Federated Learning for Pest Classification

EE 451 - Supervised Research Exposition

Ayush Munjal, 19D070014
Department of Electrical Engineering
Indian Institute of Technology Bombay, Mumbai

Guided By Prof. Rajbabu Velmurugan

Abstract

This document discusses about Federated learning and its application in pest detection. **Federated learning** is used to learn similar models on different devices without sharing their data with each other. There can be many variants of federated learning by changing the network and algorithms used. Federated learning has to address some important concerns like privacy, communication bottleneck, etc.

Contents

1 Introduction

Federated learning is a method of shared machine learning, where multiple clients helps in learning of each other without sharing their actual training data but their updates of the model, with or without a central server. Federated learning with a server is called **centralized federated learning**, while without a server is **decentralized federated learning**. In centralized federated learning, there is a central server that communicates with each client and distributes its global model to each client. The usage of global model by each client allows each client to use their own local data for training and also integrating the features learned by other client. Thus, federated learning also helps in model generalization capability. In decentralized federated learning, clients communicate with each other and share their model updates this also helps in generalization of model. Federated learning allows various devices (clients) to learn complex models which they may not be able to learn on their own due to memory limitations while protecting data of each client. However, there are some serious challenges faced in federated learning setup like preservation of privacy, communication overhead, scalability, heterogeneous hardware, etc. Federated learning is used in many applications nowadays like mobile phones, autonomous driving, smart home, virtual reality, etc.

One **global step** of centralized federated learning consists of first sharing the global model with each client, local training of each client using this model and their private data, communicating the model updates to central server, and aggregating the model updates to construct a new global model at central server. Several iterations of this global steps are carried out until convergence is achieved. Decentralized federated learning is also similar except that instead of communicating with global server, clients communicate with each other.

Let's discuss one common tool that uses federated learning Google keyboard or **Gboard**. It uses centralized federated learning and encrypted communication of model updates to the central server. Federated learning provides lower latency, smarter models, and less power consumption while ensuring privacy. The algorithm it uses is based on FedAvg algorithm (which will be discussed later in detail). It compresses model updates to reduce communication costs. Training is performed only when the phone is idle, charging and on free wireless connection to not impact the phone's performance.

1.1 Different federated learning methods

There are two types of federated learning centralized federated learning and decentralized federated learning.

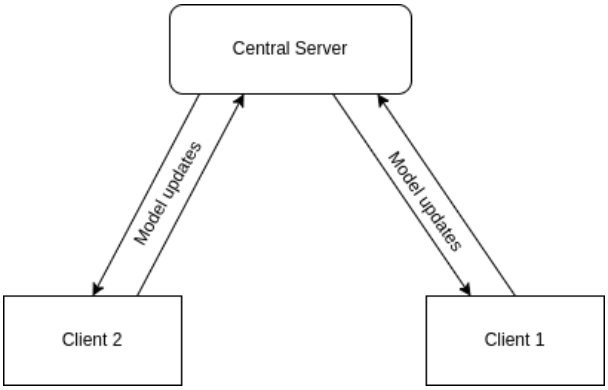

Figure 1: Centralized federated learning

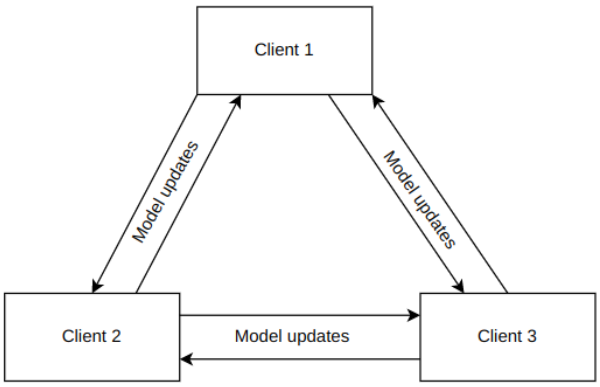

Figure 2: Decentralized federated learning

Centralized federated learning: In centralized federated learning, there is a global server which communicates with each client, i.e., receive updates from each client and distribute updated global model to each client as shown in Fig. 1. There is no communication between clients.

