Learning Directed Collaboration Graphs for Peer-to-Peer Personalized Learning

Thesis

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By

Xue Zheng, B.Eng.

Graduate Program in Electrical and Computer Engineering

The Ohio State University

2021

Thesis Committee:

Dr. Parinaz Naghizadeh, Advisor

Dr. Irem Eryilmaz

Dr. Jia Liu

Abstract

We study a fully decentralized distributed learning problem. In this setting, the agents collaboratively train machine learning models by directly exchanging information with each other, without the aid of a central server or coordination mechanism. Our goal is to study the emergence of a communication or information exchange network between these agents to enable collaborative, yet personalized learning, particularly when the learners are heterogeneous. Specifically, when learners' local datasets are non-IID, a collaboratively trained *global* model (one that would minimize the sum of losses across all agents) may sacrifice the *local* performance of several agents on their local datasets. To address this issue and enable personalized learning, we propose an algorithm that enables each agent to optimize its local performance by identifying "helpful" neighboring agents and only requesting models from them.

In particular, at each time step, an agent trains its local model based on its local dataset, while also obtaining the model trained by one of its neighboring agents in the collaboration graph. The agent then mixes its local model with the neighbor's model, where the neighbor's model is weighted by the edge weight of the collaboration graph. If the mixture model attains better/worse performance than the agent's local model, the corresponding edge weight in the collaboration graph is increased/decreased. Our method leads to the emergence of a directed collaboration graph. We show that learning over this graph brings two main advantages: (1) compared to (centrally

coordinated) federated learning, it achieves personalized learning by selectively communicating with helpful neighbors, and (2) compared to fully connected or arbitrary information exchange graphs, it reduces communication overhead. We provide analytical results on the generalization error bounds of our algorithm, and verify its performance through numerical experiments on the Fashion-MNIST dataset.

To Dr. Naghzadeh and my friends, thank you for your help and encouragement.

Acknowledgments

During my research, I received a lot of help from my fellow researchers and my friends.

Firstly, thanks to my advisor Dr. Naghzadeh for her patient guidance on every step of my research. This guidance is crucial to me, who has no previous research experience. She also spends a lot of time discussing research progress with me every week and giving me encouragement and advice.

Secondly, thanks to my defense committee for their patience and careful suggestions for my research and thesis.

Finally, thanks to my friends for their support. They are always by my side and always create a relaxed and good life atmosphere for me.

Vita

2018 ..... B.Eng., Information and Electrical Engineering, Harbin Institute of Technology at Weihai.

2020-present ..... M.Sc., Electrical and Computer Engineering, The Ohio State University.

Fields of Study

Major Field: Electrical and Computer Engineering

Table of Contents

Page

List of Tables

Table **Page**

List of Figures

Figure **Page**

Chapter 1: Introduction

Federated Learning (FL) is a Machine Learning (ML) paradigm in which a number of agents/clients collaboratively train an ML model. Each agent has access to a local dataset and uses it to update its local ML model iteratively. The updated local models will be sent back to the central server. The goal of FL is to suitably combine these local models, with the help of a central server, to ultimately obtain a global model that would minimize the average error across all agents' datasets. Formally, the optimization problem solved through FL is

$$\min_{h \in \mathcal{H}} \quad \frac{1}{K} \sum_{i=1}^{K} \mathcal{L}_{D_i}(h) \tag{1.1}$$

where K is the number of agents participating in each round, and $\mathcal{L}_{D_i}(h)$ is the loss incurred by model h on the local dataset D_i of agent i. Such distributed approach to learning enables cooperation on learning without centralizing the training data, and therefore reduces communication overhead while respecting agents' privacy.

One of the most popular methods in this paradigm is FedAvg [14]. In FedAvg, the agents first begin by performing multiple updates of their current ML models so as to minimize the loss on their local datasets. Then, at each communication round, a central server collects the local models from a subset of these agents and takes a (weighted) update of their local models, reporting the result back to all clients, to be

used as the basis for their next rounds of local training. The method is communication efficient and has been shown to have desirable properties, but continues to have some drawbacks. One of these is the sensitivity of FedAvg to heterogeneity of data distributions on each client [9]: FedAvg leads to the training of a (consensus) model which may not perform well, locally, on agents' heterogeneous datasets. This has led to interest in developing methods that combine the best of both worlds: improving learning rate by leveraging the distributed computation power and datasets of multiple learners, while maintaining the specialized and personalized nature of each (heterogeneous) agent's local model.

One of the works that has attempted to strike this balance is by Fallah et al. [6], which combines the frameworks of FL and Meta Learning [7]. In Meta Learning, an agent's goal is to learn a critical or base model which is shared between several tasks, and can be used to train an ML model when facing new tasks by performing a limited number of gradient updates starting from this base model. In the context of personalized learning in [6], FL is used to obtain an initial shared model, and then Meta Learning is used to refine this into a personalized local model. An alternative idea has been explored by Zec et al. [22], who propose a "mixture of experts" approach, in which each agent obtains a personalized model by appropriately mixing their specialized local model and the generalized global model obtained from FedAvg (where the two models are combined using a weighted sum). A closely similar idea for personalized learning has been explored by Deng et al. [5], where agents train local models, while also contributing to the training of a global model. While the algorithms in [6, 22, 5] allow for training of personalizaed models, they continue to

leverage a shared global model which is collaboratively trained with the help of a centralized server.

In contrast to these works, we are interested in methods for collaborative training of personalized models in a fully decentralized setting (i.e., without a central server) through peer-to-peer communication between agents. The communication topology is captured by a (directed, weighted) graph in this setting, with agents as the nodes and edges representing the information exchange. These edges may be directed and weighted to represent unidirectional information exchange and the usefulness of neighboring nodes' information for local optimization, respectively.

In particular, we propose a learning algorithm through which an agent iteratively adjusts its communication links (including the weights) with other agents, based on its assessment of how "useful" each peer's communicated model has been, to improve the agent's personalized model. The output of our algorithm is a *weighted and directed collaboration graph*, as well as collaboratively trained, personalized models, at each agent.

