As a result, finding an explicit, analytical solution is challenging. Instead, it is more common to rely on learning-based approaches to approximate the original data from the gradients. These methods aim to generate dummy data $\boldsymbol{x}_i'$ that has consistent dummy gradients $\nabla f_i(\boldsymbol{w}_i, (\boldsymbol{x}_i', \boldsymbol{\ell}_i'))$ with the real gradients $\nabla f_i(\boldsymbol{w}_i, (\boldsymbol{x}_i, \boldsymbol{\ell}_i))$.

We call this type of attack the gradient inversion attack. The corresponding optimization problem can be expressed as [13]

$$(\boldsymbol{x}_i'^*, \boldsymbol{\ell}_i'^*) = \arg\min_{\boldsymbol{x}_i', \boldsymbol{\ell}_i'} \left\| \nabla f_i(\boldsymbol{w}_i, (\boldsymbol{x}_i', \boldsymbol{\ell}_i')) - \nabla f_i(\boldsymbol{w}_i, (\boldsymbol{x}_i, \boldsymbol{\ell}_i)) \right\|^2. \tag{5.10}$$

In subsection 5.2.1 we mentioned the label can be inferred analytically, thus only optimizing $\boldsymbol{x}_i'$ is feasible. The implementation with $B = 1$ is shown in Algorithm 5.

For the general case of exchanging weights, it is also feasible to obtain information from the approximated gradients [15], [38], like to simulate multiple steps of the local training process [15] or do one-batch approximation [38].

Algorithm 5 Gradient Inversion Attack ($B = 1$)

Initialization of $\boldsymbol{x}'_{ik}$ ▷ Initialization
for $it = 1, ..., n$ **do**
  $\mathbb{D}_i = \left\| \nabla f_i(\boldsymbol{w}_i, (\boldsymbol{x}'_i, \ell_i)) - \nabla f_i(\boldsymbol{w}_i, (\boldsymbol{x}_i, \ell_i)) \right\|^2$
  $\boldsymbol{x}'_i \leftarrow \boldsymbol{x}'_i - \alpha \nabla_{\boldsymbol{x}'_i} \mathbb{D}_i$
end for
return x'_i

Differential Gradient Attack

Similarly, we can design a new attack for the difference of gradients leakage. We transfer the problem to minimizing the following objective

$$(\boldsymbol{x}'^*_i, \boldsymbol{\ell}'^*_i) = \arg\min_{\boldsymbol{x}'_i, \boldsymbol{\ell}'_i} \left\| \nabla f_i\left(\boldsymbol{w}_i^{(t)}, (\boldsymbol{x}'_i, \boldsymbol{\ell}'_i)\right) - \nabla f_i\left(\boldsymbol{w}_i^{(t+2)}, (\boldsymbol{x}'_i, \boldsymbol{\ell}'_i)\right) - \left(\nabla f_i\left(\boldsymbol{w}_i^{(t)}, (\boldsymbol{x}_i, \boldsymbol{\ell}_i)\right) - \nabla f_i\left(\boldsymbol{w}_i^{(t+2)}, (\boldsymbol{x}_i, \boldsymbol{\ell}_i)\right)\right) \right\|^2. \tag{5.11}$$

Since label inference is not applicable, we are looking for ways to improve the quality of the attack. In the experiments, there are difficulties in the co-convergence of data and labels. Therefore, in designing the attack, we employed a traversal of the labels to find the best solution. As shown in Algorithm 6, to avoid being trapped in a local minimum, we consider adding an extra threshold verification. When the iteration ends and the value of the objective function is less than the threshold, the data is considered to be successfully reconstructed, otherwise, the attack is repeated.

Algorithm 6 Differential Gradient Attack ($n_i = 1$)

Initialization of threshold T
for $\ell'_i = 1, ..., C$ **do**
  Initialization of $\boldsymbol{x}'_{ik}$ ▷ Initialization
  for $it = 1, ..., n$ **do**
    $\mathbb{D}_i = \left\| \nabla f_i\left(\boldsymbol{w}_i^{(t)}, (\boldsymbol{x}'_i, \ell'_i)\right) - \nabla f_i\left(\boldsymbol{w}_i^{(t+2)}, (\boldsymbol{x}'_i, \boldsymbol{\ell}'_i)\right) - \left(\nabla f_i\left(\boldsymbol{w}_i^{(t)}, (\boldsymbol{x}_i, \ell_i)\right) - \nabla f_i\left(\boldsymbol{w}_i^{(t+2)}, (\boldsymbol{x}_i, \ell_i)\right)\right) \right\|^2$
    $\boldsymbol{x}'_i \leftarrow \boldsymbol{x}'_i - \alpha \nabla_{\boldsymbol{x}'_i} \mathbb{D}_i$
  end for
  if $\mathbb{D}_i < T$ **then**
    $T = \mathbb{D}_i$
    $\boldsymbol{x}_{opt} = \boldsymbol{x}'_i, \ell_{opt} = \ell'_i$ ▷ Validation
  end if
end for
return $\boldsymbol{x}_{opt}, \ell_{opt}$

5.3. COMPARISON

In this section, we compare the information obtained by adversaries in average consensus-based and optimization-based FL. In average consensus-based FL, attacks that aim to obtain gradients or approximate gradients can occur on any target and at any time during training. Additionally, label inference can be used to accelerate and improve the quality of reconstructed information and also reduce the search space. On the other hand, optimization-based FL methods require adversaries to obtain and store channel information over successive rounds. After training stops, adversaries perform backward computations to obtain the difference of gradients and complete the attack. Label inference is not applicable to the difference of gradients in this case.

Table 5.1: Information obtained by adversary from average consensus-based and optimization-based FL

	Average consensus-based	Optimization-based
Attack requirement	Messages in one round	Messages in enough rounds
Information leakage	Gradient	Difference of gradients
Label inference attack	Available	Unavailable

We can consider implementing the average consensus-based approach differentially like differential PDMM. For example, in the first communication round, the gradients are transmitted through encrypted channels. Subsequent communications only transmit the difference of the current round's gradient and the previous round's gradient. In this approach, the adversary can obtain $\nabla f_i(\boldsymbol{w}_i^{(t)}) - \nabla f_i(\boldsymbol{w}_i^{(t-1)})$ within one observation round. However, as the model almost converges, $\nabla f_i(\boldsymbol{w}_i^{(t)}) \to 0$ for all $i \in \mathcal{N}$. Consequently, the gradient can be approximated by backward calculation from a sufficient number of successive observations, similar to how $\boldsymbol{w}_i^{(t)}$ is obtained in PDMM.

5.4. EXPERIMENTS

In this section, we demonstrate the data reconstruction performance of differential gradient attacks in decentralized networks. We will provide examples of gradient leakage attacks in FedAvg and show the comparisons in the next chapter.

Differential Gradient Attack

We first consider the simple case of logistic regression in Section 4.3 with $n_i = 1$ because it is a special example that we do not need to use the learning-based approach to implement the attack.

Assuming training continues until complete convergence and the adversary observes and stores all channel messages throughout the entire duration, the adversary can ideally recover each local model $\boldsymbol{w}_i^{(t)}$. We can use the privacy bound derived from the optimal condition, i.e. Equation 2.13. Combine the gradients in Equation 3.2 and Equation 3.3, we have

$$\begin{aligned} \frac{\partial f_i}{\partial \boldsymbol{w}_i}^{(t)} - \frac{\partial f_i}{\partial \boldsymbol{w}_i}^{(t+2)} &= \sum_{k=1}^{n_i} \left(y_{ik}^{(t)} - y_{ik}^{(t+2)} \right) \boldsymbol{x}_{ik} \\ &= \rho d_i \left(\boldsymbol{w}_i^{(t)} + \boldsymbol{w}_i^{(t+2)} \right) - 2\rho \sum_{j \in \mathcal{N}_i} \boldsymbol{w}_j^{(t+1)}, \end{aligned} \tag{5.12}$$

$$\begin{aligned}\frac{\partial f_i^{(t)}}{\partial b_i} - \frac{\partial f_i^{(t+2)}}{\partial b_i} &= \sum_{k=1}^{n_i}\left(y_{ik}^{(t)} - y_{ik}^{(t+2)}\right) \\ &= \rho d_i\left(b_i^{(t)} + b_i^{(t+2)}\right) - 2\rho \sum_{j\in\mathcal{N}_i} b_j^{(t+1)}.\end{aligned} \tag{5.13}$$

So for $n_i = 1$, (5.12) is just a scaled version of $\boldsymbol{x}_{ik}$ where the scaling is given by (5.13). Hence, we can analytically compute $\boldsymbol{x}_{ik} = \frac{\rho d_i\left(\boldsymbol{w}_i^{(t)} + \boldsymbol{w}_i^{(t+2)}\right) - 2\rho\sum_{j\in\mathcal{N}_i}\boldsymbol{w}_j^{(t+1)}}{\rho d_i\left(b_i^{(t)} + b_i^{(t+2)}\right) - 2\rho\sum_{j\in\mathcal{N}_i} b_j^{(t+1)}}$.

Here we use the iterative method with 1000 iterations to almost get the exact solution in weight update, the reconstruction is shown in Figure 5.1.

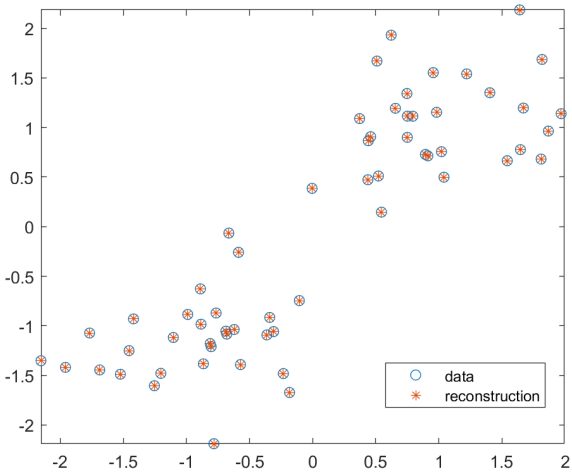

Figure 5.1: Reconstruction of the private data from the difference of gradients

It can be observed that private data is recovered successfully, but the label information cannot be inferred from it.

We then test the differential gradient attack in a general problem. We generated an RGG with $N = 100$ nodes. Two-layer perceptrons were constructed for each node using Pytorch and we used MNIST [39] as the dataset. PDMM was used with constant $\rho = 0.4$. The differential gradient attacks were implemented using the L-BFGS algorithm[40] where the number of iterations was fixed to 30.

Figure 5.2 and Figure 5.3 show the performance of the attack in multi-layer perceptron. The recovered images are the results corresponding to the minimum value of the optimization problems obtained after traversing the label.

For the case $n_i = 2$ in Figure 5.3, the order of the reconstructed images will be different, but the attack is still effective.

Label Inference

Since the label cannot be analytically computed in the difference of gradients, in the implementation we traverse all possible labels to find out the optimal solution. The above

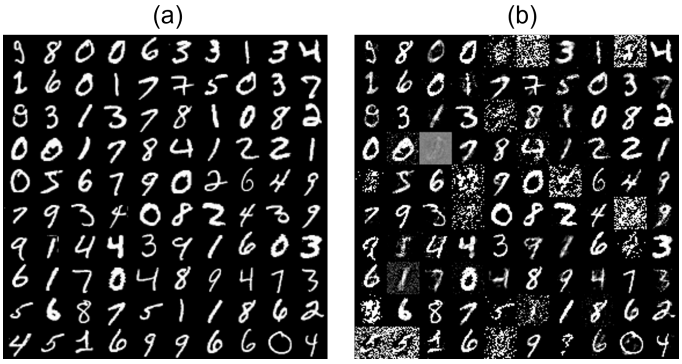

Figure 5.2: Attack for multi-layer perceptron, $n_i = 1$. (a) The ground truth. (b) The reconstruction.