Decentralized federated learning: In decentralized federated learning, there is no global server but clients communicate with each other as shown in Fig. 2 for their model updates and aggregate the update they thus received for updating their local model. In such system each client can choose which and how many clients to communicate with.

The comparison between these two types of federated learning will be discussed later in detail. Now we will discuss algorithms for centralized federated learning. Many different federated learning algorithms can be constructed by changing the global aggregation step, local training, etc. Some common federated learning algorithms are FedAvg, FedProx, local gradient descent and q-FedAvg. In this section local gradient descent, FedAvg and FedProx will be discussed in more detail.

Local gradient descent: This is a relatively simple algorithm. First, all the clients are initialized with same global model then models at each client is averaged over all the model until a given number of iterations after which local training of each client is carried out only without any further communication.

FedAvg: This is the most basic federated learning algorithm, in local training step it uses SGD optimizer, in global aggregation step it just averages the model updates received from some random subset of clients to generate the new global model.

Algorithm 1 FedAvg algorithm

$w_0 \leftarrow$ Initial value
for $t \leftarrow 0, 1, 2, \ldots$ **do**
    $c \leftarrow \max(\lfloor C \times N \rfloor, 1)$
    $R_t =$ random set of c clients
    for each client $k \in R_t$ in parallel **do**
        $w_{t+1}^k = \text{UpdateFromClient}(k, w_t)$
    $w_{t+1} \leftarrow \Sigma_{k=1}^{N} \frac{n_k}{n} w_{t+1}^k$
function UpdateFromClient(k,w)
    Batch $\leftarrow$ (split P_k into batches of size B)
    for each local epoch i from 1 to E **do**
        for batch $b \in$ Batch **do**
            $w \leftarrow w - \eta * \nabla l(w, b)$
    return w to server

FedProx: This is an algorithm constructed by making slight change in FedAvg algorithm by adding a proximal term to every client's loss function. This proximal term allows each client models to differ from the global model addressing statistical heterogeneity. Remaining steps are same as FedAvg.

Algorithm 2 FedProx algorithm

$w_0 \leftarrow$ Initial value
for $t \leftarrow 0, 1, 2, \ldots$ **do**
    $c \leftarrow \max(\lfloor C \times N \rfloor, 1)$
    $R_t =$ random set of c clients
    for each client $k \in R_t$ in parallel **do**
        $w_{t+1}^k = \text{UpdateFromClient}(k, w_t)$
    $w_{t+1} \leftarrow \Sigma_{k=1}^{N} \frac{n_k}{n} w_{t+1}^k$
function UpdateFromClient(k,w)
    $w' = \text{argmin}_{w_0} F_k(w_0) + \frac{\mu}{2} \|w_0 - w\|^2$
    return w' to server

1.2 Challenges faced

There are a lot of challenges faced in federated learning due to the nature and memory limitation of devices. Some of the common and important challenges are:

- **Communication overhead**: Since federated networks consists of many communication rounds between server to client (in case of centralized federated learning)

and client to client (in case of decentralized federated learning) thus these communications have to be scheduled in such a way to minimize the communication costs and latency.

- **Heterogeneity** : Often federated networks consists of devices that may drop out of the network due to connectivity or power issue thus the network should be robust enough to handle these situations. Also, the data of different clients although similar is mostly non-i.i.d thus network should be designed to give efficient results in such cases also.

- **Privacy**: This is one of the most important concern in federated learning. Preserving privacy is an important property that federated algorithm should satisfy. Although, at first glance it may seem that privacy is preserved as only model updates is shared not the original model but it has been shown that there exists various schemes of extracting knowledge about the model through model updates only.