Decentralized and peer-to-peer learning has been studied in many prior works, (see e.g., [12, 3]). A main difference of our approach with this line of work is in the final desired learning outcome: much of the existing works study consensus on a shared global model, while we are interested in collaborative training of personalized models.

Peer-to-peer *personalized* learning has only been studied recently in [1], [18], and [21]. One key difference of our work with recent works in [1] and [18] is that these prior works assume the communication graphs to be given and fixed a priori. In contrast, we study a graph that emerges adaptively as part of the learning process.

To the best of our knowledge, the only prior work exploring a similar problem of adaptively learning a collaboration graph is that of Zantedeschi et al. [21]. The work in [21] adjusts the collaboration graph's edges based on the similarity between the local models of the two nodes; this leads to an *undirected* collaboration graph. In contrast, we propose a different method for identifying similar neighbors, which ultimately leads to learning a *directed* collaboration graph. We show that learning under such directed collaboration graphs can outperform undirected ones.

Intuitively, directed collaboration graphs can bring the following benefits to peer-to-peer personalized learning. As also noted in prior works on personalized learning, e.g. [5], there are many factors that influence whether an externally trained model is beneficial to the local learning of any given node, including the external model's accuracy, the dataset size from which the model is obtained, and the difference in the data distribution between the two nodes. Therefore, the local models of a pair of nodes may have *asymmetric* effects on each other. For example, a node with a small dataset and low accuracy may rely more on the model of a more accurate neighboring node, while the neighboring node, who has a large dataset and a high accuracy model, will prefer to maintain its own local model. We therefore see that it is not enough to simply rely on the similarity of the models to determine the edge weight between two nodes and obtain an undirected graph. Motivated by these observations, we propose a method for learning directed graphs to solve a more general collaborative learning problems — a situation where the local models of a pair of nodes may have different "values" to each other.

1.1 Thesis Organization

The remainder of this thesis is organized as follows. In Chapter 2, we review related work on both Federated Learning and personalized learning. We present our proposed learning algorithm in Chapter 3, followed by theoretical guarantees on the generalization error bounds for our algorithm in Chapter 4. We provide numerical experiments and compare our algorithm against a number of baselines in Chapter 5. We conclude with a discussion of our main findings, and potential future directions, in Chapter 6.

Chapter 2: Related Work

2.1 Distributed Learning: Federated Learning and Decentralized Learning

In recent years, there has been increasing interest and research on Federated Learning (FL) as an architecture for distributed learning. This architecture consists of a central server and a number of distributed agents/clients. The central server repeatedly collects the local models from (a subset of) these distributed agents, combines them together to obtain a global model, and communicates this shared model back to the agents. The datasets used for training the models are kept locally on each agent, which maintains the privacy of the data. The agents only upload local models instead of data to the central server, reducing communication consumption and increasing communication efficiency.

Among the proposed algorithms in this architecture of distributed training with central sever, FedAvg in [14] has received widespread attention since its publication in 2017. Fedreated learning is also closely related to Meta Learning [7]: because federated learning and meta-learning both use cross-device collaborative training, some works have focused on combining meta-learning and federated learning, such as FedMeta in [4] and Dif-MAML in [10].

Another approach to distributed learning is to forego the central server, and instead learn in a fully decentralized/peer-to-peer architecture. The communication topology in this setting can be represented by a connected graph, where each agent is a node of the graph. Decentralized learning can offer advantages over federated learning. In particular, compared with federated learning with a central server, fully distributed learning is more likely to overcome communication bottlenecks. The reason is that the central server needs to communicate with each agent regularly, so a powerful central server is necessary. However, the central server may encounter a communication bottleneck, e.g. as shown in [13], when the number of agents increases. In such a situation, if a relatively sparse connected graph can be used for peer-to-peer communication between agents, it will help overcome the communication bottleneck problem caused by the central server. The research on fully decentralized learning has explored both undirected graphs (such as [13]) as well as directed graphs (such as [2] and [16]).

2.2 Personalized Learning

In the federated learning research mentioned above, most research aims to obtain an optimal global model. Such shared model will help agents collaboratively obtain a model that will have a good performance given IID local datasets. However, when we consider the heterogeneity in agents' local data, it becomes more difficult to collaboratively train a single global model based on the union of non-IID datasets. In addition, given non-IID local datasets, the global model may poorly perform on several agents' local datasets. For instance, as shown by Yu et al. [20], the error bound of local SGD will worsen in the case of non-IID data. To enable each agent to obtain

a better local model in a non-IID setting, a number of works have proposed focusing on personalized learning. We elaborate on some of the approaches proposed in the existing literature below.

2.2.1 Local Fine Tuning and Meta-Learning

This method is mainly based on transferring the global model obtained by federated learning to the selected agent. Then personalization is achieved locally, by fine-tuning the global model on the agent's local dataset. Because in federated learning, the chosen agents will perform several rounds of model updates on their local data after receiving the global model from the central server, local fine-tuning is internally combined with the federated learning algorithm. In addition, this fine-tuning method is often paired with Meta-learning in recent studies. For example, Fallah et al. [6] proposed a personalized federated learning algorithm, where the model is learned under the Model-Agnostic Meta-Learning (MAML) framework.

2.2.2 Multi-Task Learning

Multi-task learning is also used in personalized learning because we can regard model optimization on different agents in personalized learning as a parallel to learning different tasks in multi-task learning. For example, Smith et al. [17] proposed the MOCHA algorithm for multi-task federated learning. In addition, in multi-task learning, when the local data set is small, it can communicate with similar datasets to obtain external information to help it get better models, as done in works such as [23] and [11]. The ideas can also be applied to personalized learning. Among them, Zhang et al. [23] also uses the meta-learning ideas mentioned in the previous subsection.

2.2.3 Mixed Model

In addition, some studies such as ([22], [5], and [8]) try to mix the global model and the local model to achieve the personalization of the local model. These studies are all carried out under federated learning with a central server. The local agent will maintain the local model trained on the local dataset. After receiving the global model from the central server, the agent will mix the local and global models (e.g., through a weighted sum) under certain conditions to achieve personalization. The mixing method and ratio of each agent will be learned during training on its local dataset.