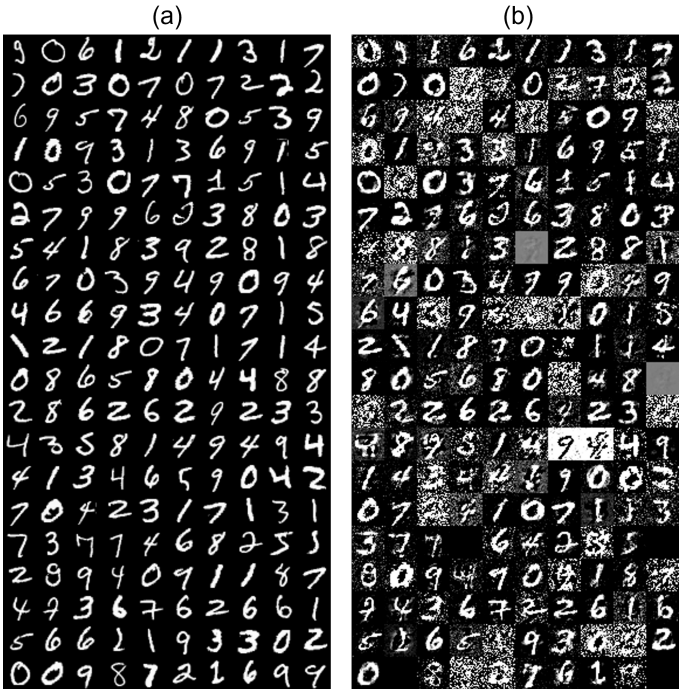

Figure 5.3: Attack for multi-layer perceptron, $n_i = 2$. (a) The ground truth. (b) The reconstruction.

results with $n_i = 1$ and $n_i = 2$ are the kept optimal solution. Since the wrong label leads to the wrong solution, successful reconstruction of the image also means that the label is successfully found. As can be seen, the attack is not available when there are duplicate labels in the batch. This is consistent with the case of gradient-based attacks. We will show other results in the next chapter.

6

DEFENSE STRATEGY

In this chapter, we present defense strategies against differential gradient attacks discussed in Chapter 5. Section 6.1 describes several types of introduced errors and perturbations, which can be used as defense mechanisms. Section 6.2 outlines the corresponding defense strategies. Additionally, Section 6.3 showcases experimental results demonstrating the effectiveness of some of these strategies.

6.1. NOISE AND PERTURBATIONS

Convergence Error

As mentioned earlier, one of the prerequisites for the adversary to make a precise inference on the privacy bound is to have the knowledge of the fully converged global model $\boldsymbol{w}^*$. However, in practical applications, the training process often stops after a finite number of communication rounds, say $t_{\max}$. As a consequence, the adversary only knows $\boldsymbol{w}^*$ up to an error and can therefore only estimate the individual $\boldsymbol{w}_i^{(t)}$s up to a certain accuracy.

To quantify this error, let $\boldsymbol{\epsilon}_i^{(t_{\max})} = \hat{\boldsymbol{w}}_i^{(t_{\max})} - \boldsymbol{w}_i^{(t_{\max})}$ be the adversary's estimation error in $\boldsymbol{w}_i^{(t_{\max})}$. Since the adversary has knowledge of $\boldsymbol{w}_i^{(t+1)} - \boldsymbol{w}_i^{(t)}$ at every iteration, we have

$$\begin{aligned}\hat{\boldsymbol{w}}_i^{(t)} &= \hat{\boldsymbol{w}}_i^{(t_{\max})} - \sum_{\tau=t}^{t_{\max}-1} \left(\boldsymbol{w}_i^{(\tau+1)} - \boldsymbol{w}_i^{(\tau)}\right) \\ &= \boldsymbol{w}_i^{(t_{\max})} - \sum_{\tau=t}^{t_{\max}-1} \left(\boldsymbol{w}_i^{(\tau+1)} - \boldsymbol{w}_i^{(\tau)}\right) + \boldsymbol{\epsilon}_i^{(t_{\max})} \\ &= \boldsymbol{w}_i^{(t)} + \boldsymbol{\epsilon}_i^{(t_{\max})},\end{aligned} \tag{6.1}$$

for $1 \leq t \leq t_{\max}$. So the adversary can only estimate Equation 2.13 up to a certain accuracy determined by $\boldsymbol{\epsilon}_i^{(t_{\max})}$ at any round t.

More specifically, the introduction of the error in the difference of gradients depends on the degree of global convergence at $t_{\max}$. Assume the network has a corrupted node

$c \in \mathcal{V}$ and adversary uses $\boldsymbol{w}_c^{(t_{\max})}$ as the approximation of $\boldsymbol{w}^*$, that is $\hat{\boldsymbol{w}}_i^{(t_{\max})} = \boldsymbol{w}_c^{(t_{\max})}$ for $\forall i \in \mathcal{V}$. Thus we have

$$\begin{aligned}\nabla f_i\left(\boldsymbol{w}_i^{(t)}\right) - \nabla f_i\left(\boldsymbol{w}_i^{(t+2)}\right) &= \rho d_i\left(\hat{\boldsymbol{w}}_i^{(t)} + \hat{\boldsymbol{w}}_i^{(t+2)}\right) - 2\rho \sum_{j\in\mathcal{N}_i} \hat{\boldsymbol{w}}_j^{(t+1)} \\ &= \rho d_i\left(\boldsymbol{w}_i^{(t)} + \boldsymbol{w}_i^{(t+2)}\right) - 2\rho \sum_{j\in\mathcal{N}_i} \boldsymbol{w}_j^{(t+1)} + 2\rho d_i \boldsymbol{\epsilon}_i^{(t_{max})} - 2\rho \sum_{j\in\mathcal{N}_i} \boldsymbol{\epsilon}_j^{(t_{max})} \\ &= \rho d_i\left(\boldsymbol{w}_i^{(t)} + \boldsymbol{w}_i^{(t+2)}\right) - 2\rho \sum_{j\in\mathcal{N}_i} \boldsymbol{w}_j^{(t+1)} + 2\rho \sum_{j\in\mathcal{N}_i} (\boldsymbol{w}_i^{(t_{max})} - \boldsymbol{w}_j^{(t_{max})}).\end{aligned} \tag{6.2}$$

The term $2\rho\sum_{j\in\mathcal{N}_i}(\boldsymbol{w}_i^{(t_{max})} - \boldsymbol{w}_j^{(t_{max})})$ is the error introduced on RHS, which only depends on the distance of model $\boldsymbol{w}_i^{(t_{max})}$ and the averaged model of i-th node's neighbors at t_{max}. Thus when $\boldsymbol{w}_i^{(t_{max})}$ is closer to the average of its neighbors, less error is introduced.

It should be noted that in a few cases, such as the example of logistic regression with $n_i = 1$ in Section 5.4, the attack is not affected by the changes in the LHS, i.e. $\nabla f_i\left(\boldsymbol{w}_i^{(t)}, (\boldsymbol{x}_i', \boldsymbol{\ell}_i')\right) - \nabla f_i\left(\boldsymbol{w}_i^{(t+2)}, (\boldsymbol{x}_i', \boldsymbol{\ell}_i')\right)$ because the changes in both the numerator and denominator are the same in the division. In such special cases, the specific value of $\boldsymbol{w}_c^{(t_{\max})}$ is not important at all, since the error term only depends on the level of convergence. However, when using the general optimization approach, we need the model $\boldsymbol{w}_i^{(t)}$ and $\boldsymbol{w}_i^{(t+2)}$ to obtain the dummy gradient. Therefore, the knowledge of $\hat{\boldsymbol{w}}_i^{(t)}$ and $\hat{\boldsymbol{w}}_i^{(t+2)}$ introduces errors in $\nabla f_i\left(\hat{\boldsymbol{w}}_i^{(t)}, (\boldsymbol{x}_i', \boldsymbol{\ell}_i')\right) - \nabla f_i\left(\hat{\boldsymbol{w}}_i^{(t+2)}, (\boldsymbol{x}_i', \boldsymbol{\ell}_i')\right)$ as well.

Training Error

As mentioned in Chapter 4, in some applications, such as training neural networks, Equation 2.6 is only solved approximately. That is, at every communication round t, the optimality condition in Equation 2.11 holds approximately, i.e.

$$\nabla f_i(\boldsymbol{w}_i^{(t+1)}) + \sum_{j\in\mathcal{N}_i} \boldsymbol{B}_{i|j}^\top \boldsymbol{z}_{i|j}^{(t)} + \rho d_i \boldsymbol{w}_i^{(t+1)} = \boldsymbol{\varepsilon}_i^{(t+1)}, \tag{6.3}$$

where $\boldsymbol{\varepsilon}_i$ denotes an approximation error. So the RHS of (2.13) in this case gets an additional term $\boldsymbol{\varepsilon}_i^{(t)} - \boldsymbol{\varepsilon}_i^{(t+2)}$. The bound is modified as

$$\nabla f_i\left(\boldsymbol{w}_i^{(t)}\right) - \nabla f_i\left(\boldsymbol{w}_i^{(t+2)}\right) = \rho d_i\left(\boldsymbol{w}_i^{(t)} + \boldsymbol{w}_i^{(t+2)}\right) - 2\rho \sum_{j\in\mathcal{N}_i} \boldsymbol{w}_j^{(t+1)} + \boldsymbol{\varepsilon}_i^{(t)} - \boldsymbol{\varepsilon}_i^{(t+2)}. \tag{6.4}$$

The training error only affects the update process of $\boldsymbol{w}_i^{(t+1)}$, for the adversary, it is possible to recover the exact $\boldsymbol{w}_i^{(t)}$ in each moment.

Transmission Error

Transmission error refers to message inaccuracies that occur during the transmission due to factors such as noise, and interference in the communication channels. We denote it as

$$\Delta\hat{\boldsymbol{z}}_{j|i}^{(t)} = \Delta\boldsymbol{z}_{j|i}^{(t)} + \boldsymbol{n}_{j|i}^{(t)}. \tag{6.5}$$

Since Equation 2.10 still hold, we have

$$\Delta\hat{\boldsymbol{z}}_{j|i}^{(t+1)} - \Delta\hat{\boldsymbol{z}}_{i|j}^{(t)} = 2\rho\boldsymbol{B}_{i|j}(\boldsymbol{w}_i^{(t+1)} - \boldsymbol{w}_i^{(t)}) + (\boldsymbol{n}_{j|i}^{(t+1)} - \boldsymbol{n}_{i|j}^{(t)}). \tag{6.6}$$

The transmission error is introduced when approximate $\hat{\boldsymbol{w}}_i^{(t)}$, i.e.

$$\begin{aligned}\hat{\boldsymbol{w}}_i^{(t)} &= \boldsymbol{w}_i^{(t_{\max})} - \sum_{\tau=t}^{t_{\max}-1}\left(\boldsymbol{w}_i^{(\tau+1)} - \boldsymbol{w}_i^{(\tau)}\right) + \frac{1}{2\rho}\boldsymbol{B}_{i|j}\sum_{\tau=t}^{t_{\max}-1}\left(\boldsymbol{n}_{j|i}^{(\tau+1)} - \boldsymbol{n}_{i|j}^{(\tau)}\right) \\ &= \boldsymbol{w}_i^{(t)} + \frac{1}{2\rho}\boldsymbol{B}_{i|j}\sum_{\tau=t}^{t_{\max}-1}\left(\boldsymbol{n}_{j|i}^{(\tau+1)} - \boldsymbol{n}_{i|j}^{(\tau)}\right).\end{aligned} \tag{6.7}$$