1.3 Applications of federated learning

Federated learning provides robust training of data for different clients by considering the data of every client without sharing this data with anybody. Federated learning is becoming more and more popular due to its great use in new technologies such as Industry 4.0, smart home, autonomous driving and virtual reality.

2 Synopsis of Papers

2.1 Federated Learning - Challenges, methods and future directions

2.1.1 Overview

This paper discuss about the major challenges faced in Federated learning setup and methods used to overcome them. The standard federated learning problem involves learning a single global model from data stored on remote devices, with only intermediate updates being communicated to central server. This problem is defined mathematically as

$$\min_w F(w),\ F(w) = \Sigma_{k=1}^{m} p_k F_k(w)$$

where m is the total number of devices/clients, $p_k \geq 0$ is the relative impact of each client on the global model ($\Sigma_{k=1}^{m} p_k = 1$), $F_k(w)$ is the local objective function of each client.

2.1.2 Major challenges

There are four major challenges faced in solving the above optimization problem.

Expensive communication: Generally, federated networks consists of large number of clients thus there is large communication bottleneck associated with this network. Thus, the model updates have to be communicated efficiently over the network this can be accomplished by reducing total number of communication rounds and reducing the size of transmitted messages at each round.

Systems heterogeneity: Devices (clients) may differ in a lot of ways like storage capacity, computational limits and communication capabilities due to hardware of devices, power level of each device, network connectivity, etc. It may be possible that only some of the devices are working at a particular instance due to either network condition or power level. Thus, federated learning network should anticipate low amount of participation, tolerate heterogeneous hardware, and robust to dropping of devices from the network.

Statistical heterogeneity: Data generated by each device can differ significantly from other devices both in terms of size and nature. Thus, generally in federated learning has to be designed so that it gives good performance for non-i.i.d data of devices.

Privacy concern: Federated learning protects data of each device by only sharing model updates instead of the actual data used by each device. However, privacy is not guaranteed even by sharing of model updates only.

2.1.3 Approach used for challenges:

Communication efficiency: There are three major ways of reducing communication bottleneck:

- Local updating: One of the method is to break a single problem to different problems that can be solved in parallel, thus global objective function is decomposed into different problems and then solved parallelly to provide speedup.

- Compression schemes: This is done to reduce the size of messages communicated by quantization and sparsification. However they do not work efficiently in case of low device participation problem.

- Decentralized training: This method is shown to be faster and reduce communication cost in case of high latency or low bandwidth networks.

Systems heterogeneity: The three major approach used to tackle this issue are:

- Asynchronous communication: Asynchronous schemes are more flexible to issues like dropping out of clients. However, they rely on delay assumptions which may violated in many applications.

- Active sampling: This involves exploiting the fact that only small subset of devices participate at each round of training and sampling their data at each round of training.

- Fault tolerance: This assigns more priority to those devices that are less likely to drop due to network issues by assigning bias for each device.

Statistical heterogeneity: These are the methods used to tackle the non-i.i.d nature of dataset of devices:

- Modeling heterogeneous data: This includes using techniques like **meta learning** and **multitask learning** to address data heterogeneity.

- Convergence guarantees for non-i.i.d. data: FedProx algorithm uses an interplay between systems heterogeneity and statistical heterogeneity. FedProx provides convergence guarantees for non-i.i.d. data case.

Privacy: This is one of the most important challenges in federated learning, federated learning network has to ensure privacy without dropping the accuracy too much. Global privacy requires that model updates are private to all third parties and local privacy which require they are also private to the server. Several **cryptographic algorithms** are used to encode model updates.

2.2 Decentralized Federated Learning: A Segmented Gossip Approach

2.2.1 Overview

Centralized federated learning networks relies on large nodes-to-server bandwidths, which is not the case in many practical applications. Further, centralized federated learning also suffers the risk of single point failure. This paper discuss a model segment level decentralized federated learning to tackle this problem making full utilization of node-to-node bandwidth. Model updates of one agent are communicated only to some particular agents in this approach.