Chapter 3: Model and Proposed Algorithm

We next present our decentralized learning algorithm for adaptively learning a collaboration graph, while also using this graph to collaboratively train ML models. Our algorithm will achieve this by learning a directed graph and allowing nodes to share information with each other according to the directed graph. In particular, each agent will communicate with other agents in its local neighborhood, requesting the models trained by these neighbors; the graph is then updated to place more weight on neighbors that have been deemed to be more similar, so as to guide future information exchange decisions. Our goal is to obtain a directed graph: when the information is deemed to be useful by an agent in a pair, only this agent, who can benefit from communication, will establish a directed link to the other to get the model from the helpful neighboring node in the future as well; a reciprocal relation may never be established. Further, when the models of both parties are deemed useless to each other, the two nodes will become less likely to communicate in the future; this choice saves communication costs and prevents nodes from being disturbed by useless information (leading to better personalized models).

3.1 Learning Objective

We begin by formalizing the objective of each agent $i \in \{1, 2, ...N\}$. N is the number of agents selected every time. As mentioned earlier, we are focused on the problem of personalized learning. This means that each agent i is attempting to train an ML model that would minimize loss on its own local data.

Formally, denote the local dataset of node $i \in \{1, 2, ...N\}$ by D_i. For a given ML model $h \in \mathcal{H}$, we use $\mathcal{L}_{D_i}(h)$ to denote the loss incurred by model h on the local dataset D_i. The goal of agent i is to train a model h to minimize this loss.

Given a weighted collaboration graph, the final model obtained by each node i will be a weighted mixture of its local model h_i with the local models h_j of agents j who are its neighbors in the collaboration graph. The weights are determined by the edge weights in the collaboration graph. Formally, let α_{ij} denote the weights of a link connecting agent i to agent j in the directed collaboration graph, and let α_{ii} denote the weight of node i on its self-loop in the graph. We assume these weights are such that $\alpha_{ii} + \sum_j \alpha_{ij} = 1$. Then, the mixed model obtained by agent i will be given by $\alpha_{ii} h_i + \sum_j \alpha_{ij} h_j$.

Putting these together, agent i chooses its local model h_i such that

$$h_i^t = \arg\min_{h \in \mathcal{H}} \quad \mathcal{L}_{D_i}(\alpha_{ii} h + \sum_j \alpha_{ij} h_j^{t-1}) \ . \tag{3.1}$$

Remark: contrasting with the agent goals in FL. It is worth contrasting this objective function with that of Federated Learning (or other consensus learning algorithms). In those problems, all agents aim to learn a common model h such that

$$h^* = \arg\min_{h \in \mathcal{H}} \sum_i \mathcal{L}_{D_i}(h) \ . \tag{3.2}$$

In is easy to see that this global objective may not align with personalized/local accuracy. A consensus learning can still be desirable if the datasets D_i are IID, or when agents with non-IID local datasets believe that they will need to make decisions on data not in their current datasets in the future. Nonetheless, our focus here is on personalized learning by agents who have potentially similar datasets to only a subset of other agents, and are interested in minimizing their local loss, as shown in (3.1).

3.2 Proposed Algorithm

Our algorithm proceeds in two steps. The first step obtains the directed graph, and the second step trains the ML models at each node, collaboratively, while exchanging information based on the obtained graph. We detail each step below.

3.2.1 Obtain the directed graph

At the beginning of our algorithm, the collaboration graph is initialized arbitrarily. For instance, the graph may be initialized to be fully connected and have weights of 0.5 on all edges. Starting from this initialization, the following steps are performed repeatedly.

1. Train the ML model of each node i on its local data D_i separately (e.g., using K stochastic gradient descent updates). The goal of this training is to minimize the local loss at node i.

2. Based on the current graph, the obtained models from step 1 are transferred between neighboring nodes. Specifically, if node i has a directed edge with weight

α_{ij} to node j, node i requests the local model h_j of node j. Then, the neighbor model h_j and the local model h_i are mixed according to the corresponding weight to yield the mixture $\bar{h}_i := \alpha_{ii}h_i + \alpha_{ij}h_j$.

3. The mixed model $\bar{h}_i$ from step 2 is fine-tuned on the local dataset D_i of node i (e.g. by conducting K' stochastic gradient descent updates), to minimize the local loss at node i.

4. The graph weights α_{ij} are updated by comparing the accuracy of h_i (the initial local model from step 1) with that of $\bar{h}_i$ (the fine-tuned model of step 3). If the fine-tuned accuracy is greater than the local accuracy, the weight of relying on the model of agent j will increase. On the other hand, if the fine-tuned accuracy is worse than the local accuracy, the weight of relying on the model of agent j will decrease.

The stopping condition can be either when no graph weight is changing substantially (which indicates that the graph has converged), or after a certain number of iterations (to limit computation power used).

3.2.2 Train on the directed graph

Once a directed graph is obtained from the previous step, the algorithm conducts collaborative learning over this directed graph. Specifically, we freeze the mixing weights (i.e., keep the edge weights in the graph fixed), and perform the following steps repeatedly. The steps are largely similar to those followed when obtaining mixed models when training the graph.

1. Transfer models between neighboring nodes according to the directed edges in the collaboration graph. Specifically, if node i has a directed link to node j, then node j shares its current local model h_j with node i.

2. Train the ML model h of each node i on its local data D_i separately to minimize the local loss at node i evaluated at the mixed model $\bar{h}_i = \alpha_{ii}h + \sum_j \alpha_{ij}h_j$. Note that the h_j's stay fixed during these updates. Conduct the specified number of updates (e.g., using K stochastic gradient descent updates on the initial model at that node). Set $h_i = \bar{h}_i$. Go back to step 1.

Chapter 4: Analytical Results

In this chapter, we show a generalization error bound for the model obtained in step 2 of our algorithm (i.e., for a fixed graph). The obtained bound allows us to observe how the selected edge weights, together with the differences or similarities of different local datasets, affect the generalization ability of the obtained models.