Thus we have

$$\begin{aligned}\nabla f_i\left(\boldsymbol{w}_i^{(t)}\right) - \nabla f_i\left(\boldsymbol{w}_i^{(t+2)}\right) &= \rho d_i\left(\hat{\boldsymbol{w}}_i^{(t)} + \hat{\boldsymbol{w}}_i^{(t+2)}\right) - 2\rho\sum_{j\in\mathcal{N}_i}\hat{\boldsymbol{w}}_j^{(t+1)} \\ &= \rho\sum_{j\in\mathcal{N}_i}\left(\hat{\boldsymbol{w}}_i^{(t)} - \hat{\boldsymbol{w}}_j^{(t+1)}\right) - \rho\sum_{j\in\mathcal{N}_i}\left(\hat{\boldsymbol{w}}_j^{(t+1)} - \hat{\boldsymbol{w}}_i^{(t+2)}\right) \\ &= \rho d_i\left(\boldsymbol{w}_i^{(t)} + \boldsymbol{w}_i^{(t+2)}\right) - 2\rho\sum_{j\in\mathcal{N}_i}\boldsymbol{w}_j^{(t+1)} \\ &+ \frac{1}{2}\sum_{j\in\mathcal{N}_i}\left(\boldsymbol{B}_{i|j}\sum_{\tau=t}^{t_{\max}-1}\left(\boldsymbol{n}_{j|i}^{(\tau+1)} - \boldsymbol{n}_{i|j}^{(\tau)}\right) - \boldsymbol{B}_{j|i}\sum_{\tau=t+1}^{t_{\max}-1}\left(\boldsymbol{n}_{i|j}^{(\tau+1)} - \boldsymbol{n}_{j|i}^{(\tau)}\right)\right) \\ &- \frac{1}{2}\sum_{j\in\mathcal{N}_i}\left(\boldsymbol{B}_{j|i}\sum_{\tau=t+1}^{t_{\max}-1}\left(\boldsymbol{n}_{i|j}^{(\tau+1)} - \boldsymbol{n}_{j|i}^{(\tau)}\right) - \boldsymbol{B}_{i|j}\sum_{\tau=t+2}^{t_{\max}-1}\left(\boldsymbol{n}_{j|i}^{(\tau+1)} - \boldsymbol{n}_{i|j}^{(\tau)}\right)\right) \\ &= \rho d_i\left(\boldsymbol{w}_i^{(t)} + \boldsymbol{w}_i^{(t+2)}\right) - 2\rho\sum_{j\in\mathcal{N}_i}\boldsymbol{w}_j^{(t+1)} \\ &+ \frac{1}{2}\sum_{j\in\mathcal{N}_i}\boldsymbol{B}_{i|j}\left(\boldsymbol{n}_{j|i}^{(t_{\max})} + \boldsymbol{n}_{i|j}^{(t_{\max})} - \boldsymbol{n}_{i|j}^{(t)} - \boldsymbol{n}_{i|j}^{(t+1)}\right) \\ &- \frac{1}{2}\sum_{j\in\mathcal{N}_i}\boldsymbol{B}_{j|i}\left(\boldsymbol{n}_{i|j}^{(t_{\max})} + \boldsymbol{n}_{j|i}^{(t_{\max})} - \boldsymbol{n}_{j|i}^{(t+1)} - \boldsymbol{n}_{j|i}^{(t+2)}\right) \\ &= \rho d_i\left(\boldsymbol{w}_i^{(t)} + \boldsymbol{w}_i^{(t+2)}\right) - 2\rho\sum_{j\in\mathcal{N}_i}\boldsymbol{w}_j^{(t+1)} \\ &+ \sum_{j\in\mathcal{N}_i}\boldsymbol{B}_{i|j}\left(\boldsymbol{n}_{j|i}^{(t_{\max})} + \boldsymbol{n}_{i|j}^{(t_{\max})}\right) - \frac{1}{2}\sum_{j\in\mathcal{N}_i}\boldsymbol{B}_{i|j}\left(\boldsymbol{n}_{i|j}^{(t)} + \boldsymbol{n}_{i|j}^{(t+1)} + \boldsymbol{n}_{j|i}^{(t+1)} + \boldsymbol{n}_{j|i}^{(t+2)}\right)\end{aligned} \tag{6.8}$$

The training error is also introduced in the dummy gradient $\nabla f_i\left(\hat{\boldsymbol{w}}_i^{(t)}, (\boldsymbol{x}_i', \boldsymbol{\ell}_i')\right) - \nabla f_i\left(\hat{\boldsymbol{w}}_i^{(t+2)}, (\boldsymbol{x}_i', \boldsymbol{\ell}_i')\right)$ as the case of convergence error.

6.2. DEFENSE STRATEGIES

Early Stopping

With the high convergence of the global model, $\nabla f_i\left(\boldsymbol{w}_i^{(t)}\right) - \nabla f_i\left(\boldsymbol{w}_i^{(t+2)}\right) \to 0$. Since the error term in Equation 6.2 is fixed for $\forall\, t \in [1, t_{\max}]$, for the adversary, it is necessary to store the observation start from the early stage to reduce relative error. To some extent, it increases the computational and spatial costs of the attacks. Furthermore, early stopping serves two purposes. On one hand, it helps prevent model overfitting; on the other hand, it can also increase $\boldsymbol{\epsilon}_i^{(t_{\max})}$, thereby enhancing the extent of privacy preservation.

Inexact Update

Since the successful implementation of the attack is based on the subproblem reaching a known certain point (the optimal point or the point that a single step reached), the inexact approximation of subproblems can form a natural defence. To achieve privacy preservation, we can implement a strategy of intentionally blurring the adversary's approximations of the privacy bounds. In Chapter 4 we mentioned two inexact update methods for distributed machine learning and gave the convergence guarantee with finitely summable errors. We can divide the inexact update into two cases. When $k_{\max} = 1$ or $k_{\max} = \infty$ or use QA-PDMM, the adversary can still infer the privacy bound as long as they hold the knowledge of hyperparameter, i.e. α and μ. Thus nodes should treat their own hyperparameters as private information as well, to avoid precise attacks by adversaries. Another case is $k_{\max} > 1$, its intermediate state in training is unknowable to the adversary and therefore the introduction of errors is inevitable.

Quantization

Adaptive transmission quantization can be considered as introducing additive noise with a uniform distribution in the channel. Similar to inexact updates, this noise effectively blurs the approximation of privacy bound, adding a level of uncertainty to the adversary's knowledge. At the same time, quantization leads to lower bandwidth requirements and decreased communication costs.

Larger Batchsize

When the node has a larger local dataset , the difficulty of attacks increases significantly. More samples mean more information is contained in the difference of gradients, making the attack harder to converge. In fact, this has a certain similarity to the batchsize in the stochastic gradient descent. If $B = 1$, then its gradient update direction is the optimal direction for that sample; if B is larger than the direction is the average of samples, which obviously increases the difficulty of gradient inversion. Indeed, current gradient inversion attacks and label inference mainly focus on scenarios with small batches.

6.3. EXPERIMENTS

Early Stopping and Quantization

Figure 6.1 investigates the effect of early stopping of the training after a finite number of rounds $t_{\max}$. Here we used the example in Section 4.3 and PDMM updates with $k_{\max} = 1$ so the privacy bound can be modified as Equation 4.3 to ensure the error in the reconstruction is solely due to an inaccurate approximation of $\boldsymbol{w}_i^{(t)}$.

The reconstruction error is defined as the average Euclidean distance between the reconstructed samples $\hat{\boldsymbol{x}}_{ik}$ and the original data samples $\boldsymbol{x}_{ik}$ given by

$$\frac{1}{N}\sum_{i=1}^{N} \|\hat{\boldsymbol{x}}_{ik} - \boldsymbol{x}_{ik}\|_2. \tag{6.9}$$

We assume there is one corrupt node, say node j, in the network and used $\hat{\boldsymbol{w}}_i^{(t_{\max})} = \boldsymbol{w}_j^{(t_{\max})}$ for all honest nodes i as the approximated global-converged model.

We can see that for the green curve, as expected, the reconstruction error will decrease as $t_{\max}$ increases without quantization, i.e., the longer the adversary waits, the higher the reconstruction accuracy will be.

The FedAvg uses the gradient at $t = t_{\max}$ to do the reconstruction, i.e. $\boldsymbol{x}_{ik} = \frac{\frac{\partial f_i}{\partial \boldsymbol{w}_i}}{\frac{\partial f_i}{\partial b_i}}$. With the FedAvg algorithm, the reconstructed error does not depend on $t_{\max}$ so the adversary can launch the attack at any time and no previous observation is needed. Also, the reconstruction accuracy is consistently better than the accuracy for decentralized FL.

When we consider the approach of quantization, the noise is introduced in transmissions. We set fixed cell width Δ as the precision of the quantizer and test the impact on the reconstruction in Figure 6.1. The results show that as the variation in $\boldsymbol{z}_{i|j}$ tends to level off with convergence, the quantizer can effectively prevent the improvement of reconstruction accuracy.

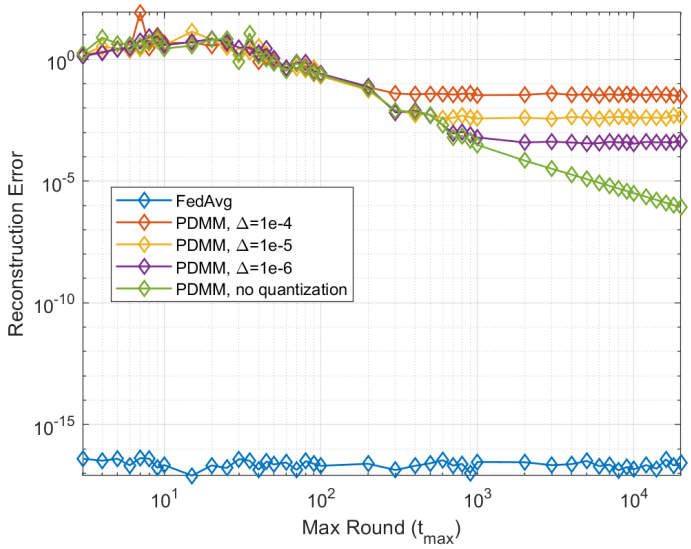

Figure 6.1: Reconstruction error of as a function of $t_{\max}$ (logistic regression)

Inexact Update

When the system is trained using inexact updates and the adversary has unknown knowledge about the level of inaccuracy, the adversary can only attempt to approximate the upper bound of privacy inferred from the optimal conditions, as indicated by Equation 2.13. We assume that the adversary has access to the accurate value of $\boldsymbol{w}_i^{(t)}$, i.e., $t_{\max} \to \infty$. As shown in Figure 6.2, a lower number of iterations can increase the level of privacy preservation. For a certain error curve, it rises a bit as the used model gradually converges. This may be due to the fact that the gradient difference gradually converges to zero as the model converges. Also, we fix the learning rate and the number of iterations to solve the subproblems, so the fluctuations caused by the introduced errors are higher compared to the pre-training period.

Similar results are shown in Figure 6.3 of MLP with MNIST and CIFAR-10 [41] datasets. This time we alternate the data into images so the performance display is visualized. The noise reacts to the image resulting in a degradation of the image quality.

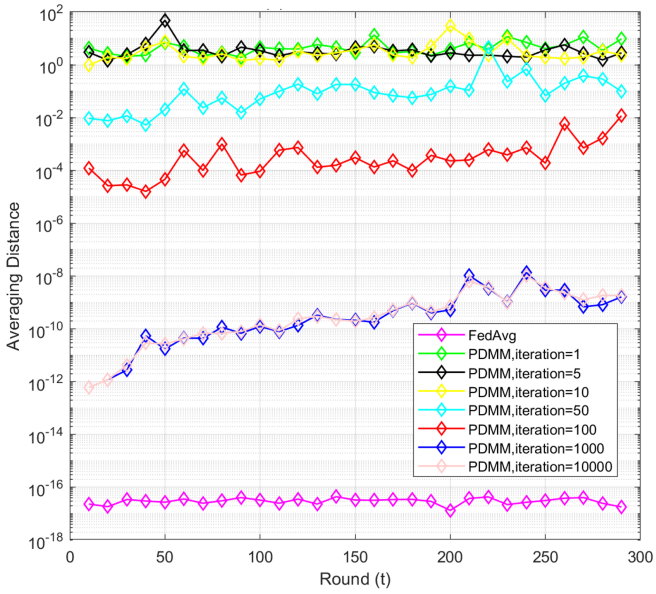

Figure 6.2: Reconstruction error as a function of $k_{\max}$ (logistic regression)

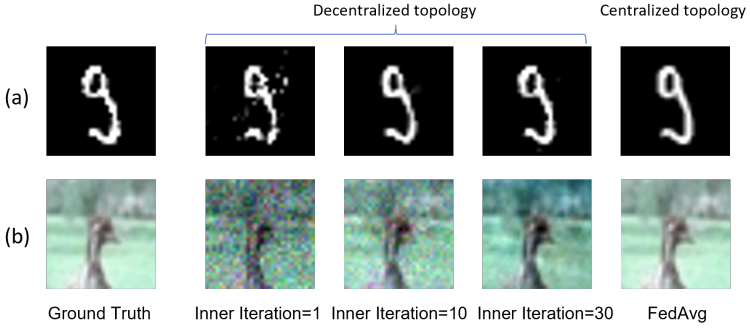

Figure 6.3: Reconstruction error as a function of $k_{\max}$ (MLP). (a) The MNIST dataset. (b) The CIFAR-10 dataset.