2.2.2 Method

Consider a network with n agents. This paper introduces segmented pulling in which an agent collects different parts of model parameters from different agents and rebuild a mixed model by aggregation. Let $\mathcal{W}$ denote the model parameters, the agent breaks it in S non-overlapping segments as $\mathcal{W} = (\mathcal{W}[1], ..., \mathcal{W}[S])$, for each segment l, the agent chooses another agent j_l and selects the corresponding segment $\mathcal{W}_{j_l}[S_l]$ from it. This step is parallelized to make full use of bandwidth. Thus, the new model is constructed as$\mathcal{W}' = (\mathcal{W}_{j_1}[1], ..., \mathcal{W}_{j_S}[S])$. Instead of segmenting only one new model, we segment R new models and then aggregate them as

$$\mathcal{W}'[l] = \frac{\Sigma_{j \in P_l} |D_j| \mathcal{W}_j[l]}{\Sigma_{j \in P_l} |D_j|}$$

where P_l denote the set of agents which provide the segment l and $|D_j|$ denote the size of dataset of agent j. Then, we can combine these aggregated results to get the updated model.

This setup can handle the dropping of agents. If an agent goes offline then the request it sent to other agents are cancelled. If an online agent notices that some agent from whom it requested the update has went offline it can remake its request to some other agent and update the list of offline agents. Similarly, it can handle the participation of new agents easily.

2.2.3 Results

Important observations from results are:

- This method provides better convergence speed without compromising convergence accuracy.
- This method was found to be much more scalable than centralized scheme.
- It gave better results than basic gossip approach where no segmentation was used.

2.3 Multiple diseases and pest detection based on federated learning and faster r-cnn

2.3.1 Overview

This paper discuss pest detection technique using federated learning and faster region convolutional neural network. Federated learning is used to tackle the issue of high data storage, insufficient data for each orchards and communication costs. FedAvg algorithm is improved to provide better convergence and training speed. CNN architecture is modified to improve the results.

2.3.2 Problem details

There are multiple orchard farms and using their data we have to develop the model for pest and diseases detection. Different orchards are based on different geographical locations. Data of each orchard is **unbalanced** and **insufficient** for e.g., an orchard A does not have data sample for some disease x while other orchard has data for disease x but no data for some other disease y.

2.3.3 Approach used

This paper employs a federated learning network where each orchard is a client and there is a global server and there is no sharing of data between the orchards.

Federated learning: FedAvg algorithm performs better when data characteristics of each client are similar. It does not give good convergence for unbalanced data. This paper improves the standard FedAvg algorithm by adding a restriction term to the client loss function, allows each client model to differ from the global model.

CNN architecture: Faster R-CNN model uses VGG convolutional network, but this doesn't provide high accuracy for small targets due to gradient explosion and gradient dispersion. Thus, the VGG convolutional network is replaced by ResNet-101 architecture to solve this problem. First 92 layers of ResNet-101 are used for feature extraction and remaining 9 layers are used for target detection. L2-normalization is used due to large size difference in feature maps.

Online Hard Example Mining (OHEM): OHEM is used as an effective method to improve the detection accuracy of CNNs. It re-enters those cases that are not accurately

identified (difficult cases) by the network to improve the performance. In this paper, loss values are calculated and then sorted from large to small, then first K maximum loss values are selected as difficult cases and added to network for further training.

Some other techniques are also used to improve performance. **Soft Non-maximum suppression** is used instead of Non-maximum suppression, since the latter may fail to report some cases where pest is detected as true. Sample expansion is used to increase dataset size by adding more images by performing operations like rotation, scaling, cropping, flipping, etc., for better training of model.

2.3.4 Results

Important observations from results were:

- Model can accurately identify diseases in full fruit diseases while not misclassifying the good fruits nearby.

- Each orchard was able to recognize all diseases while some diseases were missing from each orchard dataset, this generalization in presence of data imbalance was achieved due to Federated learning.

