Federated Machine Learning

November 8, 2019

1 Introduction

There has been a recent surge of interest in machine learning in both academia and industry. Now, it has become harder to find areas where machine learning has not been applied as the areas of applications are no longer confined by disciplines, from engineering to history. However, the need for better and more scalable machine learning models and algorithms often leads to ethical concerns.

For instance, some implementations without the full understanding of the dataset and model have perpetuated unfairness in socio-economic contexts [7]. Generative algorithms, such as Generative Adversarial Nets and GPT-2, have raised some concerns of dispersing misinformation by malicious parties [2]. Moreover, fundamental data collection methods for training and validating models have been often overlooked as necessary components of large-scale machine learning, though many practices have been deemed to have violated individual privacy.

With these problems in mind, researchers are now focusing on the balanced development of ethics and technologies that could help mitigate these troublesome aspects of machine learning. Recent papers suggest some fundamental solutions in terms of homomorphic encryption protocols, but it is computationally expensive and algebraically limiting at times while providing the most secure methods of privacy preserving machine learning [1]. The most plausible solution to privacy concerns is Federated Machine Learning, a multi-party computation (MPC) framework that could guarantee individual privacy without compromising the accuracy of a generalized model.

This project aims to implement the recent privacy-preserving multi-party computation frameworks such as Federated Averaging and Split Neural Nets without networking components. As it is a relatively new field of machine learning, we also plan to investigate possible attacks on Federated Machine Learning schema using information leakage. Time permitting, we plan to implement these information leakage attacks in our demo.

2 Background

Federated Machine Learning proposes a solution to privacy-preserving large-scale machine learning by training models locally on an individual client device then updating the model parameters or finishing the rest of the model training on the central server.

Problem Statement.

We would like to first define the fundamental objective in machine learning called the "finite-sum problem:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{\theta})"$$

In the context of machine learning, we would like to minimize our loss of predictions over examples, so we can take f to be a loss function, $f_i(\boldsymbol{\theta}) = l(x_i, y_i; \boldsymbol{\theta})$ where l is a loss on the prediction made on example (x_i, y_i) with a set of model parameters $\boldsymbol{\theta}$. For the multi-party computation scheme with data partitioned over K-clients. Let S_k be the set of indices of data points on client $k \in \{1, 2, \dots K\}$ with $n_k := |S_k|$. We can restate the problem to "the Federated Machine Learning context:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} f(\boldsymbol{\theta}) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(\boldsymbol{\theta}) \qquad F_k(\boldsymbol{\theta}) := \frac{1}{n_k}\sum_{i\in S_k} f_i(\boldsymbol{\theta}) \qquad n = \sum_{k=1}^{K} n_k"$$

Moreover, the optimization problem implicit in federated learning scheme is often based on the following assumptions [3]. This problem is often termed Federated Optimization.

(1) **Non-IID**. The training data on a client is based on the usage by a user, which is often not representative of the population.

(2) **Unbalanced**. Some users can be heavy users while others are not.

(3) **Massively Distributed**. The number of clients participating in an optimization is much larger than the average number of examples per client. That is, $|C_t| \gg \sum_{k=1}^{K} n_k/K$

(4) **Limited Communication**. Oftentimes, clients are offline or on limited bandwidth/-expensive connections.

Algorithms.

There are several proposed algorithms for solving the Federated Optimization problem. One of the most promising algorithms is **Federated Averaging**, which is already being used in the Google Keyboard app. The algorithm not only provides privacy by design, but also personalizes models to individual users. It also claims to be more resource efficient in terms of communication rounds. Federated Stochastic Gradient Descent (FedSGD) is a special case of Federated Averaging, where the algorithm is given below:[3]

Server Executes :

Initialize the parameters $\boldsymbol{\theta}_0$.

for each round $t = 1, 2, \dots$ **do**

Pick random sample of clients C_t where $1 \leq |C_t| \leq K$.

for each client $k \in C_t$ **in parallel do**

$\boldsymbol{\theta}_{t+1}^k \leftarrow \text{ClientUpdate}(k, \boldsymbol{\theta}_t)$

$$\boldsymbol{\theta}_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} \boldsymbol{\theta}_{t+1}^k$$

Each Client Executes ClientUpdate$(k, \boldsymbol{\theta})$:

Batch data S_k into collection $\mathcal{B}$.