Our proof is closely similar to that of [5], which analyzed the generalization error of their proposed adaptive personalized federated learning (APFL) algorithm. As mentioned in the introduction, the APFL algorithm enables personalized learning in a Federated Learning paradigm, where agents train their local models, while also contributing to the training of a global model. This is done through agents training a weighted mixture of their local models with the global FL model. We use a similar mixed model in our algorithm, with the main differences that the mixing is done with the local models of several other agents directly, and weighed by a similarity measure calculated through learning. We extend the proof of [5] to this decentralized learning scenario.

Formally, define the local true risk minimizer of agent i by

$$h_i^* = \arg\min_{h \in \mathcal{H}} \quad \mathcal{L}_{D_i}(h) \ , \tag{4.1}$$

where $\mathcal{L}_{D_i}(h) = \mathbb{E}_{(\mathbf{x},y)\sim D_i}[l(h(\mathbf{x}), y)]$, and $l(\cdot)$ is a loss function; we consider the squared hinge loss in classification tasks, which is $l(h(\mathbf{x}), y) = (\max\{0, 1 - yh(\mathbf{x})\})^2$.

Further, define the local empirical risk minimizer of agent i by

$$\hat{h}_i^* = \arg\min_{h\in\mathcal{H}} \quad \hat{\mathcal{L}}_{D_i}(h) \tag{4.2}$$

The mixed true and empirical risk minimizers are defined similarly

$$h_{loc,i}^* = \arg\min_{h\in\mathcal{H}} \quad \mathcal{L}_{D_i}(\alpha_{ii}h + \sum_j \alpha_{ij}\hat{h}_{loc,j}^*) \tag{4.3}$$

$$\hat{h}_{loc,i}^* = \arg\min_{h\in\mathcal{H}} \quad \hat{\mathcal{L}}_{D_i}(\alpha_{ii}h + \sum_j \alpha_{ij}\hat{h}_{loc,j}^*) \tag{4.4}$$

We also use the following definition (from [5]).

Definition 1. *Let $\mathcal{S}$ be a fixed set of samples, and consider a hypothesis class $\mathcal{H}$. The worst-case disagreement between two pairs of models is defined as*

$$\lambda_{\mathcal{H}}(S) = \sup_{h,h'\in\mathcal{H}} \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x},y)\in\mathcal{S}} |h(\mathbf{x}) - h'(\mathbf{x})| \ .$$

The generalization bound for our proposed decentralized personalized learning algorithm is as follows.

Theorem 1. *Let $\mathcal{H}$ be a hypothesis class with a VC dimension d. Assume the loss function l is Lipschitz continuous with a constant G and bounded on $[0, B]$. Then, with probability at least $1-\delta$ there exists a constant C such that the risk of the mixed model $\bar{h}_i^{(t+1)} = \alpha_{ii}\hat{h}_{loc,i}^{*t} + \sum_j \alpha_{ij}\hat{h}_j^{*t}$ on the local distribution D_i is bounded by*

$$\begin{aligned}\mathcal{L}_{D_i}(\bar{h}_i) \leq N\alpha_{ii}^2[\mathcal{L}_{D_i}(h_i^*) + 2C\sqrt{\frac{d + \log(1/\delta)}{m_i}} + G\lambda_{\mathcal{H}}(S_i)] \\ + N\sum_j \alpha_{ij}^2[\hat{\mathcal{L}}_{D_j}(\hat{h}_j^*) + C\sqrt{\frac{d + \log(1/\delta)}{m_j}} + B||D_i - D_j||_1] \ .\end{aligned} \tag{4.5}$$

where N is the number of nodes, m_i is the size of the training data at node i, S_i is the training data drawn from D_i, and $||D_i - D_j||_1 = \int |P_{(\mathbf{x},y)\sim D_i} - P_{(\mathbf{x},y)\sim D_j}| \mathrm{d}\mathbf{x}\mathrm{d}y$ is the difference between distributions D_i and D_j.

Proof. Starting from the risk $\mathcal{L}_{D_i}(\bar{h}_i)$, we have

$$\begin{aligned}
\mathcal{L}_{D_i}(\bar{h}_i) &= \mathcal{L}_{D_i}(\alpha_{ii}\hat{h}^*_{loc,i} + \sum_j \alpha_{ij}\hat{h}^*_{loc,j}) \\
&= \mathbb{E}_{(\mathbf{x},y)\sim D_i}[(\max\{1 - y(\alpha_{ii}\hat{h}^*_{loc,i} + \sum_j \alpha_{ij}\hat{h}^*_{loc,j})\})^2] \\
&= \mathbb{E}_{(\mathbf{x},y)\sim D_i}[(\alpha_{ii}\max\{1 - y\hat{h}^*_{loc,i}\} + \sum_j \alpha_{ij}\max\{1 - y\hat{h}^*_{loc,j}\})^2] \\
&\leq N\alpha_{ii}^2\mathbb{E}_{(\mathbf{x},y)\sim D_i}[(\max\{1 - y\hat{h}^*_{loc,i}\})^2] + N\sum_j \alpha_{ij}^2\mathbb{E}_{(\mathbf{x},y)\sim D_i}[(\max\{1 - y\hat{h}^*_{loc,j}\})^2] \\
&= N\alpha_{ii}^2\mathcal{L}_{D_i}(\hat{h}^*_{loc,i}) + N\sum_j \alpha_{ij}^2\mathcal{L}_{D_i}(\hat{h}^*_{loc,j}) \qquad (4.6)
\end{aligned}$$

Next, using the uniform VC dimension error bound over $\mathcal{H}$ [15], we know

$$|\mathcal{L}_{D_i}(h) - \hat{\mathcal{L}}_{D_i}(h)| \leq C\sqrt{\frac{d + \log(1/\delta)}{m_i}}, \quad \forall h \in \mathcal{H} \ . \qquad (4.7)$$

