CHAPTER

Graph-aware federated learning

10

Songtao Lu[a], **Pengwei Xing**[b], **and Han Yu**[b]
[a]*IBM Thomas J. Watson Research Center, Yorktown Heights, NY, United States*
[b]*Nanyang Technological University, Singapore, Singapore*

10.1 Introduction

Even the existing federated learning (FL) framework has been shown successfully striking the balance between the communication and computation complexities, however, the server-to-slaver type of learning structure still forgoes the topology feature and/or information of distributed sampled data for further increasing the communication efficiency and/or the generalization performance. In this chapter, we investigate more advanced graph-aware federated learning (GFL) models and algorithms, and then present the theoretical justifications and numerical performance. This chapter mainly includes the key FL techniques with emphasis on different goals of building GFL systems.

Decentralized federated learning (DFL) is one of the most straightforward ways that can further improve both the computing and communication efficiency of the classic FL system, where there are n devices connected over a graph and which jointly learn a common interest machine learning model. In contrast to the classic FedAvg [18], DFL is more robust to failure of message passing among the nodes and staleness of transmitting data packages. Also, DFL has the nature of having low communication overheads and keeping the data confidential during the data transmission stage, as there is no central server that requests the model parameters from all the devices. For example, in a health medical system, the patients' data and information are private, so they are not shareable over either hospitals or doctors. However, note that there is still a need that a machine learning model can extract the critical latent space structures through a sufficiently large dataset to remove the outliers. It has been shown that DFL is one of the most promising strategies to improve communication efficiency and model performance through aggregating electronic health records over multiple data resources [17].

In practice, the global model might not be unique in the sense that there could be multiple global models, each possibly containing different features of data due to the heterogeneity of data distributions. Selecting the correct memberships for the local models to each global model is not trivial, which would result in a combinatorial problem. One of the most straightforward ways is to try different global models for fitting the local data and select the best one at each iteration, which is called an

Federated Learning. https://doi.org/10.1016/B978-0-44-319037-7.00018-1

iterative federated clustering algorithm and proposed in [6]. However, this way is time-consuming as it needs all the global models to traverse all the local data sets. As the data distributions possess a cluster structure, a K-means type of algorithm is appropriate for clustering the local models based on the similarity among the nodes. In [24], federated stochastic expectation maximization (FeSEM) is proposed based on measuring the distance between the two model parameters in which one is the local model parameter and the other is the weighted mean of the parameters of the neighboring models selected by the EM algorithm. It turns out that FeSEM can learn the memberships of the local model and optimize the local loss values in an alternative way, and achieve state-of-the-art results.

Inspired by the hierarchical learning strategy, a bilevel optimization based FL model was recently proposed, which considers the membership selection (or more general feature representation) and local data adaptation as two levels of an optimization problem. In such a way, the upper level (UL) model can integrate domain knowledge into the FL model, while the lower level (LL) model can use this information for adapting the local data personally. For example, the graph knowledge about the similarity among the nodes would be proportional to the heterogeneity of data features, which can be used for enhancing the generalization performance of FL models. Despite a line of graph learning-related works that mostly focus on aggregating graph neural networks (GNNs) [10], the graph information in these studies only appears in the respective local models rather than incorporating any global graph structure information. To the best of our knowledge, the scenario in which FL clients are organized into graph embedding is rarely investigated. Under this setting, we will introduce the bilevel optimization enhanced graph-aided federated learning, which applies the graph embedding techniques to capture inherent information over the topology of FL clients.

In this chapter, we will address the heterogeneity issue of the FL systems, and bring in the GFL models and corresponding algorithms, as well as their theoretical and numerical justifications.

10.2 Decentralized federated learning

When clients are connected over a graph, the communication only happens between two neighboring nodes. In this setting, we consider a communication graph denoted by $\mathcal{G} \triangleq \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ denotes the set of the vertices in this graph and $\mathcal{E}$ stands for the edges. We use n to represent the total number of nodes, i.e., $|\mathcal{V}|$, in this graph, and subsequently $i \in [n]$ as the index of each node. The optimization problem of the distributed learning system can be mathematically formulated as follows:

$$(\mathbb{P}1) \quad \min_{\boldsymbol{\theta}_i \in \mathbb{R}^d, \forall i} \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\xi_i \sim \mathcal{D}_i} F_i(\boldsymbol{\theta}_i; \xi_i) \text{ such that } \boldsymbol{\theta}_i = \boldsymbol{\theta}_j, \quad j \in \mathcal{N}_i, \forall i, \tag{10.1}$$

where $\mathcal{N}_i$ represents the set of node i neighbors, $F_i(\boldsymbol{\theta}_i; \xi_i)$ denotes the (possible non-convex) loss function at the ith node, ξ_i is the data sample collected at node i and following a certain distribution $\mathcal{D}_i$. The goal of this model in Eq. (10.1) is to learn a global model $\boldsymbol{\theta} \in \mathbb{R}^d$ such that the sum of total loss functions $\{F_i(\boldsymbol{\theta}; \xi_i), \forall i\}$ is minimized based on the datasets $\{\mathcal{D}_i, \forall i\}$ sampled over graph $\mathcal{G}$.