Local Dataset Size

Figure 6.4 shows reconstruction results for the MNIST dataset (top) and the CIFAR-10 dataset (bottom) for different values of n_i. As can be seen from the figure, the inference of labels is sometimes wrong with increased n_i. As an example, for the MNIST dataset and $n_i = 4$, the digit 0 is reconstructed but it was not included in the training set.

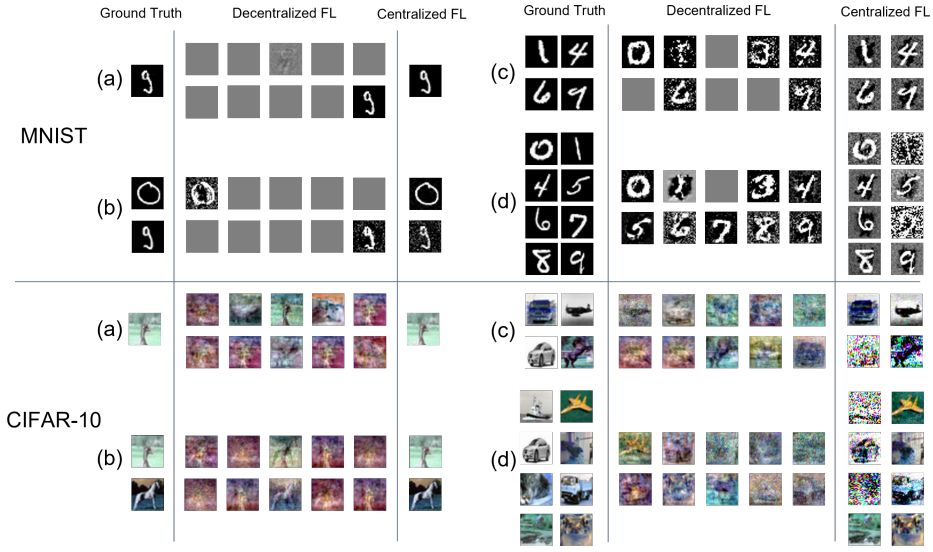

Figure 6.4: Reconstructed inputs for centralized and decentralized FL using the datasets MNIST (top) and CIFAR-10 (bottom) for different batch size $n_i = 1, 2, 4, 8$ ((a)-(d), respectively).

7

Conclusion and Future Work

7.1. Conclusion

In this report, we explored privacy concerns in optimization-based decentralized FL. Optimization-based decentralized FL brings the idea that implicitly imposes constraints in collaborative training, thus combining the steps of local training and aggregation. This approach was considered in the previous literature to be friendly to heterogeneous (non-IID) data. Here we also present its superiority in privacy preservation.

We first extended the upper bound of privacy leakage in distributed optimization to encompass the framework of decentralized federated learning. We show that the upper bound of leaked information obtained by passive adversaries is essentially the difference of gradients in successive periods and can be used to recover private data.

Based on that, we compared it to the well-known gradient leakage attacks prevalent in centralized topologies. Average-consensus based FL leaks the gradient in the channel directly and the adversary can launch an attack at any moment; on the contrary, leaking the difference of gradients reduces the risk and for the adversary, the attack requires monitoring the global data transmission until all nodes are fully converged or almost converged. This implies a much higher computational and storage overhead.

The results show that the optimization-based decentralized FL outperforms the average consensus-based FL in the centralized topology from the privacy-preserving point of view. Label inference is not available in decentralized FL and the reconstruction of the gradient inversion attack is poor.

We also discussed the effect of the perturbations introduced during training for adversaries. We find that these perturbations, without compromising accuracy, effectively interfere with the attack. In this case, the corresponding defense strategies, e.g. early stopping, inexact update and quantization are the preferred means of protecting privacy.

7.2. Future Work

We list some directions that are valuable for further research.

Asynchronous ADMM/PDMM

In this report, the privacy analysis focuses on the synchronous ADMM/PDMM framework. Asynchronous ADMM/PDMM is a more flexible approach since nodes do not need to have a global clock. For asynchronous ADMM/PDMM, intuitively, the upper bound of privacy leakage is also applicable. Because the information adversary eavesdrops include messages in all channels in the entire duration. Thus adversary has the knowledge of each local clock. Currently, we have not verified this in the experiments.

Start Time t_{start} for Adversary

As the model converges, the variation of $\boldsymbol{z}$ becomes smaller and the difference of gradients gradually approaches 0. It implies that the amount of effective information available to the adversary in the late training period is less. One unexplored issue is the effect of the start time of adversary listens on the data reconstruction. We know that the end time of eavesdropping introduces a fixed reconstruction error $2\rho d_i \boldsymbol{\epsilon}_i^{(t_{max})} - 2\rho \sum_{j\in\mathcal{N}_i} \boldsymbol{\epsilon}_j^{(t_{max})}$, and the reconstruction error acts on the difference of gradients at the time t_{start} and $t_{start}+2$. Therefore, it is intuitive that the earlier the eavesdropping is implemented, the smaller the relative error introduced and the better the adversary can obtain the attack. This still requires more experiments to verify this.

Quantitative Evaluation of Defense Strategy

Chapter 6 proposed several defense strategies and gave validation for the efficiency from the aspect that the quality of reconstructed data is worse than the case that is unprotected or in the centralized topology. Quantifying the effectiveness of privacy preservation is a topic that requires further research. The convergence error and training error in the subproblem is related to (sub)linear convergence rates, while the adaptive quantization error follows a uniform distribution. In addition to affecting the adversary's computation of the true gradient difference, convergence error and transmission error also impact the computation of the pseudo-gradient, since they directly affect $\hat{\boldsymbol{w}}_i^{(t)}$.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, and S. Hampson, "Communication-efficient learning of deep networks from decentralized data," p. 10,

[2] C. Wallac and S. Giraldo, *Federated learning, machine learning, decentralized data*, https://blog.cloudera.com/federated-learning-machine-learning-decentralized-data/, Accessed: 2020-12-8.

[3] P. H. Jin, Q. Yuan, F. Iandola, and K. Keutzer, "How to scale distributed deep learning?" *arXiv preprint arXiv:1611.04581*, 2016.

[4] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," *Advances in neural information processing systems*, vol. 30, 2017.

[5] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu, "D^2: Decentralized training over decentralized data," in *International Conference on Machine Learning*, PMLR, 2018, pp. 4848–4856.

[6] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," *arXiv preprint arXiv:1908.07782*, 2019.

[7] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Püschel, "D-admm: A communication-efficient distributed algorithm for separable optimization," *IEEE Transactions on Signal processing*, vol. 61, no. 10, pp. 2718–2723, 2013.

[8] W. Li, Y. Liu, Z. Tian, and Q. Ling, "Communication-censored linearized admm for decentralized consensus optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 6, pp. 18–34, 2019.

[9] H. Chen, Y. Ye, M. Xiao, M. Skoglund, and H. V. Poor, "Coded stochastic admm for decentralized consensus optimization with edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5360–5373, 2021.

[10] G. Zhang and R. Heusdens, "Distributed optimization using the primal-dual method of multipliers," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 1, pp. 173–187, 2017.

[11] T. W. Sherson, R. Heusdens, and W. B. Kleijn, "Derivation and analysis of the primal-dual method of multipliers based on monotone operator theory," *IEEE transactions on signal and information processing over networks*, vol. 5, no. 2, pp. 334–347, 2018.

[12] K. Niwa, N. Harada, G. Zhang, and W. B. Kleijn, "Edge-consensus learning: Deep learning on p2p networks with nonhomogeneous data," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 668–678.

[13] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in neural information processing systems*, vol. 32, 2019.

[14] B. Zhao, K. R. Mopuri, and H. Bilen, "Idlg: Improved deep leakage from gradients," *arXiv preprint arXiv:2001.02610*, 2020.

[15] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 937–16 947, 2020.

[16] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via gradinversion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 337–16 346.

[17] F. Boenisch, A. Dziedzic, R. Schuster, A. S. Shamsabadi, I. Shumailov, and N. Papernot, "When the curious abandon honesty: Federated learning is not private," *arXiv preprint arXiv:2112.02918*, 2021.

[18] J. Geng, Y. Mou, Q. Li, *et al.*, "Improved gradient inversion attacks and defenses in federated learning," *IEEE Transactions on Big Data*, 2023.

[19] L. Fowl, J. Geiping, W. Czaja, M. Goldblum, and T. Goldstein, "Robbing the fed: Directly obtaining private data in federated learning with modified models," *arXiv preprint arXiv:2110.13057*, 2021.

[20] J. Zhu and M. Blaschko, "R-gap: Recursive gradient attack on privacy," *arXiv preprint arXiv:2010.07733*, 2020.

[21] W. Wei, L. Liu, M. Loper, *et al.*, "A framework for evaluating gradient leakage attacks in federated learning," *arXiv preprint arXiv:2004.10397*, 2020.

[22] H. Yang, M. Ge, K. Xiang, and J. Li, "Using highly compressed gradients in federated learning for data reconstruction attacks," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 818–830, 2022.

[23] D. Pasquini, M. Raynal, and C. Troncoso, "On the privacy of decentralized machine learning," *arXiv preprint arXiv:2205.08443*, 2022.

[24] E. K. Ryu and S. Boyd, "Primer on monotone operator methods," *Appl. comput. math*, vol. 15, no. 1, pp. 3–43, 2016.

[25] T. W. Sherson, R. Heusdens, and W. B. Kleijn, "Derivation and analysis of the primal-dual method of multipliers based on monotone operator theory," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 2, pp. 334–347, Jun. 2019, ISSN: 2373-776X, 2373-7778. DOI: 10.1109/TSIPN.2018.2876754. [Online]. Available: https://ieeexplore.ieee.org/document/8496887/ (visited on 05/23/2022).

[26] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[27] Q. Li, "Communication efficient privacy-preserving distributed optimization using adaptive differential quantization," *Signal Processing*, 2022.

[28] Q. Li, R. Heusdens, and M. G. Christensen, "Privacy-preserving distributed optimization via subspace perturbation: A general framework," *IEEE Transactions on Signal Processing*, vol. 68, pp. 5983–5996, 2020.

[29] J. Dall and M. Christensen, "Random geometric graphs," *Physical Review E*, vol. 66, no. 1, p. 016 121, Jul. 24, 2002, ISSN: 1063-651X, 1095-3787. DOI: 10.1103/PhysRevE.66.016121. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.66.016121 (visited on 09/17/2022).

[30] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE transactions on information theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[31] J. A. Jonkman, T. Sherson, and R. Heusdens, "Quantisation effects in distributed optimisation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 3649–3653.

[32] M. O'Connor, G. Zhang, W. B. Kleijn, and T. D. Abhayapala, "Function splitting and quadratic approximation of the primal-dual method of multipliers for distributed optimization over graphs," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 4, pp. 656–666, 2018.

[33] G. Zhang, K. Niwa, and W. B. Kleijn, "Revisiting the primal-dual method of multipliers for optimisation over centralised networks," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 8, pp. 228–243, 2022.

[34] R. Pathak and M. J. Wainwright, "Fedsplit: An algorithmic framework for fast federated optimization," *Advances in neural information processing systems*, vol. 33, pp. 7057–7066, 2020.

[35] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International Conference on Machine Learning*, PMLR, 2020, pp. 5132–5143.

[36] J. Geng, Y. Mou, F. Li, *et al.*, "Towards general deep leakage in federated learning," *arXiv preprint arXiv:2110.09074*, 2021.

[37] A. Wainakh, F. Ventola, T. Müßig, *et al.*, "User label leakage from gradients in federated learning," *arXiv preprint arXiv:2105.09369*, 2021.

[38] J. Xu, C. Hong, J. Huang, L. Y. Chen, and J. Decouchant, "Agic: Approximate gradient inversion attack on federated learning," in *2022 41st International Symposium on Reliable Distributed Systems (SRDS)*, IEEE, 2022, pp. 12–22.

[39] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE signal processing magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[40] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[41] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

A

Appendix

The appendix includes the submitted conference paper about information leakage analysis and the corresponding attack in decentralized federated learning.