Then, following techniques similar to those in [5], we can get

$$\begin{aligned}
\mathcal{L}_{D_i}(\hat{h}^*_{loc,i}) &\leq \mathcal{L}_{D_i}(h^*_i) + 2C\sqrt{\frac{d + \log(1/\delta)}{m_i}} + \hat{\mathcal{L}}_{D_i}(\hat{h}^*_{loc,i}) - \hat{\mathcal{L}}_{D_i}(\hat{h}^*_i) \\
&\leq \mathcal{L}_{D_i}(h^*_i) + 2C\sqrt{\frac{d + \log(1/\delta)}{m_i}} + G\lambda_{\mathcal{H}}(S_i) \ . \qquad (4.8)
\end{aligned}$$

Lastly, from Lemma 1 in [5] we know that

$$\mathcal{L}_D(h) \leq \mathcal{L}_{D'}(h) + B||D - D'||_1 \ . \qquad (4.9)$$

This in turn means that

$$\mathcal{L}_{D_i}(\hat{h}^*_{loc,j}) \leq \mathcal{L}_{D_j}(\hat{h}^*_{loc,j}) + B||D_i - D_j||_1 \ . \qquad (4.10)$$

Using (4.7), we also have

$$\mathcal{L}_{D_j}(\hat{h}^*_{loc,j}) \leq \hat{\mathcal{L}}_{D_j}(\hat{h}^*_{loc,j}) + C\sqrt{\frac{d + \log(1/\delta)}{m_j}} \tag{4.11}$$

Substituting equations (4.8), (4.10), and (4.11), in (4.6), we get:

$$\begin{aligned}\mathcal{L}_{D_i}(\bar{h}_i) \leq N\alpha_{ii}^2\Big(\mathcal{L}_{D_i}(h_i^*) + 2C\sqrt{\frac{d + \log(1/\delta)}{m_i}} + G\lambda_{\mathcal{H}}(S_i)\Big) \\ + N\sum_j \alpha_{ij}^2\Big(\mathcal{L}_{D_j}(\hat{h}^*_{loc,j}) + C\sqrt{\frac{d + \log(1/\delta)}{m_j}} + B||D_i - D_j||_1\Big)\end{aligned} \tag{4.12}$$

This completes the proof. □

Interpretation of the bound. Intuitively, the bound indicates that the if D_i and D_j are similar, then mixing with neighbor j will not increase the risk at node i by much. Therefore, the mixing weight α_{ij} can be made larger. Similarly, if the local model of node j is such that the node has high local empirical risk $\hat{\mathcal{L}}_{D_j}(\hat{h}^*_j)$ itself, then a large mixing weight with this neighbor will increase the risk at node i, too. Lastly, it is beneficial to mix with neighbors who have larger local datasets m_j. Our algorithm for learning the collaboration graph adjusts the weights following similar logic: intuitively, it tries to increase the mixing weights with neighbors that have similar, as well as well-performing, local models.

Chapter 5: Experiments

In this chapter, we illustrate the performance of our algorithm using numerical experiments. The Fashion-MNIST dataset was used in these experiments. To verify whether our proposed method of learning a directed graph performs well in terms of yielding accurate personalized models under non-IID data conditions, we will compare our method with three baselines: 1. A fully connected graph, 2. training with local data only, and 3. distributed learning on an assumed collaboration graph based on knowledge about the nature of the local datasets. We compare our algorithm with these baselines under the settings of non-IID and IID local training datasets.

Below, we first detail our experiment setup, including the dataset, the selection process of the training data, and the baselines, followed by our results.

5.1 Dataset

The Fashion-MNIST data set [19] is used in this experiment. It has 70,000 experimental examples, including 60,000 training examples and 10,000 test examples. All of them are 28×28 grayscale pictures and belong to one of ten clothing classes.

Our method of obtaining the non-IID local datasets for each node is based on [14]. Each non-IID dataset combines a majority dataset and a minority dataset. Each local dataset is parameterized by the number of examples N, a proportion p,

and a number n of classes in the majority dataset. To obtain a local dataset with N samples and parameters p and n for a given node, we first divide the Fashion-MNIST dataset into 10 sets, each containing one of the different labels. Then, denote the set labels belonging to the majority dataset by $a_1, a_2, \ldots, a_n$ and the set labels belonging to the minority dataset by $b_1, b_2, \ldots, b_{10-n}$. We sample $\frac{Np}{n}$ datapoints from the classes $a_1, a_2, \ldots, a_n$ to get the *majority* dataset, and sample $\frac{N(1-p)}{10-n}$ datapoints from the classes $b_1, b_2, \ldots, b_{10-n}$ to get the *minority* dataset. Note that with a choice of $p = \frac{n}{10}$, this selection process will lead to IID local datasets. On the other hand, selecting a large p will ensure that the local data consists largely of a majority class, allowing us to obtain non-IID local datasets. We select both the training dataset and the testing dataset of each node using the described procedure. Each node contains 1000 examples in its training data and 300 examples in its testing data.

5.2 Baselines

We compare our algorithm against three baselines, as detailed below:

1) Undirected fully connected graph: Our first baseline is one of decentralized learning over a fully connected graph; this is similar to conducting federated learning, but without a central server. Any two nodes will communicate with each other in a peer-to-peer fashion. In each round, each node will update its local model by conducing several SGD updates on its local dataset, and then sends the obtained model to its neighbors (here, all other nodes). After the node receives the models from the neighbors, it will average them, and use this new model as the initialization to further update the local model of this round. The averaging of the models from

other nodes can in general be weighted; for instance, the original FedAvg method proposes using the size of the local training data as the weights.

2) Local learning: The model of each node is trained locally by performing several SGD update, based on its local dataset only. There is no model or data exchange with other nodes.

3) An hypothetical graph: Our last baseline considers an undirected graph selected a priori according to the similarity between the node's data. In particular, if two nodes have the same majority and minority classes in their local datasets, they will be connected to each other in this assumed graph. For instance, in our experiment, we consider a four node environment, divided into two groups; each group contains two nodes. The data of the two nodes in the same group have the same majority classes and minority classes, and so their datasets are IID; we assume these nodes should exchange information with each other according to this similarity. However, the data between the two groups are not similar, so no communication occurs between the two groups.