There are many existing algorithms that can solve this problem in a distributed way, including distributed stochastic gradient descent [2], primal–dual algorithms [12,13], (variance-reduced) gradient tracking [16,20], etc. The main idea is to update each local optimization variable by aggregating its neighbor models' parameters, followed by one step of stochastic gradient descent type of update. The issue here is that each round of the model aggregation would incur a large number of parameters sent over the network, which might not be feasible or incur heavy delays in the model update. Therefore, communication-efficient transmission schemes are motivated and proposed for reducing the overload of passing model parameters per round [11].

To present the DFL algorithms, we first define the mixing matrix $\mathbf{W}$ which represents the connectivity of the nodes as follows: 1) $w_{ij} > 0$ if $(i, j) \in \mathcal{E}$ and 0 otherwise; 2) $\mathbf{W}$ is doubly stochastic, i.e., $\mathbb{1}^T\mathbf{W} = \mathbb{1}^T$ and $\mathbf{W}\mathbb{1} = \mathbb{1}$, where w_{ij} denotes the (i, j)th entry of matrix $\mathbf{W}$ and $\mathbb{1}$ stands for the all one vector. Typical rules that satisfy these two properties include the Laplacian, Metropolis–Hasting, and maximum-degree weights [23]. Let the gradient estimate of the loss function at point $\boldsymbol{\theta}$ be defined by

$$\widehat{\nabla} f_i(\boldsymbol{\theta}) \triangleq m^{-1} \sum_{k=1}^{m} \nabla F_i(\boldsymbol{\theta}; \xi_{ik}), \tag{10.2}$$

where ξ_{ik} denotes the kth data sample at node i and m is the mini-batch size. Then, the local update of the model parameter is

$$\boldsymbol{\theta}_i^{t+1} = \boldsymbol{\theta}_i^t - \alpha^t \widehat{\nabla} f_i(\boldsymbol{\theta}_i^t), \tag{10.3}$$

where t denotes the index of iteration and α^t is the step size. Similar as the classic FL scheme, for every τ iterations, we additionally perform one round of communication based on the local update, i.e.,

$$\boldsymbol{\theta}_i^{t+1} = \sum_{j \in \mathcal{N}_i} \mathbf{W}_{ij} \boldsymbol{\theta}_j^t - \alpha^t \widehat{\nabla} f_i(\boldsymbol{\theta}_i^t) \quad \text{if } \mathrm{mod}(t, \tau) = 0, \tag{10.4}$$

where the step $\sum_{j \in \mathcal{N}_i} \mathbf{W}_{ij} \boldsymbol{\theta}_j^t$ refers to the model aggregation and $1/\tau$ refers to the communication frequency. The model update by Eq. (10.3) followed by Eq. (10.4) per every τ steps is a straightforward extension from classic distributed stochastic gradient descent (DSGD) to SGD-based DFL. Here, we call this algorithm DFL-SGD. It is not hard to see that DFL-SGD tries to balance the trade-off between the iterates' convergence and communication efficiency. Intuitively, the local update optimizes the individual model without sharing the data samples, and the model aggression step

enforces the consensus among the distributed model parameters so that the learned model can leverage the networked data to reduce the variance of the local stochastic gradient estimate.

The DFL-SGD algorithm performs well when the network is homogeneous. When the heterogeneity of data distributions $\{\mathcal{D}_i, \forall i\}$ increases, it is well known that performance of distributed SGD (DSGD) algorithm decreases as the discrepancy term of the data distributions will show up at the denominator of the convergence rate of DSGD [21]. As DFL-SGD inherits from DSGD, this issue remains. A more advanced technique, stochastic gradient tracking, is proposed to approximate the global gradient estimate locally and improve the numerical performance of DSGD deployed in the heterogeneous networks [16]. When the communication efficiency is taken into account, the local updates of the model are as follows:

$$\boldsymbol{\theta}_i^{t+1} = \boldsymbol{\theta}_i^t - \alpha^t \boldsymbol{\vartheta}_i^t, \tag{10.5a}$$