3 Comparison of federated learning algorithms

This experiment aims to compare the performance of two major federated learning algorithms FedAvg and FedProx. First, the variation of accuracy of these algorithms are tested against various parameters involved in the algorithms. Then the accuracy is compared of these algorithms for same parameter settings.

3.1 Dataset used

The dataset used to evaluate performance of FedAvg and FedProx algorithms is MNIST dataset. MNIST is a dataset of handwritten digits from 0 to 9. It is widely used in computer vision and deep learning. It is a dataset of 60,000 small square 28*28 pixel grayscale images.

3.2 Method

This is a comparison between FedAvg and FedProx algorithms for same model architecture and other parameters.

3.2.1 FedAvg

FedAvg is the most basic Federated learning algorithm. It has been shown to work well for non-convex problems but doesn't guarantee convergence and may diverge in many practical situations where data is heterogeneous.

3.2.2 FedProx

FedProx is an improvement on FedAvg to give better results when data is non-i.i.d., it adds a proximal term to the loss function which allows variance of each client model from aggregate model. Proximal strength controls how much can the clients model vary from the global model.

3.3 Results

Each client dataset consisted of data from a particular MNIST label. Initially, number of clients was set to 4 and then increased to study the variation of accuracy. The keras model consisted of simple architecture with Dense and softmax layers. Stochastic Gradient Descent (SGD) as optimizer and Sparse Categorical Accuracy as metrics was used. Accuracy was compared of FedAvg and FedProx for different parameters like number of clients, epochs, learning rate of client, learning rate of server, etc., as discussed below. The code is given **here**.

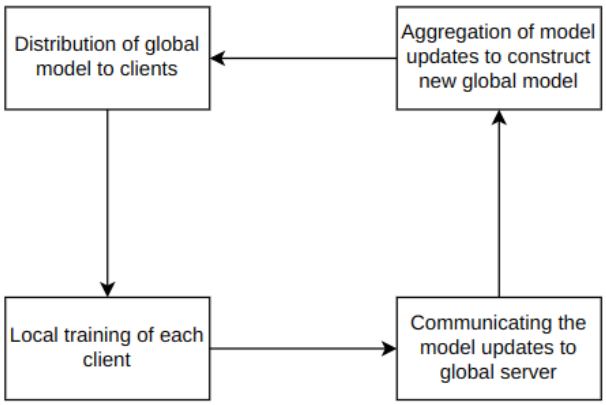

Figure 3: One global round of model updates

First, FedAvg algorithm was tested for 4 clients by changing clients, epochs, learning rate of client, learning rate of server, etc. The results are shown below, where server optimizer is the learning rate of global server optimizer, client optimizer is learning rate of each client optimizer (assumed identical in this case), number of epochs is number of iterations for each client in one global round, and total rounds is the total number of global rounds. A single global round consists of distributing global model to each client, local training of each client, collecting and averaging model updates, and producing updated global model at server as shown in Fig. 3.

For the above experiment, the learning rate of server was chosen to be 1 and for client to be 0.02. We can observe from the results in Table 1 that as number of epochs increase and number of total rounds increase there is significant increase in categorical accuracy as expected. Now we keep number of epochs and total rounds to be constant 10 and 20 respectively and change the learning rate of both server and clients.

Number of epochs	Total rounds	Categorical accuracy
5	10	30.0%
5	20	56.7%
10	10	54.2%
10	20	81.6%

Table 1: Variation of accuracy for FedAvg with number of epochs and total rounds

Server Optimizer	Client Optimizer	Categorical accuracy
1	0.02	81.6%
1.5	0.02	89.0%
1	0.05	51.8%

Table 2: Variation of accuracy for FedAvg with learning rate

We can observe from the results of table 2 that increase in server learning rate increases accuracy while slight increase in client learning rate decreases accuracy. Now, we will make same change in parameters but for FedProx algorithm, now we will have another parameter Proximal strength. The code for FedProx algorithm is given **here**