5.3 Results and Discussion

5.3.1 Is collaborative learning not helpful between dissimilar models?

Some earlier research on personalized learning (e.g. [21] and [1]) establish collaboration graphs based on the model's similarity between two nodes. When the local models of the two nodes are similar, there is a higher edge weight between the two nodes. Conversely, when the model similarity between two nodes is slight, there will

be a smaller edge weight. As the similarities between two models are the same (i.e., as the measure of similarity used is symmetric), an undirected collaboration graph will be formed in this way. But will the undirected graph that relies on model similarity like this be the best?

To investigate this question, we designed the following experiment. We selected three different types of clothing with certain similarities from the Fashion-MNIST data set: Sandal, Sneaker, and Ankle boot. In this experiment, we have two nodes, and $p = 0.7$ in the two nodes. The majority class on node 0 is Sandal and Sneaker, and the majority class on node 1 is the Ankle boot. From the experimental results shown in Figure 1, we can see that the models between the two nodes are not similar, but the mutual communication of the models between the two nodes is beneficial. This outcome may be related to the similarity of the majority classes tasks processed by the two nodes.

Another experiment shown in Figure 2 ($p = 0.7$, the majority classes of node 0 is T-shirt/top and Trouser, and the majority class of node 1 is Pullover) showed similar results. Therefore, we show that it may not be sufficient to consider only the model similarity of two nodes to determine whether they should communicate.

5.3.2 Is an underacted graph or a directed graph better?

We next designed the following experiment to verify whether a pair of nodes have similar (symmetric) effects on each other. We used two nodes, and $p = 0.7$. The majority classes of node 0 include T-shirt/top, Trousers, and Pullover. The majority classes of node 1 include Sandal, Shirt, and Bag. From the experimental results in Figure 3, we can see that node 0 can benefit from the communication with node

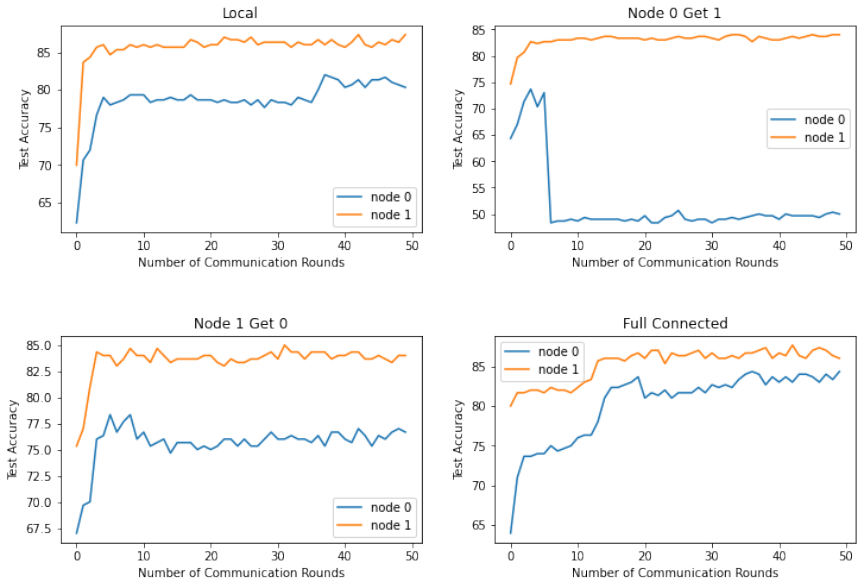

Figure 5.1: Even though the data of the two nodes is not IID, they benefit from collaboration due to similarity of the majority class between them.

1, and node 0 will be negatively influenced due to the communication with node 1. Also, based on such experimental results, we propose to use directed graphs instead of undirected graphs.

5.3.3 The influence of IID and non-IID data

To verify the influence of the distribution of the data, we designed the following experiment, with results shown in Table 5.1. The majority classes of Node 0 are T-shirt/top and trousers, and The majority class of Node 1 trousers. We have selected three different values of p, which are 0.7, 0.6, and 0.45. As can be seen from the table below, a directed graph with a height of $p = 0.7$ has better results, but when the data set is gradually made more IID, the fully distributed FedAvg has better results. This performance also matches with the worsening performance of FedAvg

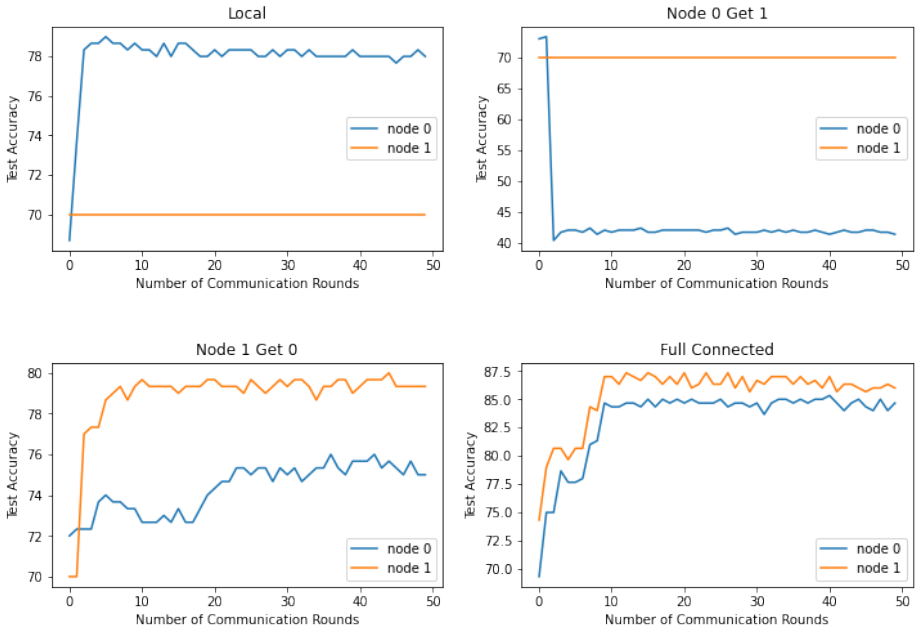

Figure 5.2: Even though the data of the two nodes is not IID, they benefit from collaboration, and given only partial similarity in their majority classes.

under high non-IID in previous research (e.g. [20]). Therefore, using a directed graph to replace the fully distributed FedAvg under a highly non-IID setting will improve results from fully distributed FedAvg.