$$\boldsymbol{\vartheta}_i^{t+1} = \boldsymbol{\vartheta}_i^t + \widehat{\nabla} f_i(\boldsymbol{\theta}_i^{t+1}) - \widehat{\nabla} f_i(\boldsymbol{\theta}_i^t), \tag{10.5b}$$

where $\boldsymbol{\vartheta}_i^t$ is called gradient tracker and initialized as vector $\mathbf{0}$. It can be seen that the gradient tracking based local update Eq. (10.5a) is analogous to Eq. (10.3) with the difference being only the gradient estimate. In Eq. (10.3), the local stochastic gradient estimate is directly used for the model update, while in Eq. (10.5a) an auxiliary variable is adopted instead. From the theory perspective, $\boldsymbol{\vartheta}_i^t$ can keep tracking the network total stochastic gradient $n^{-1}\sum_{i=1}^n \widehat{\nabla} f_i(\boldsymbol{\theta}_i^t)$ through the update rule Eq. (10.5b). Similar to DFL-SGD, gradient tracking based DFL (DFL-GT) also conducts the model aggregation after every τ local update steps as follows:

$$\boldsymbol{\theta}_i^{t+1} = \sum_{j\in\mathcal{N}_i} \mathbf{W}_{ij}\boldsymbol{\theta}_j^t - \alpha^t \boldsymbol{\vartheta}_i^t \quad \text{if } \mathrm{mod}(t,\tau) = 0, \tag{10.6a}$$

$$\boldsymbol{\vartheta}_i^{t+1} = \sum_{j\in\mathcal{N}_i} \mathbf{W}_{ij}\boldsymbol{\vartheta}_j^t + \widehat{\nabla} f_i(\boldsymbol{\theta}_i^{t+1}) - \widehat{\nabla} f_i(\boldsymbol{\theta}_i^t) \quad \text{if } \mathrm{mod}(t,\tau) = 0. \tag{10.6b}$$

This DFL algorithm has been studied in [17] for decentralized learning of electronic health records. Under the mild assumption on the gradient Lipschitz continuity of the loss function, it has been established in [14] that DFL-GT algorithm can find the first-order stationary points (FOSPs) with a convergence rate of $\mathcal{O}(1/\sqrt{nT})$ in the sense that both the first-order stationarity in terms of the gradient size of the loss function in the consensus space, i.e., $\mathbb{E}\|n^{-1}\sum_{i=1}^n \nabla f_i(\bar{\boldsymbol{\theta}}^t)\|^2$, and the consensus violation, i.e., $\mathbb{E}[n^{-1}\sum_{i=1}^n \|\boldsymbol{\theta}_i^t - \bar{\boldsymbol{\theta}}^t\|^2]$, are shrinking with the speed of $\mathcal{O}(1/\sqrt{nT})$, where $f_i(\boldsymbol{\theta}_i) \triangleq \mathbb{E}_{\xi_i\sim\mathcal{D}_i} F_i(\boldsymbol{\theta}_i;\xi_i)$, $\bar{\boldsymbol{\theta}} = n^{-1}\sum_{i=1}^n \boldsymbol{\theta}_i$, and the expectation is taken over both the randomness of data samples and the index of the iterations. Moreover, it has been also proven in [14] that τ can be chosen as large as $\mathcal{O}(T^{1/4})$ (i.e., the communication efficiency of DFL-GT is $\mathcal{O}(T^{3/4})$ rather than $\mathcal{O}(T)$ required in the classic DSGD), and this class of algorithms is also extendable to solve problems in time-varying directed graphs.

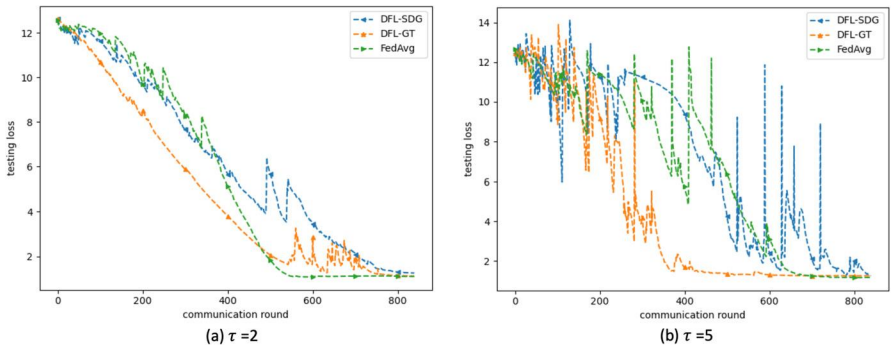

FIGURE 10.1

Loss value v.s. communication rounds on a synthetic dataset (testing loss).