Privacy Analysis of Decentralized Federated Learning

Wenrui Yu[*]
Delft University of Technology
w.yu-6@student.tudelft.nl

Qiongxiu Li[*]
Tsinghua University
qiongxiuli@mail.tsinghua.edu.cn

Milan Lopuhaä-Zwakenberg
University of Twente
m.a.lopuhaa@utwente.nl

Mads GræsbChristensen
Aalborg University
mgc@es.aau.dk

Richard Heusdens
Netherlands Defence Academy and Delft University of Technology
r.heusdens@tudelft.nl
[*]

Abstract

In this paper we demonstrate that decentralized federated learning (FL) outperforms centralized FL when privacy is concerned. Recently, privacy issues in FL have received a lot of attention. Most of the existing work focuses on centralized FL where it is assumed that a central server is available. As for the decentralized case where no central server is required, little research has been done as it is generally difficult to analytically track information loss over iterations. In this paper, we take a first step to perform a theoretical privacy analysis of decentralized FL. By analyzing the information exchange in the decentralized network, we derive an upper bound on privacy leakage and show information-theoretically that the privacy loss in decentralized FL is less than or equal to the loss in centralized FL. Unlike centralized FL, where local gradients of individual participants are shared, differences of local gradients over successive iterations are revealed in decentralized FL. Traditional gradient inversion attacks can still be applied to decentralized FL to reconstruct the input data, but the reconstruction performance is severely degraded. One reason is that in centralized FL the label information can often be computed analytically from the gradients, while this is not possible in decentralized FL. Numerical simulations support our theoretical findings.

1 Introduction

Federated Learning (FL) performs collaborative training between multiple participants/nodes/clients without directly sharing each node's raw data [1]. FL can be implemented using a centralized/star topology or a decentralized topology, as shown in Figure 1[2]. The centralized topology, which is the predominant topology in FL, has a central server which communicates with each and every node individually. In such a setting, the main procedure of FL can be summarized into three steps: 1) all

[**] Equal contribution

Preprint. Under review.

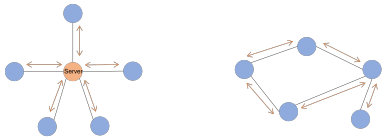

Figure 1: Two topologies in federated learning

nodes first train local models based on their own private dataset and send the local model updates, such as gradients, to the server; 2) the server aggregates the local models and determines a global model and returns this model to all nodes; 3) all nodes update the local models based on the updated global model and send the model updates back to the server, after which step 2 and 3 are repeated until convergence. In practice a centralized server might not be available as it requires high communication bandwidth and the server must be trusted by all clients. In addition, centralized topologies have a single point of failure and are therefore vulnerable to attacks aiming to bring down the entire network. Decentralized processing offers an alternative as information is only exchanged between (locally) connected nodes, thereby eliminating the need for a central server for model aggregation.

Decentralized FL protocols fall into two main categories. The first category contains average-consensus based protocols. With these protocols, nodes still train their local model, but instead of sending model parameters to a central server, the data aggregation is done in a distributed manner. Examples of these protocols are the empirical methods where the data aggregation is done using average consensus techniques such as gossiping SGD [3], D-PSGD [4] and variations thereof [5, 6]. However, as shown in [7], these methods do not perform well in the case that the data at the nodes are not independent and identically distributed (non-IID). The second category contains protocols that are based on distributed optimization. These (iterative) methods formulate the underlying problem as a constrained optimization problem and solve the optimization problem using distributed solvers like ADMM [8, 9, 10] or PDMM [11, 12, 13]. The constraints are formulated in such a way that, after convergence, the learned models at all nodes are identical. Hence, there is no explicit separation between updating local models and the update of the global model, i.e., the three steps in centralized FL mentioned before are executed simultaneously. As shown in [13], these methods have the advantage that they also work well for non-IID data.

Although the data is not exchanged directly with a server or neighboring nodes, FL is known to be vulnerable to privacy attacks as the exchanged information, such as gradients or weights, still poses a risk for privacy leakage. Most of the existing work on privacy leakage focuses on the centralized case. One example is the gradient inversion attack [14, 15, 16, 17, 18, 19, 20, 21, 22, 23], which is an iterative method for finding input data that produce a gradient similar to the gradient generated by the private data. Such attacks are based on the assumption that similar gradients are produced by similar datasets. To recover inputs, knowing label information helps accelerate the optimization process and improves the reconstructing performance. It is shown in [15] that for a neural network (NN) trained with cross-entropy loss, label information can be analytically computed from the gradients of the last layer. For that reason, the label information is usually assumed to be known in existing work [16].

In [24] it is shown that average-consensus based decentralized FL protocols do not offer privacy advantages over centralized FL, as the traditional gradient inversion attack can still be applied to these protocols. The privacy loss of optimization-based decentralized FL protocols, on the other hand, has, to the best of our knowledge, been rarely investigated. Analyzing privacy leakage in distributed algorithms is generally a challenging task as it is difficult to track information leakage over several iterations. In this paper, we take a first step to perform a theoretical privacy analysis of decentralized FL by analyzing the information flow in the network. Our main contributions are summarized below:

- By analyzing the gradient information shared in optimization-based decentralized FL, we derive an upper bound on the privacy loss which is the difference of local gradients over successive iterations, and show that from an information theoretical point of view the privacy loss in decentralized FL is less than or equal to the loss in centralized FL where local gradients are revealed. To the best of our knowledge, this is the first theoretical privacy analysis in this context.

- We show that, in the case of optimization-based decentralized FL, a series of homogeneous gradient inversion attacks can still be applied to reconstruct the original input data, but that the reconstruction performance is significantly degraded compared to centralized FL as there is less information available to the adversary. In addition, analytical label recovery [15] is no longer possible. Consequently, optimization-based decentralized FL is less vulnerable to privacy attacks than centralized FL.

In the following a theoretical privacy analysis is given for optimization-based decentralized FL and numerical verifications are provided to support our theoretical findings.

2 Preliminaries

2.1 Decentralized problem formulation

We will model the network of nodes/agents by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices representing the nodes and $\mathcal{E}$ is the set of undirected edges representing the communication links in the network. We use $\mathcal{N}_i = \{j \in \mathcal{V} : (i,j) \in \mathcal{E}\}$ to denote the set of neighbors of node i and $d_i = |\mathcal{N}_i|$ is the degree of node i. Each node i has a local dataset $\{(\boldsymbol{x}_{ik}, \ell_{ik}) : k = 1, \ldots, n_i\}$, where $\boldsymbol{x}_{ik} \in \mathbb{R}^v$ is an input sample, $\ell_{ik} \in \mathbb{R}$ is the associated label and n_i is the number of input samples. The dimension v of the data samples is application-dependent. Collecting the $\boldsymbol{x}_{ik}$s and ℓ_{ik}s, we define $\boldsymbol{x}_i = (\boldsymbol{x}_{i1}^\intercal, \ldots, \boldsymbol{x}_{in_i}^\intercal)^\intercal$ and $\boldsymbol{\ell}_i = (\ell_{i1}, \ldots, \ell_{in_i})^\intercal$, where the superscript $(\cdot)^\intercal$ denotes matrix transposition. Let $f_i(\boldsymbol{w}_i, (\boldsymbol{x}_i, \boldsymbol{\ell}_i))$ denote the cost function of node i where $\boldsymbol{w}_i \in \mathbb{R}^u$ is the model weight to be learned from the input dataset $(\boldsymbol{x}_i, \boldsymbol{\ell}_i)$, whose dimension, again, depends on the application. In the remainder of the paper we will omit the $(\boldsymbol{x}_i, \boldsymbol{\ell}_i)$ dependency for notational convenience when it is clear from the context and simply write $f_i(\boldsymbol{w}_i)$.

The goal of optimization-based decentralized FL is to collaboratively learn a global model, given the local datasets $\{(\boldsymbol{x}_i, \boldsymbol{\ell}_i) : i \in \mathcal{V}\}$, without any centralized coordination. The underlying problem can be posed as a constrained optimization problem given by

$$\begin{aligned} \min_{\{\boldsymbol{w}_i\,:\,i\in\mathcal{V}\}} \quad & \sum_{i\in\mathcal{V}} f_i(\boldsymbol{w}_i), \\ \text{subject to} \quad & \forall (i,j) \in \mathcal{E} : \boldsymbol{B}_{i|j}\boldsymbol{w}_i + \boldsymbol{B}_{j|i}\boldsymbol{w}_j = \boldsymbol{0}, \end{aligned} \tag{1}$$

where $\boldsymbol{B}_{i|j} \in \mathbb{R}^{u\times u}$ is used to guarantee consensus after convergence, i.e., $\forall i \in \mathcal{V}, \forall j \in \mathcal{V}$: $\boldsymbol{w}_i = \boldsymbol{w}_j$. Hence, $\boldsymbol{B}_{i|j} = -\boldsymbol{B}_{j|i} = \pm \boldsymbol{I}_u$, where $\boldsymbol{I}_u$ is the $u \times u$ identity matrix. In the following we will use the following convention:

$$\forall\,(i,j) \in \mathcal{E} :\ \boldsymbol{B}_{i|j} = \begin{cases} \boldsymbol{I}_u, & \text{if } i < j, \\ -\boldsymbol{I}_u, & \text{if } i > j. \end{cases} \tag{2}$$

2.2 Threat model

We consider two widely-used adversary models: the eavesdropping and the passive (or honest-but-curious) adversary model. The passive adversary consists of a number of colluding nodes, referred to as *corrupt nodes*, which will follow the algorithm instructions but will use received information to infer the input data of the other, non-corrupt nodes. We will refer to the latter as *honest nodes*. To this end, the adversaries have the following information at their disposal: (a) all messages communicated through non-encrypted channels, and (b) all information collected by the corrupt nodes.

3 Decentralized FL using ADMM/PDMM

The optimization problem (1) can be solved using distributed solvers like ADMM [25] and PDMM [11, 12]. ADMM is guaranteed to converge to the optimal solution for arbitrary convex, closed and proper (CCP) objective functions f_i, whereas PDMM will converge in the case of differentiable and strongly convex functions [12]. Recently, it has been shown that these solvers are also effective when applied to non-convex problems like training deep neural networks (DNNs) [13]. From a monotone operator theory perspective [26, 12], ADMM is a $\frac{1}{2}$-averaged version of PDMM and can, therefore, be analyzed in the same framework. Due to the averaging, ADMM is generally slower than PDMM,

assuming it converges. ADMM/PDMM solves the optimization problem (1) iteratively, where the update equations for node i are given by [27]

$$\boldsymbol{w}_i^{(t+1)} = \arg\min_{\boldsymbol{w}_i} \left(f_i(\boldsymbol{w}_i) + \sum_{j\in\mathcal{N}_i} \boldsymbol{z}_{i|j}^{(t)\intercal} \boldsymbol{B}_{i|j} \boldsymbol{w}_i + \frac{\rho d_i}{2} \boldsymbol{w}_i^2 \right), \tag{3}$$

$$\forall j \in \mathcal{N}_i : \boldsymbol{z}_{j|i}^{(t+1)} = (1-\theta)\boldsymbol{z}_{j|i}^{(t)} + \theta\big(\boldsymbol{z}_{i|j}^{(t)} + 2\rho \boldsymbol{B}_{i|j} \boldsymbol{w}_i^{(t+1)}\big), \tag{4}$$

where ρ is a constant controlling the rate of convergence. The parameter $\theta \in (0, 1]$ controls the operator averaging, where $\theta = \frac{1}{2}$ (Peaceman-Rachford splitting) results in ADMM and $\theta = 1$ (Douglas-Rachford splitting) yields PDMM. Equation (3) updates the local variables (weights) $\boldsymbol{w}_i$, whereas (4) represents the exchange of auxiliary variables in the network. The optimality condition for (3) is given by[2]

$$0 = \nabla f_i(\boldsymbol{w}_i^{(t+1)}) + \sum_{j\in\mathcal{N}_i} \boldsymbol{B}_{i|j}^{\intercal} \boldsymbol{z}_{i|j}^{(t)} + \rho d_i \boldsymbol{w}_i^{(t+1)}. \tag{5}$$