5.3.4 Experiment on a four-node network

We also conducted experiments on a four node network. We compared the learning following our model, with a directed collaboration graph learned by our proposed algorithm as described in Section 3.2.1, and the models trained on this directed graph as described in Section 3.2.2. We compared the outcomes of learning against the three baselines described in section 5.2. In this experiment, we used a non-IID local dataset setting, where $p = 0.7$. The majority classes of Node 0 and Node 1 are T-shirt/top

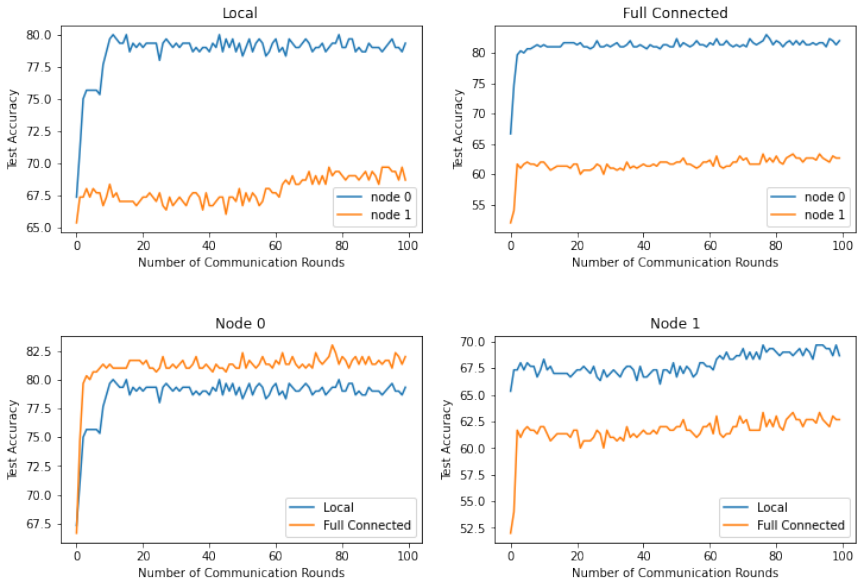

Figure 5.3: Collaboration benefits the local learning of node 0, but does not benefit the local learning of node 1. This shows that personalized learning should be done over *directed* communication graphs.

and Trouser, the majority class of Node 2 and Node 3 is Pullover. The experimental results are shown in Figure 4.

The experimental results show that the learned collaboration graph is better than local learning, undirected fully connected graph, and the hypothetical graph. In particular, comparing the hypothetical graph and the undirected fully connected graph, we can see that Node 0 and Node 1 only get a slight gain by exchanging information with Node 2 and Node 3. At the same time, Node 2 and Node 3 obtained a more significant improvement through the exchange. This may be because the local accuracy of Node 2 and Node 3 is low, while the local accuracy of Node 0 and Node 1 is high, so Node 2 and Node 3 are more dependent on the models from their neighbors.

	Local		Node 0 gets model from Node 1		Node 1 gets model from Node 0		Fully connected	
p	Node 0	Node 1	Node 0	Node 1	Node 0	Node 1	Node 0	Node 1
0.7	81.67	78.33	43.00	78.00	85.33	84.67	79.67	81.67
0.6	79.00	76.33	79.33	81.67	74	77.33	74.67	80.00
0.45	85.33	66.67	78.67	61.33	76.00	79.67	80.00	82.33

Table 5.1: Effects of IID vs non-IID local datasets on the benefits of personalized vs. Federated learning.

Therefore, when our collaboration graph gives the Node 0 and Node 1 models higher weights, the training leads to a higher accuracy rate.

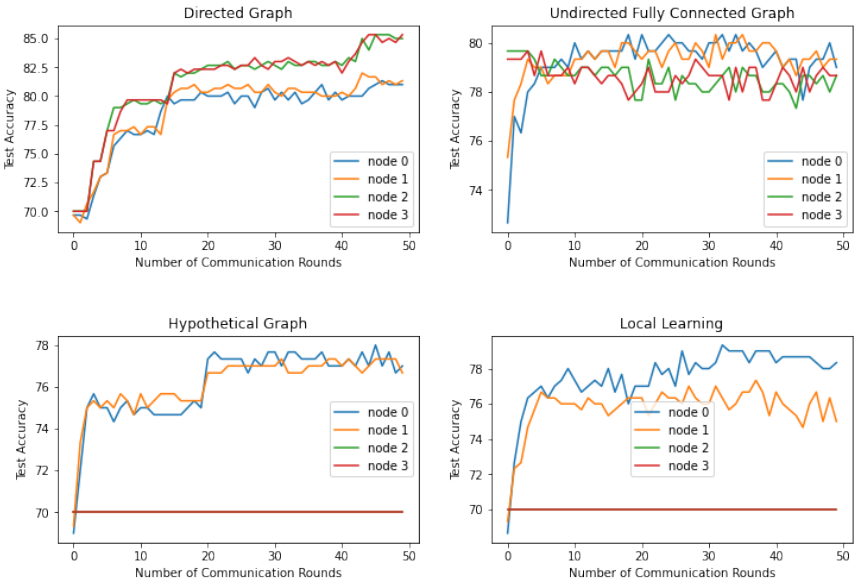

Figure 5.4: Test accuracy for FEDAVG on collaboration graph learned by our proposed algorithm, compared with that under an undirected fully connected graph, a hypothetical collaboration graph based on assumed datasimilarities, and local learning. The local datasets of agents are non-IID, with $p = 0.7$, and the majority classes of Node 0 and Node 1 begin T-shirt/top and Trouser, the majority class of Node 2 and Node 3 being Pullover. We observed that the learning over the directed collaboration graph identified by our proposed algorithm outperforms the other three baselines.