The general DFL framework with algorithms DFL-SGD and DFL-GT is summarized in Algorithm 1.

Algorithm 1 Decentralized federated learning framework (DFL).

Initialize step size sequence α^t
for $t = 0, 1, 2, \ldots, T$ **do**
    Randomly sample m data points locally
    Estimate gradient of the local loss function by Eq. (10.2)
    Perform local model update by Eq. (10.3) or Eq. (10.5a) ▷ at each node in parallel
    if t is a multiple of τ, i.e., $\mod(t, \tau) = 0$ **then**
        Update model parameters by Eq. (10.4) or Eq. (10.6a) and Eq. (10.6b)
    end if
end for

We set up a simple topology of five nodes, using the same groundtruth weights, which are then multiplied by inputs belonging to various intervals of uniform distribution (including $[-30, 30]$, $[40, 100]$, $[-50, -55]$, $[-1, 1]$, $[-0.1, 0.1]$) with different five nonlinear functions (tanh, sigmoid, ReLu, sine, and cosine) to generate different heterogeneous data.

The numerical results of comparing FedAvg, DFL-SGD, and DFL-GT are shown in Fig. 10.1, where we use a three-layer nonlinear multilayer perceptron to regress these data, and use the same step size scheduling rule (i.e., $15/\sqrt{t}$) for DFL-SGD and DFL-GT. It can be seen that FedAvg converges the fastest because it aggregates the model completely for each round. DFL-GT converges faster than DFL-SGD as the gradient tracker over the graph can approximate the full gradient as the iterates proceed.

10.3 Multi-center federated learning

Besides the advanced algorithms designed for heterogeneous FL networks, cluster-structured models are also considered for tackling the issues of non-i.i.d. data distributions over distributed networks. The underlying assumption is that there exist K global models and the scattered local models only belong to one of these global ones. In FeSEM [24], a multi-center model aggregation model is used for FL, where all the local models are partitioned into K clusters, denoted as $\{\mathcal{C}_k, k \in [K]\}$. Aside from minimizing the consensus errors and regression loss function, an assignment of each model to its nearest cluster is also included in the learning process. Specifically, the overall problem is formulated as follows:

$$(\mathbb{P}2) \min_{\boldsymbol{\theta}_i, \bar{\boldsymbol{\theta}}_k, r_{ik}, \forall i,k} \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\xi \sim \mathcal{D}_i} F_i(\boldsymbol{\theta}_i; \xi) + \frac{\lambda}{n} \sum_{k=1}^{K} \sum_{i=1}^{n} r_{ik} \text{dist}(\boldsymbol{\theta}_i, \bar{\boldsymbol{\theta}}_k), \tag{10.7}$$

where λ controls the trade-off between the supervised loss and multi-center based model discrepancy, $\text{dist}(\cdot, \cdot)$ denotes the distance between two model parameter spaces, $\bar{\boldsymbol{\theta}}_k$ represents the kth center of cluster $\mathcal{C}_k$, and r_{ik} is the binary assignment variable that indicate whether the ith model is classified to the kth cluster or not. The regularization term measures the distance between each local model to its nearest global model.

For the multi-center FL model, the expectation-maximization (EM) method is one of the most standard techniques for solving clustering related problems. Combining with a local supervised learning update, a stochastic block coordinate descent type of algorithm is proposed for the FL setting, called federated stochastic EM (FeSEM), in [24]. The main idea of FeSEM is adapting the EM algorithm in searching for the multiple centers $\bar{\boldsymbol{\theta}}_k$ while fixing local model parameters $\{\boldsymbol{\theta}_i, \forall i\}$, and then performing local updates of $\boldsymbol{\theta}_i$ for τ steps to minimize the local loss function values while keeping the assignment of every model to each cluster learned from the previous step unchanged. More detailed procedures of FeSEM are shown in Algorithm 2 and explained as follows:

E-Step. The first step of each model is to look for the nearest clusters based on the distance between the local model parameter $\boldsymbol{\theta}_i^t$ and the existing centers $\{\bar{\boldsymbol{\theta}}_j^t, j \in [K]\}$ and obtain the assignment variable by

$$r_{ik}^t = \begin{cases} 1, & \text{if } k = \arg\min_j \text{dist}(\boldsymbol{\theta}_i^t, \bar{\boldsymbol{\theta}}_j^t), \\ 0, & \text{otherwise.} \end{cases} \tag{10.8}$$

M-Step. Then, the center of each cluster is calculated by

$$\bar{\boldsymbol{\theta}}_k^t = \frac{1}{\sum_{i=1}^{n} r_{ik}} \sum_{i=1}^{n} r_{ik}^t \boldsymbol{\theta}_i^t \tag{10.9}$$

while fixing model parameters and current center assignments.