Since the adversary can eavesdrop all communication channels, by inspection of (5), transmitting the auxiliary variables $\boldsymbol{z}_{j|i}$ would reveal $\nabla f_i(\boldsymbol{w}_i^{(t)})$, as $\boldsymbol{w}_i^{(t)}$ can be determined from (4). Encrypting $\boldsymbol{z}_{j|i}$ at every iteration would solve the problem but is computationally too demanding. To overcome this problem, only initial values $\boldsymbol{z}_{j|i}^{(0)}$ are securely transmitted and $\Delta\boldsymbol{z}_{j|i}^{(t+1)} = \boldsymbol{z}_{j|i}^{(t+1)} - \boldsymbol{z}_{j|i}^{(t)}$ are transmitted (without any encryption) during subsequent iterations [27, 28]. Thus, upon receiving $\Delta\boldsymbol{z}_{j|i}^{(t+1)}$, the auxiliary variable $\boldsymbol{z}_{j|i}^{(t+1)}$ can be constructed as

$$\boldsymbol{z}_{j|i}^{(t+1)} = \boldsymbol{z}_{j|i}^{(t)} + \Delta\boldsymbol{z}_{j|i}^{(t+1)} = \sum_{\tau=1}^{t+1} \Delta\boldsymbol{z}_{j|i}^{(\tau)} + \boldsymbol{z}_{j|i}^{(0)}. \tag{6}$$

Hence, $\boldsymbol{z}_{j|i}^{(t+1)}$ can only be determined whenever $\boldsymbol{z}_{j|i}^{(0)}$ is known, so that eavesdropping only reveals

$$\left\{\Delta\boldsymbol{z}_{j|i}^{(t)} : t \geq 1, (i,j) \in \mathcal{E}\right\}. \tag{7}$$

Details of decentralized FL based on distributed optimization are summarized in Algorithm 1. Although exact solutions of (3) are usually unavailable, convergence analysis of inexact updates has been extensively investigated. It is shown in [13] that, using quadratic approximations, PDMM achieves good performance for non-convex tasks such as training DNNs. Also, [29] gives convergence guarantees in the case of transmitting quantized variables.

4 Privacy bound

In this section, we derive an upper bound on the privacy loss of Algorithm 1. For simplicity, we will consider $\theta = 1$, i.e., PDMM, but the results can be easily generalized to arbitrary $\theta \in (0, 1]$. We have the following result.

Theorem 1. *Assume there is at least one corrupt node in the network, then for each honest node i the adversary can learn $\{\boldsymbol{w}_i^{(t)} : t \geq 1\}$ as well as*

$$\nabla f_i\big(\boldsymbol{w}_i^{(t)}\big) - \nabla f_i\big(\boldsymbol{w}_i^{(t+2)}\big), \quad t \geq 1. \tag{8}$$

Proof. We first prove that all $\{\boldsymbol{w}_i^{(t)} : t \geq 1\}$ are known to the adversary. Using (4) we have

$$\begin{aligned}\Delta\boldsymbol{z}_{j|i}^{(t+1)} - \Delta\boldsymbol{z}_{i|j}^{(t)} &= \boldsymbol{z}_{j|i}^{(t+1)} - \boldsymbol{z}_{j|i}^{(t)} - \big(\boldsymbol{z}_{i|j}^{(t)} - \boldsymbol{z}_{i|j}^{(t-1)}\big) \\ &= 2\rho\boldsymbol{B}_{i|j}\boldsymbol{w}_i^{(t+1)} - 2\rho\boldsymbol{B}_{i|j}\boldsymbol{w}_i^{(t)} = 2\rho\boldsymbol{B}_{i|j}(\boldsymbol{w}_i^{(t+1)} - \boldsymbol{w}_i^{(t)}).\end{aligned} \tag{9}$$

By collecting all $\Delta\boldsymbol{z}_{j|i}^{(t+1)}$s, the adversary has knowledge of $\boldsymbol{w}_i^{(t+1)} - \boldsymbol{w}_i^{(t)}$ at every iteration. Moreover, since $\boldsymbol{w}_i^{(t)} \to \boldsymbol{w}^*$ for all $i \in \mathcal{V}$, the adversary can infer the individual $\boldsymbol{w}_i^{(t)}$s.

[2]Note that ADMM can also be applied to non-differentiable problems where the optimality condition can be expressed in terms of subdifferentials: $0 \in \partial f_i(\boldsymbol{w}_i^{(t+1)}) + \sum_{j\in\mathcal{N}_i} \boldsymbol{B}_{i|j}^{\intercal} \boldsymbol{z}_{i|j}^{(t)} + \rho d_i \boldsymbol{w}_i^{(t+1)}$.

Algorithm 1 Decentralized FL using differential ADMM/PDMM

Initialization of $\boldsymbol{z}^{(0)}$ ▷ Initialization
for $t = 0, 1, \ldots$ **do**
  for each node $i \in \mathcal{V}$ **in parallel do** ▷ Update nodes
    $\boldsymbol{w}_i^{(t+1)} = \arg\min_{\boldsymbol{w}_i} \left(f_i(\boldsymbol{w}_i) + \sum_{j\in\mathcal{N}_i} \boldsymbol{z}_{i|j}^{(t)\top} \boldsymbol{B}_{i|j} \boldsymbol{w}_i + \frac{\rho d_i}{2} \boldsymbol{w}_i^2 \right)$
    for each $j \in \mathcal{N}_i$ **do**
      $\boldsymbol{z}_{j|i}^{(t+1)} = (1-\theta)\boldsymbol{z}_{j|i}^{(t)} + \theta \left(\boldsymbol{z}_{i|j}^{(t)} + 2\rho \boldsymbol{B}_{i|j} \boldsymbol{w}_i^{(t+1)} \right)$
      $\Delta \boldsymbol{z}_{j|i}^{(t+1)} = \boldsymbol{z}_{j|i}^{(t+1)} - \boldsymbol{z}_{j|i}^{(t)}$
    end for
  end for
  for each $i \in \mathcal{V}$, $j \in \mathcal{N}_i$ **do** ▷ Data exchange (unicast)
    Node$_j \leftarrow$ Node$_i(\Delta \boldsymbol{z}_{j|i}^{(t+1)})$
  end for
  for each $i \in \mathcal{V}$, $j \in \mathcal{N}_i$ **do** ▷ Secondary Update
    $\boldsymbol{z}_{j|i}^{(t+1)} = \boldsymbol{z}_{j|i}^{(t)} + \Delta \boldsymbol{z}_{j|i}^{(t+1)}$
  end for
end for

To prove (8), consider two successive $\boldsymbol{z}$-updates (4). We have

$$\boldsymbol{z}_{i|j}^{(t+1)} - \boldsymbol{z}_{i|j}^{(t-1)} = 2\rho \boldsymbol{B}_{i|j} \big(\boldsymbol{w}_i^{(t)} - \boldsymbol{w}_j^{(t+1)} \big). \tag{10}$$

By combining (10) and the optimality condition (3) at iteration t and $t+2$, we obtain

$$\begin{aligned} \nabla f_i\big(\boldsymbol{w}_i^{(t)}\big) - \nabla f_i\big(\boldsymbol{w}_i^{(t+2)}\big) &= \sum_{j\in\mathcal{N}_i} \boldsymbol{B}_{i|j}^{\intercal} \left(\boldsymbol{z}_{i|j}^{(t+1)} - \boldsymbol{z}_{i|j}^{(t-1)} \right) + \rho d_i \left(\boldsymbol{w}_i^{(t+2)} - \boldsymbol{w}_i^{(t)} \right) \\ &= \rho d_i \big(\boldsymbol{w}_i^{(t)} + \boldsymbol{w}_i^{(t+2)} \big) - 2\rho \sum_{j\in\mathcal{N}_i} \boldsymbol{w}_j^{(t+1)}. \end{aligned} \tag{11}$$

Since all terms on the RHS are known to the adversary, thus (8) is known, which completes the proof. □

Some remarks are in place here. First of all, in any practical situation we stop the algorithm after a finite number of iterations, say $t_{\max}$. As a consequence, the adversary only knows $\boldsymbol{w}^*$ up to an error and can therefore only estimate the individual $\boldsymbol{w}_i^{(t)}$s up to a certain accuracy. To quantify this error, let $\boldsymbol{\epsilon}_i^{(t_{\max})} = \hat{\boldsymbol{w}}_i^{(t_{\max})} - \boldsymbol{w}_i^{(t_{\max})}$ be the adversary's estimation error in $\boldsymbol{w}_i^{(t_{\max})}$. Since $\boldsymbol{w}_i^{(t)} = \boldsymbol{w}_i^{(t_{\max})} - \sum_{\tau=t}^{t_{\max}-1} \big(\boldsymbol{w}_i^{(\tau+1)} - \boldsymbol{w}_i^{(\tau)} \big)$ and the adversary has knowledge of $\boldsymbol{w}_i^{(t+1)} - \boldsymbol{w}_i^{(t)}$ at every iteration, we conclude that

$$\hat{\boldsymbol{w}}_i^{(t)} = \boldsymbol{w}_i^{(t)} + \boldsymbol{\epsilon}_i^{(t_{\max})}, \quad 1 \le t \le t_{\max}, \tag{12}$$

so that the adversary can only estimate (8) up to a certain accuracy determined by $\boldsymbol{\epsilon}_i^{(t_{\max})}$.

Secondly, in some applications, such as training neural networks, (3) is only solved approximately. That is, at every iteration t we work with approximations of $\boldsymbol{w}_i^{(t)}$ and as a consequence, the optimality condition (3) holds approximately. That is, we have

$$\nabla f_i(\boldsymbol{w}_i^{(t+1)}) + \sum_{j\in\mathcal{N}_i} \boldsymbol{B}_{i|j}^{\intercal} \boldsymbol{z}_{i|j}^{(t)} + \rho d_i \boldsymbol{w}_i^{(t+1)} = \boldsymbol{\varepsilon}_i^{(t+1)},$$

where $\boldsymbol{\varepsilon}_i$ denotes an approximation error. Since (9) and (10) still hold, the RHS of (11) in this case gets an additional term $\boldsymbol{\varepsilon}_i^{(t)} - \boldsymbol{\varepsilon}_i^{(t+2)}$. In certain cases, however, we exactly know what the error terms are. As an example, consider the case where we use single gradient-descent step to solve (3). We then have

$$\boldsymbol{w}_i^{(t+1)} = \boldsymbol{w}_i^{(t)} - \mu(\nabla f_i(\boldsymbol{w}_i^{(t)}) + \sum_{j\in\mathcal{N}_i} \boldsymbol{B}_{i|j}^{\intercal} \boldsymbol{z}_{i|j}^{(t)} + \rho d_i \boldsymbol{w}_i^{(t)}), \tag{13}$$

where μ denotes the step size, so that $\boldsymbol{\varepsilon}_i^{(t)} = \mu^{-1}(\boldsymbol{w}_i^{(t)} - \boldsymbol{w}_i^{(t+1)})$. Assuming μ is known, the adversary still knows (8). Overall, we conclude that (8) is an upper bound of privacy loss.

The following corollary shows that the result of Theorem 1 implies that the privacy leakage of decentralized FL is less than or equal to that of centralized FL.

Corollary 1. *For each honest node i, let $\boldsymbol{X}_i$ and $\nabla f_i\big(\boldsymbol{W}_i^{(t)}\big)$ be random variables having realizations $\boldsymbol{x}_i$ and $\nabla f_i\big(\boldsymbol{w}_i^{(t)}\big)$, respectively. Moreover, let $\mathcal{A}_c = \{\nabla f_i\big(\boldsymbol{W}_i^{(t)}\big) : t \geq 1\}$ denote the set of information collected by the adversary in centralized FL. Similarly, let $\mathcal{A}_d = \{\nabla f_i\big(\boldsymbol{W}_i^{(t)}\big) - \nabla f_i\big(\boldsymbol{W}_i^{(t+2)}\big) : t \geq 1\}$ denote the set of information collected by the adversary in decentralized FL. We then have*

$$I(\boldsymbol{X}_i; \mathcal{A}_c) \geq I(\boldsymbol{X}_i; \mathcal{A}_d), \tag{14}$$

where $I(\cdot\,;\cdot)$ denotes mutual information [30].