Chapter 6: Conclusion and Future Work

6.1 Conclusion

To solve the problem of learning a personalized model when the agent data is heterogeneous, we propose a fully decentralized federated learning method to make the local model more personalized by mixing the local model with the models from only "similar" neighboring agents. We found that in the case of highly non-IID data, this method can maintain the personalization of the local model and make the local model obtain higher performance than fully cooperative and consensus learning approaches like Federated Learning. Our experiments show that under high non-IID setting, some nodes will be disconnected from other nodes, thus avoiding negative interference from nodes with different data/tasks. We also observed that when the data is IID, this algorithm will approach the performance of a fully decentralized, fully-linked version of Federated Learning.

In addition, we see that nodes with a high degree of similarity form smaller *clusters* or communities in our learned collaboration graph, and information is exchanged between all nodes within a cluster. These clusters can be seen as scattered federated

learning. In fact, one could follow centrally coordinated or consensus learning algorithms developed in previous works within these clusters, without losing personalization. We again emphasize that the cluster-global model obtained by the nodes inside the cluster will have higher personalized accuracy than a global model that would have been obtained by mixing all nodes' models (e.g. as done in FL), because the dataset in a cluster is more IID while that of other clusters may in general not be sufficiently similar. Our algorithm can still identify similar clusters, and allow the exchange of local models between clusters to aid better personalized learning when beneficial.

Our other discovery is that the benefits of communication between the two nodes may be in general different and asymmetric. Nodes with large amounts of data and higher accuracy of their local models will rely more on their own local models; on the other side, nodes with lower data accuracy and smaller datasets tend to rely more on external information. Other research ([21] and [1]) on personalized learning under completely decentralized conditions are also based on this opinion. However, these works did not take into account the accuracy of external models, as done in our algorithm, to update the learning graph. Compared with our algorithm, this algorithm can obtain good weights through less computational consumption, and convergence has been proven.

6.2 Future Work

6.2.1 Mixing weight

Our current algorithm is based on trying and changing the weight, so we need to count the proportion of different results in several attempts to determine whether to

increase or decrease the weight. This method requires a lot of computing power to get the result. In future work, we hope to find an effective algorithm to obtain the weights of the mixed model. For example, as mentioned in [5], the edge weight can be updated by applying gradient descent in each round. Compared with our algorithm, this algorithm may obtain good weights while using less computational consumption.

6.2.2 Analytical study of convergence and bounds on speed of convergence

In addition, we only proved the existence of bound under a fixed graph in the thesis but did not prove the convergence and convergence rate of learning during the changing of weights, and we also did not prove the graph's convergence. We plan to try to prove these three parts in future research. Among them, I am most interested in the convergence rate under a fixed graph. In the next step, I will try to prove whether training on our graph will converge faster than the fully-linked graph.

6.2.3 Extended experiment settings

In the experiment part, due to the limitation of computer computing power, our main experiments are done under two or four nodes settings. We plan to try to apply this algorithm to more nodes in future research. We will try to study whether our algorithm will still be effective and the efficiency under more nodes.

Bibliography

[1] Inês Almeida and Joao Xavier. Djam: Distributed jacobi asynchronous method for learning personal models. *IEEE Signal Processing Letters*, 25(9):1389–1392, 2018.

[2] Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Mike Rabbat. Stochastic gradient push for distributed deep learning. In *International Conference on Machine Learning*, pages 344–353. PMLR, 2019.

[3] Monik Raj Behera, Sudhir Upadhyay, Suresh Shetty, and R. den Otter. Federated learning using peer-to-peer network for decentralized orchestration of model weights. 2021.

[4] Fei Chen, Mi Luo, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. Federated meta-learning with fast convergence and efficient communication. *arXiv preprint arXiv:1802.07876*, 2018.

[5] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

[6] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[8] Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*, 2020.

[9] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[10] Mert Kayaalp, Stefan Vlaski, and Ali H Sayed. Dif-maml: Decentralized multi-agent meta-learning. *arXiv preprint arXiv:2010.02870*, 2020.

[11] Jiyi Li, Tomohiro Arai, Yukino Baba, Hisashi Kashima, and Shotaro Miwa. Distributed multi-task learning for sensor network. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 657–672. Springer, 2017.

[12] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *arXiv preprint arXiv:1705.09056*, 2017.

[13] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *NIPS*, 2017.

[14] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[15] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. In *International Conference on Machine Learning*, pages 4615–4625. PMLR, 2019.

[16] Angelia Nedić and Alex Olshevsky. Stochastic gradient-push for strongly convex functions on time-varying directed graphs. *IEEE Transactions on Automatic Control*, 61(12):3936–3947, 2016.

[17] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. *arXiv preprint arXiv:1705.10467*, 2017.

[18] Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. Decentralized collaborative learning of personalized models over networks. In *Artificial Intelligence and Statistics*, pages 509–517. PMLR, 2017.

[19] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[20] Hao Yu, Rong Jin, and Sen Yang. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. In *International Conference on Machine Learning*, pages 7184–7193. PMLR, 2019.

[21] Valentina Zantedeschi, Aurélien Bellet, and Marc Tommasi. Fully decentralized joint learning of personalized models and collaboration graphs. In *International Conference on Artificial Intelligence and Statistics*, pages 864–874. PMLR, 2020.

[22] Edvin Listo Zec, Olof Mogren, John Martinsson, Leon René Sütfeld, and Daniel Gillblad. Federated learning using a mixture of experts. *arXiv preprint arXiv:2010.02056*, 2020.

[23] Xi Sheryl Zhang, Fengyi Tang, Hiroko H Dodge, Jiayu Zhou, and Fei Wang. Metapred: Meta-learning for clinical risk prediction with limited patient electronic health records. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2487–2495, 2019.