for each local epoch $i = 1, 2, \dots, E$ **do**

for each batch $b \in \mathcal{B}$ **do**

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \cdot \nabla F_k(\boldsymbol{\theta}; b)$

Return $\boldsymbol{\theta}$ to server

Another algorithm is **Split Neural Nets (SplitNN)**, where a part of training (upto a "cut layer") is done locally then the rest is done on the central server. For notation, let $\mathbf{A}_t^k$ be the forward pass logits of the client side upto the cut layer, $\mathbf{W}_t$ be the network weights of the server-side networks, and $\mathbf{H}_t^k$ be the network weights of the client-side networks, which will be combined with $\mathbf{W}_t$ and updated by each client for use. Here, we will use l for denoting a general loss function to be minimized. The algorithm is claimed to have achieved computation resource and bandwidth efficiency [5]. The vanilla configuration of the algorithm is done in the following way:[6]

Server Executes :

for each round $t = 1, 2, \dots$ **do**

for each client $k \in C_t$ **in parallel do**

$\mathbf{A}_t^k \leftarrow \text{ClientUpdate}(k, t)$

$\mathbf{W}_t \leftarrow \mathbf{W}_t - \alpha \cdot \nabla l(\mathbf{W}_t; \mathbf{A}_t)$

Send $\nabla l(\mathbf{A}_t; \mathbf{W}_t)$ to client k for $\text{ClientBackprop}(k, t)$

Each Client Executes ClientUpdate$(k, \boldsymbol{\theta})$:

$\mathbf{A}_t^k = \varphi$

for each local epoch $i = 1, 2, \dots, E$ **do**

for each batch $b \in \mathcal{B}$ **do**

Concatenate $f(b, \mathbf{H}_t^k)$ to $\mathbf{A}_t^k$

Return $\mathbf{A}_t^k$ to server

Each Client Executes ClientBackprop$(k, t, \nabla F_k(\mathbf{A}_t; \mathbf{W}_t))$:

for each batch $b \in \mathcal{B}$ **do**

$\mathbf{H}_t^k \leftarrow \mathbf{H}_t^k - \alpha \cdot \nabla l(\mathbf{A}_t; \mathbf{W}_t; b)$

3 Proposed Methodology

We plan to replicate the proposed federated learning algorithms as a demo. Since the project is more focused on the theoretical aspects of Federated Machine Learning, we will not be implementing the remote networking component of the algorithms. Instead, we will "simulate" the networking environment in a local machine by creating multiple functions representing individual clients and computing server-side computation in `__main__()` function.

We will investigate the computation and communication efficiency by counting the number of communication rounds when each function is called. We will be timing each computation components done in each client and server to measure computation resource efficiency. Time and resource permitting, we would also like to investigate the scalability of each algorithm.

For the dataset, we will be using the MNIST handwritten digits dataset. MNIST dataset will provide sufficient number of examples without resource-heavy features. In order to simulate the non-IID assumption, we will sort the dataset according to the labels and split the dataset into subsets in order (possibly of different size), then assign them to virtual clients. We would like to see how biased individual client data could affect the performance of the algorithm.

Time permitting, we would also like to study the possible attacks on distributed learning scheme. Recent studies are now looking into possible attacks taking advantage of information leakage that may be inherent in algorithm/implementation design [4]. We would like to see how malicious parties can corrupt training data with fake(generated) dataset and extract private information in distributed learning scheme where some information is leaked by design.

References

[1] M. Al-Rubaie and J. M. Chang, *Privacy preserving machine learning: Threats and solutions*, arXiv preprint arXiv:1804.11238, (2018).

[2] B. Hitaj, G. Ateniese, and F. Perez-Cruz, *Deep models under the gan: information leakage from collaborative deep learning*, in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2017, pp. 603–618.

[3] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, et al., *Communication-efficient learning of deep networks from decentralized data*, arXiv preprint arXiv:1602.05629, (2016).

[4] P. Vepakomma, O. Gupta, A. Dubey, and R. Raskar, *Reducing leakage in distributed deep learning for sensitive health data*.

[5] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, *Split learning for health: Distributed deep learning without sharing raw patient data*, arXiv preprint arXiv:1812.00564, (2018).

[6] P. Vepakomma, T. Swedish, R. Raskar, O. Gupta, and A. Dubey, *No peek: A survey of private distributed deep learning*, arXiv preprint arXiv:1812.03288, (2018).

[7] Y. Wang and M. Kosinski, *Deep neural networks can detect sexual orientation from faces*, 2017.