Proof. Let $\mathcal{A}_e = \{\nabla f_i\big(\boldsymbol{W}_i^{(1)}\big), \nabla f_i\big(\boldsymbol{W}_i^{(2)}\big)\}$. Then

$$\begin{aligned} I(\boldsymbol{X}_i; \mathcal{A}_c) - I(\boldsymbol{X}_i; \mathcal{A}_d) &\overset{(a)}{=} I(\boldsymbol{X}_i; \mathcal{A}_e, \mathcal{A}_d) - I(\boldsymbol{X}_i; \mathcal{A}_d) \\ &\overset{(b)}{=} I(\boldsymbol{X}_i; \mathcal{A}_e | \mathcal{A}_d) \geq 0, \end{aligned}$$

where (a) holds since $\mathcal{A}_c$ can be constructed from $\mathcal{A}_d \cup \mathcal{A}_e$, and (b) follows from the chain rule for mutual information. □

Note that $I(\boldsymbol{X}_i; \mathcal{A}_e | \mathcal{A}_d) > 0$ for any meaningful learning process. As a consequence, we will have strict inequality in any practical case, showing that decentralized FL outperforms centralized FL when privacy is concerned.

5 Reconstruction of input data

Given the knowledge of gradient differences, traditional gradient inversion attacks as commonly used in centralized FL can, after some modifications, also be applied to reconstruct the input data in decentralized Fl. In what follows we will first briefly discuss how traditional attacks work and then explain how to apply them in the decentralized setting and point out the main differences.

5.1 Gradient inversion attack in centralized FL

In centralized FL, individual nodes will exchange local gradients $\nabla f_i(\boldsymbol{w}_i^{(t)})$ with the server. For each node's local dataset $(\boldsymbol{x}_i, \boldsymbol{\ell}_i)$, the adversary can (partially) recover the input data $(\boldsymbol{x}_i, \boldsymbol{\ell}_i)$ as [14]

$$(\boldsymbol{x}_i'^*, \boldsymbol{\ell}_i'^*) = \underset{\boldsymbol{x}_i', \boldsymbol{\ell}_i'}{\arg\min} \left\| \nabla f_i(\boldsymbol{w}_i, (\boldsymbol{x}_i', \boldsymbol{\ell}_i')) - \nabla f_i(\boldsymbol{w}_i, (\boldsymbol{x}_i, \boldsymbol{\ell}_i)) \right\|^2, \tag{15}$$

or variants thereof [16]. In fact, the gradient inversion attack iteratively finds input data that produce a gradient similar to the gradient generated by the (private) input data.

It has been shown recently that the label information $\boldsymbol{\ell}_i$ does not need to be optimized as it can be analytically inferred from the shared gradients at the early iterations of model training [15, 31]. The reason for this is the following. Consider a classification task where the neural network has L layers and is trained with cross-entropy loss. For simplicity assume $n_i = 1$ (one data sample at each node). Let $\boldsymbol{y} = (y_1, \ldots, y_C)$ denote the outputs (logits), where y_i is the score (confidence) predicted for the ith class. With this, the cross-entropy loss over one-hot labels is given by

$$f_i(\boldsymbol{w}_i) = -\log\big(\frac{e^{y_{\ell_i}}}{\sum_j e^{y_j}}\big) = \log\Big(\sum\nolimits_j e^{y_j}\Big) - y_{\ell_i}, \tag{16}$$

where $\log(\cdot)$ denotes the natural logarithm. Let $\boldsymbol{w}_{i,L,c}$ denote the weights in the output layer L corresponding to output y_c. The gradient of $f_i(\boldsymbol{w}_i)$ with respect to $\boldsymbol{w}_{i,L,c}$ can then be expressed as [15]:

$$\nabla' f_i(\boldsymbol{w}_{i,L,c}) \triangleq \frac{\partial f_i(\boldsymbol{w}_i)}{\partial \boldsymbol{w}_{i,L,c}} = \frac{\partial f_i(\boldsymbol{w}_i)}{\partial y_c} \frac{\partial y_c}{\partial \boldsymbol{w}_{i,L,c}} = g_c \boldsymbol{a}_{L-1}, \tag{17}$$

where $\boldsymbol{a}_{L-1}$ is the activation at layer $L-1$ and g_c is the gradient of the cross entropy (16) with respect to logit c given by

$$g_c = \frac{e^{y_c}}{\sum_j e^{y_j}} - \delta_{c,\ell_i}, \tag{18}$$

where $\delta_{c,\boldsymbol{\ell}_i}$ is the Kronecker-delta. Hence, $g_c < 0$ for $c = \ell_i$ and $g_c > 0$ otherwise. Since the activation $\boldsymbol{a}_{L-1}$ is independent of the class index c, the ground-truth label ℓ_i can be inferred from the shared gradients since $\nabla' f_i(\boldsymbol{w}_{i,L,\ell_i}) \nabla' f_i(\boldsymbol{w}_{i,L,c}) = g_{\ell_i} g_c \|\boldsymbol{a}_{L-1}\|^2 < 0$ for $c \neq \ell_i$ and positive only for $c = \ell_i$. Similar results hold for the case where $n_i > 1$ [31].

5.2 Differential gradient attack in decentralized FL

Motivated by the gradient attack described above, we propose a differential gradient attack for decentralized FL given by

$$(\boldsymbol{x}_i'^*, \boldsymbol{\ell}_i'^*) = \arg\min_{\boldsymbol{x}_i', \boldsymbol{\ell}_i'} \Big\| \nabla f_i \left(\boldsymbol{w}_i^{(t)}, (\boldsymbol{x}_i', \boldsymbol{\ell}_i') \right) - \nabla f_i \left(\boldsymbol{w}_i^{(t+2)}, (\boldsymbol{x}_i', \boldsymbol{\ell}_i') \right) - \left(\nabla f_i \left(\boldsymbol{w}_i^{(t)}, (\boldsymbol{x}_i, \boldsymbol{\ell}_i) \right) - \nabla f_i \left(\boldsymbol{w}_i^{(t+2)}, (\boldsymbol{x}_i, \boldsymbol{\ell}_i) \right) \right) \Big\|^2. \tag{19}$$

Similar to the centralized case, standard algorithms like L-BFGS [32] can be adopted for optimization.

Although the differential gradient attack looks similar to the traditional gradient attack, there are some important differences, in particular with respect to label recovery and the eavesdropping times.

Similar to the discussion above, we find

$$\begin{aligned} \nabla f(\boldsymbol{w}_{i,L,c}^{(t)}) - \nabla f(\boldsymbol{w}_{i,L,c}^{(t+2)}) &= g_c^{(t)} \boldsymbol{a}_{L-1}^{(t)} - g_c^{(t+2)} \boldsymbol{a}_{L-1}^{(t+2)} \\ &= \Big(\frac{e^{y_c^{(t)}}}{\sum_j e^{y_j^{(t)}}} - \delta_{c,\ell_i} \Big) \boldsymbol{a}_{L-1}^{(t)} - \Big(\frac{e^{y_c^{(t+2)}}}{\sum_j e^{y_j^{(t+2)}}} - \delta_{c,\ell_i} \Big) \boldsymbol{a}_{L-1}^{(t+2)}. \end{aligned} \tag{20}$$

Hence, if $c \neq \ell_i$, then both $g_c^{(t)} < 0$ and $g_c^{(t+2)} < 0$ so that we cannot use the sign information of g_c to recover the correct label; the adversary needs to consider all labels to find out the best fit, which inevitably increases the computation overhead and degrades the fidelity of the reconstructed inputs (we will present numerical validations later in Section 6).

Note that in centralized FL, the adversary can use the gradient at early iterations to conduct the gradient inversion attack. In decentralized FL, however, the adversary can only accurately estimate the difference of gradients after enough iterations. That is, the smaller the error $\epsilon_i^{(t_{\max})}$ is, the more accurate the reconstruction of the input data will be. We will validate this statement in Section 6.

As for the defense strategies, a straightforward approach is to adopt noise-insertion mechanisms such as differential privacy to achieve privacy-preservation.

6 Numerical Experiments

To validate our claims that decentralized FL protocol has privacy advantages over centralized FL, we conduct numerical validations using two examples[3]. The first example is decentralized logistic regression, the second one is learning a multi-layer perceptron.

6.1 Decentralized Logistic Regression

Consider a logistic model with model parameters $\boldsymbol{w}_i \in \mathbb{R}^v$ (weights) and $b_i \in \mathbb{R}$ (bias) where each node has a local dataset $\{(\boldsymbol{x}_{ik}, \ell_{ik}) : k = 1, \ldots, n_i\}$, where $\boldsymbol{x}_{ik} \in \mathbb{R}^v$ is an input sample, $\ell_{ik} \in \{0, 1\}$ is the associated label. In addition, let $y_{ik} = \boldsymbol{w}_i^T \boldsymbol{x}_{ik} + b_i$ denote the output of the model given the input $\boldsymbol{x}_{ik}$. Note that the bias term can be included in the weight vector. Here we explicitly separate the bias from the true network weights as it will lead to more insight into how to reconstruct the input data from the observed gradients. With this, the loss (log-likelihood) function has the form

$$f_i(\boldsymbol{w}_i, b_i) = -\sum_{k=1}^{n_i} \left(\ell_{ik} \log \frac{1}{1 + e^{-y_{ik}}} + (1 - \ell_{ik}) \log \frac{e^{-y_{ik}}}{1 + e^{-y_{ik}}} \right). \tag{21}$$

[3] `https://anonymous.4open.science/r/decentralized-FL-and-differential-gradient-attack-3016/`

Hence, (11) becomes

$$\frac{\partial f_i^{(t)}}{\partial \boldsymbol{w}_i} - \frac{\partial f_i^{(t+2)}}{\partial \boldsymbol{w}_i} = \sum_{k=1}^{n_i} \left(\frac{1}{1+e^{-y_{ik}^{(t)}}} - \frac{1}{1+e^{-y_{ik}^{(t+2)}}} \right) \boldsymbol{x}_{ik}$$
$$= \rho d_i \big(\boldsymbol{w}_i^{(t)} + \boldsymbol{w}_i^{(t+2)} \big) - 2\rho \sum_{j \in \mathcal{N}_i} \boldsymbol{w}_j^{(t+1)}, \tag{22}$$

$$\frac{\partial f_i^{(t)}}{\partial b_i} - \frac{\partial f_i^{(t+2)}}{\partial b_i} = \sum_{k=1}^{n_i} \left(\frac{1}{1+e^{-y_{ik}^{(t)}}} - \frac{1}{1+e^{-y_{ik}^{(t+2)}}} \right)$$
$$= \rho d_i \big(b_i^{(t)} + b_i^{(t+2)} \big) - 2\rho \sum_{j \in \mathcal{N}_i} b_j^{(t+1)}. \tag{23}$$

Since the adversary has the knowledge of all $\{(\boldsymbol{w}_i^{(t)}, b_i^{(t)}) : t \geq 1\}$, it thus can reconstruct (22) and (23) by collecting sufficient observations, from which the private data can be inferred. Note that in the special case where $n_i = 1$, (22) is just a scaled version of $\boldsymbol{x}_{ik}$ where the scaling is given by (23). Hence, we can analytically compute $\boldsymbol{x}_{ik}$ as $\boldsymbol{x}_{ik} = \frac{\rho d_i \left(\boldsymbol{w}_i^{(t)} + \boldsymbol{w}_i^{(t+2)} \right) - 2\rho \sum_{j \in \mathcal{N}_i} \boldsymbol{w}_j^{(t+1)}}{\rho d_i \left(b_i^{(t)} + b_i^{(t+2)} \right) - 2\rho \sum_{j \in \mathcal{N}_i} b_j^{(t+1)}}$.

Experimental setup: To validate the theory presented above, we generated a connected random geometric graph (RGG) [33] with $N = 60$ nodes. Each node i holds n_i data samples $\boldsymbol{x}_{ik} \in \mathbb{R}^2$ and binary labels $\ell_i \in \{0, 1\}$. The two datasets were generated from random samples drawn from a unit variance Gaussian distribution with mean $\boldsymbol{\mu}_0 = (-1, -1)^\intercal$ ($\ell_{ik} = 0$) and mean $\boldsymbol{\mu}_1 = (1, 1)^\intercal$ ($\ell_{ik} = 1$). PDMM was used for decentralized learning with constant $\rho = 0.4$ and the weight update (3) was implemented using a single gradient descent iteration (see (13)) with fixed step-size $\mu = 0.1$. To obtain a fair comparison between decentralized and centralized FL, each node in the centralized setting contained the same dataset as the one used for decentralized FL and the local model updates were implemented by a single gradient descent iteration having the same learning rate as the rate used in decentralized FL.