Number of epochs	Total rounds	Proximal Strength	Categorical accuracy
5	10	1	33.7%
5	20	0.5	38.0%
10	10	1.5	42.0%
10	10	2.0	46.3%
10	20	2.0	93.3%

Table 3: Variation of accuracy for FedProx with different parameters

The above results are for 4 clients, server learning rate = 1 and client learning rate = 0.02. We can observe from the results in Table 3 that, again accuracy increases as either number of total rounds increase, also as proximal strength increases the accuracy also increases. We can also observe that in general FedProx gives better accuracy than FedAvg for same learning rates and number of epochs. Now we study the variation in accuracy with change in number of clients for both FedAvg and FedProx.

Number of clients	FedAvg accuracy	FedProx accuracy
4	81.6%	93.3%
8	78.1%	91.9%
16	75.5%	89.3%

Table 4: Variation of accuracy for FedAvg and FedProx with number of clients

We can observe from the results in table 4 that as number of clients increase accuracy of both algorithms decreases, for keeping other parameters same as we would require more training for more number of clients.

3.4 Observations from the results obtained

- Accuracy of both algorithms increase if total number of rounds and number of epochs are increased keeping other parameters same
- As proximal strength increase accuracy of FedProx algorithm increases significantly
- As number of clients increase accuracy of both algorithms drop
- For same parameters FedProx gives better accuracy than FedAvg

4 Conclusion

Federated learning is an emerging field in machine learning. It has vast range of applications especially in Internet of Things (IoT). Federated learning is an important part of Industry 4.0. Although, it is primarily machine learning field there are important applications of other fields in federated learning also like communication networks, network theory for creating federated network and communication, cryptography for privacy needs, etc. There are a lot of variations in federated learning by changing the network, algorithms used, encryption, local training of devices, etc. Thus, this provides us with a large number of models that we can construct for our devices. With the help of federated learning we can achieve better results for pest classification because of limitations of the devices used at orchards but federated network will help them to learn more complex features and also the features from other orchards.

References

[1] A. Imteaj, U. Thakker, S. Wang, J. Li and M. H. Amini, "A Survey on Federated Learning for Resource-Constrained IoT Devices," in IEEE Internet of Things Journal, vol. 9, no. 1, pp. 1-24, 1 Jan.1, 2022, doi: 10.1109/JIOT.2021.3095077.

[2] Hu, Chenghao & Jiang, Jingyan & Wang, Zhi. (2019). Decentralized Federated Learning: A Segmented Gossip Approach.

[3] T. Li, A. K. Sahu, A. Talwalkar and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," in IEEE Signal Processing Magazine, vol. 37, no. 3, pp. 50-60, May 2020, doi: 10.1109/MSP.2020.2975749.

[4] F. Deng, W. Mao, Z. Zeng, H. Zeng and B. Wei, "Multiple Diseases and Pests Detection Based on Federated Learning and Improved Faster R-CNN," in IEEE Transactions on Instrumentation and Measurement, vol. 71, pp. 1-11, 2022, Art no. 3523811, doi: 10.1109/TIM.2022.3201937.

[5] T. Gafni, N. Shlezinger, K. Cohen, Y. C. Eldar, and H. V. Poor, "Federated Learning: A signal processing perspective," IEEE Signal Processing Magazine, vol. 39, no. 3, pp. 14–41, May 2022, doi: 10.1109/msp.2021.3125282.

[6] T. Zhang, L. Gao, C. He, M. Zhang, B. Krishnamachari, and S. Avestimehr, "Federated Learning for Internet of Things: Applications, Challenges, and Opportunities," CoRR, vol. abs/2111.07494, 2021, [Online]. Available: https://arxiv.org/abs/2111.07494

[7] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," 2016.

[8] https://ai.googleblog.com/2017/04/federated-learning-collaborative.html