Convergence properties: Figure 2(a) shows the loss (21), averaged over all nodes, as a function of the iteration t for both centralized FL using FedAvg [1] and decentralized FL using PDMM. The number of data samples per node was set to $n_i = 1$. Figure 2(a) show that PDMM converges slightly faster than FedAvg. This is because PDMM combines the local model update and global model aggregation and jointly optimizes them, while FedAvg updates the model in two separate steps [13].

Figure 2(b) investigates the effect of early termination of the iterations after a finite number of iterations $t_{\max}$ on the reconstruction of $\boldsymbol{x}_{ik}$. The reconstruction error is defined as the average Euclidean distance between the reconstructed samples $\hat{\boldsymbol{x}}_{ik}$ and the original data samples $\boldsymbol{x}_{ik}$ given by $\frac{1}{N} \sum_{i=1}^{N} \|\hat{\boldsymbol{x}}_{ik} - \boldsymbol{x}_{ik}\|_2$. We assume there is one corrupt node, say node j, in the network and used $\hat{\boldsymbol{w}}_i^{(t_{\max})} = \boldsymbol{w}_j^{(t_{\max})}$ for all honest nodes i. We can see that, as expected, the reconstruction error will decrease as $t_{\max}$ increases, i.e., the longer the adversary waits, the higher the reconstruction accuracy will be. Note that in this example we have $n_i = 1$ so that $\boldsymbol{x}_{ik}$ can be analytically determined. Hence, the error in the reconstruction is solely due to an inaccurate estimation of the difference of gradients. Consequently, the adversary must continuously eavesdrop on the communication channels until the model has converged with sufficient accuracy. With centralized FL, the reconstructed error does not depend on the iteration number and the reconstruction accuracy is consistently better than the accuracy for decentralized FL.

6.2 Decentralized training of a multi-layer perceptron

Experimental setup: We generated an RGG with $N = 100$ nodes. Two-layer perceptrons were constructed for each node using Pytorch and we used two datasets: MNIST [34] and CIFAR-10 [35]. PDMM was used for decentralized learning with constant $\rho = 0.4$. Both the gradient and the differential gradient attacks were implemented using the L-BFGS algorithm[36] where the number of iterations was fixed to 500. The code was run on a NVIDIA RTX A6000 GPU.

Reconstructed of input data: Figure 3 and Figure 4 shows reconstruction results for both centralized and decentralized learning. Figure 3 shows example reconstructions, one for each dataset, while

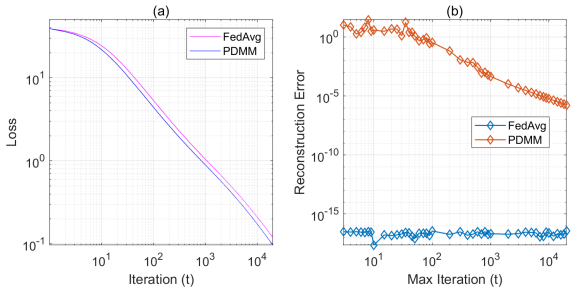

Figure 2: Performance comparisons of centralized and decentralized logistic regression. (a) Loss versus number of iterations. (b) Reconstruction error as a function of the maximum iteration number.

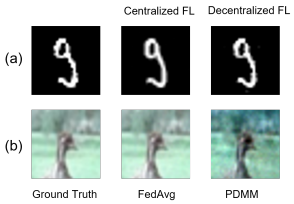

Figure 3: Reconstructed inputs using (a) the MNIST and (b) the CIFAR-10 dataset.

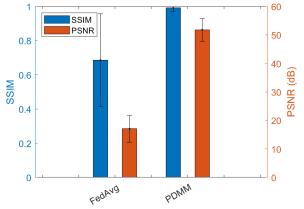

Figure 4: Comparison of the SSIM and PSNR of reconstructed images

Figure 4 shows objective results using the structural similarity index measure (SSIM) and the peak signal-to-noise ratio (PSNR), averaged over the complete dataset. We can see that the quality of the reconstructed images in decentralized FL is lower than the quality in centralized FL.

Since the label cannot be analytically computed as in the centralized case, in the implementation we traverse all possible labels to find out the optimal solution for the decentralized case. Figure 5 shows reconstruction results for the MNIST dataset (top) and the CIFAR-10 dataset (bottom) for different values of n_i. As can be seen from the figure, unlike the case of centralized FL where most of the time the reconstructed labels are correct, whereas with decentralized FL, the label is sometimes wrong. As an example, for the MNIST dataset and $n_i = 4$, the digit 0 is reconstructed while this digit was not included in the training set. In addition, the reconstruction quality for centralized FL is clearly better than the one for decentralized FL. We conclude that optimization-based decentralized FL is more difficult to attack than centralized FL, thereby consolidating our claim that it has privacy advantages over centralized FL.

7 Conclusions

In this paper, we investigated the privacy leakage in decentralized FL. We showed that optimization-based decentralized FL outperforms centralized FL when privacy is concerned. We analyzed the information flow in the network over iterations and derived an upper bound on the information loss. Using this bound, we theoretically showed that the privacy leakage in decentralized FL is less than or equal to the leakage in centralized FL, and that analytical label recovery is generally not possible in decentralized FL. As a consequence, with decentralized FL, the quality of reconstructed samples is significantly lower than the quality obtained with centralized FL, especially for large batch sizes. Experimental results confirmed our findings.

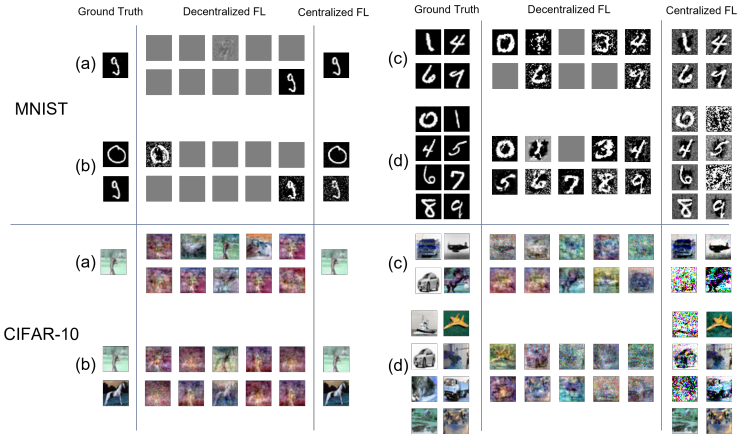

Figure 5: Reconstructed inputs for centralized and decentralized FL using the datasets MNIST (top) and CIFAR-10 (bottom) for different batch size $n_i = 1, 2, 4, 8$ ((a)-(d), respectively).

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[2] T. Li, A. K. Sahu, A. Talwalkar and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Process. Magazine, vol. 37, no. 3, pp. 50-60,*, 2020.

[3] P. H. Jin, Q. Yuan, F. Iandola, and K. Keutzer. How to scale distributed deep learning? *arXiv preprint arXiv:1611.04581*, 2016.

[4] X. Lian, C. Zhang, H. Zhang, C. Hsieh, W. Zhang and J. Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in neural information processing systems*, 30, 2017.

[5] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu. D^2: Decentralized training over decentralized data. In *International Conference on Machine Learning*, pages 4848–4856. PMLR, 2018.

[6] C. Hu, J. Jiang, and Z. Wang. Decentralized federated learning: A segmented gossip approach. *arXiv preprint arXiv:1908.07782*, 2019.

[7] H. Gao, M. Lee, G. Yu, and Z. Zhou. A graph neural network based decentralized learning scheme. *Sensors*, 22(3):1030, 2022.

[8] J.F.C. Motaand J. M.F. Xavier, P. M.Q. Aguiar, and M. Püschel. D-admm: A communication-efficient distributed algorithm for separable optimization. *IEEE Transactions on Signal processing*, 61(10):2718–2723, 2013.

[9] W. Li, Y. Liu, Z. Tian, and Q. Ling. Communication-censored linearized admm for decentralized consensus optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 6:18–34, 2019.

[10] H. Chen, Y. Ye, M. Xiao, M. Skoglund, and H.V. Poor. Coded stochastic admm for decentralized consensus optimization with edge computing. *IEEE Internet of Things Journal*, 8(7):5360–5373, 2021.

[11] G. Zhang and R. Heusdens. Distributed optimization using the primal-dual method of multipliers. *IEEE Transactions on Signal and Information Processing over Networks*, 4(1):173–187, 2017.

[12] T. Sherson, R. Heusdens, W. B. Kleijn. Derivation and analysis of the primal-dual method of multipliers based on monotone operator theory. *IEEE Trans. Signal Inf. Process. Netw., vol. 5, no. 2, pp 334-347*, 2018.

[13] K. Niwa, N. Harada, G. Zhang, and W.B. Kleijn. Edge-consensus learning: Deep learning on p2p networks with nonhomogeneous data. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 668–678, 2020.

[14] L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

[15] B. Zhao, KR Mopuri, and H. Bilen. iDLG: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

[16] J. Geiping. H. Bauermeister, H. Dröge and M. Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *NeurIPS*, 33:16937–16947, 2020.

[17] H. Yin, A. Mallya, A. Vahdat, JM Alvarez, J. Kautz, and P. Molchanov. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16337–16346, 2021.

[18] F. Boenisch, A. Dziedzic, R. Schuster, AS. Shamsabadi, I. Shumailov, and N. Papernot. When the curious abandon honesty: Federated learning is not private. *arXiv preprint arXiv:2112.02918*, 2021.

[19] J. Geng, Y. Mou, Q. Li, F. Li, O. Beyan, S. Decker, and C. Rong. Improved gradient inversion attacks and defenses in federated learning. *IEEE Transactions on Big Data*, 2023.

[20] L. Fowl, J. Geiping, W. Czaja, M. Goldblum, and T. Goldstein. Robbing the fed: Directly obtaining private data in federated learning with modified models. *arXiv preprint arXiv:2110.13057*, 2021.

[21] J. Zhu and M. Blaschko. R-gap: Recursive gradient attack on privacy. *arXiv preprint arXiv:2010.07733*, 2020.

[22] W. Wei, L. Liu, M. Loper, Ka-Ho Chow, ME Gursoy, S. Truex, and Y. Wu. A framework for evaluating gradient leakage attacks in federated learning. *arXiv preprint arXiv:2004.10397*, 2020.

[23] H. Yang, M. Ge, K. Xiang, and J. Li. Using highly compressed gradients in federated learning for data reconstruction attacks. *IEEE Transactions on Information Forensics and Security*, 18:818–830, 2022.

[24] D. Pasquini, M. Raynal, and C. Troncoso. On the privacy of decentralized machine learning. *arXiv preprint arXiv:2205.08443*, 2022.

[25] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine learning, vol. 3, no. 1, pp. 1–122*, 2011.

[26] E. Ryu, S. P. Boyd. Primer on monotone operator methods. *Appl. Comput. Math., vol. 15, no. 1, pp. 3-43,*, 2016.

[27] Q. Li, R. Heusdens and M. G. Christensen. Communication efficient privacy-preserving distributed optimization using adaptive differential quantization. *Signal Process.*, 2022.

[28] Q. Li, R. Heusdens and M. G. Christensen. Privacy-preserving distributed optimization via subspace perturbation: A general framework. In *IEEE Trans. Signal Process., vol. 68, pp. 5983 - 5996*, 2020.

[29] J. A. G. Jonkman, T. Sherson, and R. Heusdens. Quantisation effects in distributed optimisation. In *Proc. Int. Conf. Acoust., Speech, Signal Process., pp. 3649–3653*, 2018.

[30] T. M. Cover and J. A. Tomas. *Elements of information theory*. John Wiley & Sons, 2012.

[31] A. Wainakh, F. Ventola, T. Müßig, J. Keim, C. G. Cordero, E. Zimmer, T. Grube, K. Kersting, and M. Mühlhäuser. User label leakage from gradients in federated learning. *arXiv preprint arXiv:2105.09369*, 2021.

[32] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, 23(4):550–560, 1997.

[33] J. Dall and M. Christensen. Random geometric graphs. *Physical review E, vol. 66, no. 1, pp. 016121*, 2002.

[34] L. Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[35] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[36] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.