Algorithm 2 FeSEM – Federated Stochastic EM [24].

Initialize K, $\{\boldsymbol{\theta}_i^0\}$, $\{\bar{\boldsymbol{\theta}}_k^t\}$
for $t = 0, 1, 2, \ldots, T$ **do**
    E-Step
    Calculate distance $d_{ik}^t = \text{dist}(\boldsymbol{\theta}_i^t, \bar{\boldsymbol{\theta}}_k^t), \forall i, k$
    Update r_{ik}^t using d_{ik}^t by Eq. (10.8)
    M-Step
    Group devices to $\mathcal{C}_k^t$ using r_{ik}^t
    Update $\bar{\boldsymbol{\theta}}_k^t$ using r_{ik}^t and $\boldsymbol{\theta}_i^t$ by Eq. (10.9) ▷ Compute the new center
    for each cluster $k = 1, \ldots, K$ **do**
        for $i \in \mathcal{C}_k^t$ **do**
            Send $\bar{\boldsymbol{\theta}}_k^t$ to device i
            $\theta_i^{t+1} \leftarrow$ **Local_Update**$(i, \bar{\boldsymbol{\theta}}_k^t)$
        end for
    end for
end for

Local models. Given the multi-centers, each local device will fine-tune the model for individual data distribution by minimizing the regression loss function and the dissimilarity between the local and assigned global model parameters, which means that each $\boldsymbol{\theta}_i$ needs solving the following problem:

$$\min_{\theta_i} \mathbb{E}_{\xi \sim \mathcal{D}_i} F_i(\boldsymbol{\theta}_i; \xi) + \lambda \sum_{k=1}^{K} r_{ik} \text{dist}(\boldsymbol{\theta}_i, \bar{\boldsymbol{\theta}}_k). \tag{10.10}$$

Here, Eq. (10.10) is an unconstrained optimization problem. The model parameters can be updated efficiently by multiple steps of SGD, which corresponds to the local SGD update in the standard FL scenario. Finally, iterating the above three steps results in the FeSEM algorithm for solving the multi-center FL problem.

Algorithm 3 Local_Update.

Input: i – device index, $\bar{\boldsymbol{\theta}}_k^t$
Output: $\boldsymbol{\theta}_i^{t+1}$ – updated local model
Initialize $\boldsymbol{\theta}_i^t$ by $\bar{\boldsymbol{\theta}}_k^t$
for τ local training steps **do**
    Update $\boldsymbol{\theta}_i$ by any efficient algorithm with data $\mathcal{D}_i$ ▷ needed for Eq. (10.10)
end for
Return $\boldsymbol{\theta}_i$

10.4 Graph-knowledge based federated learning

Even though the multi-center FL takes the cluster structure into consideration, FeSEM still assumes that there is no overlap among the global models, which might not be true or restrict for practical heterogeneous networks. Actually, FeSEM can be considered as a special case of the two levels of the optimization problem, where one level is minimizing the regularization term in Eq. (10.7) for searching the cluster structure and the other is minimizing the supervised loss for data feature extraction. Mathematically, the class of multi-task learning problems can be formulated as a bilevel optimization as follows [12]:

$$\min_{\boldsymbol{\varphi}} \ell(\boldsymbol{\varphi}) \triangleq f\left(\boldsymbol{\varphi}, \{\boldsymbol{\theta}_i^*(\boldsymbol{\varphi})\}\right) \tag{10.11a}$$

$$\text{such that } \boldsymbol{\theta}_i^*(\boldsymbol{\varphi}) \in \arg\min_{\boldsymbol{\theta}_i} \mathbb{E}_{\xi \in \mathcal{D}_i} G_i(\boldsymbol{\varphi}, \boldsymbol{\theta}_i; \xi),\ \forall i \in [n], \tag{10.11b}$$

where $\boldsymbol{\varphi}$ is the UL decision variable, $\{\boldsymbol{\theta}_i, \forall i\}$ denote the LL model parameters, $f(\cdot,\cdot)$ and $G_i(\cdot,\cdot)$ respectively represent the UL and LL loss functions, and $\{\boldsymbol{\theta}_i^*(\boldsymbol{\varphi}), \forall i \in [n]\}$ stand for the optimal solutions of the LL optimization variables. This model can cover a wide range of hierarchical FL learning problems, e.g., federated acoustic speech recognition [3], personalized meta-learning [4], multi-agent actor–critic schemes in reinforcement learning [15], etc.

Note that the structured graph knowledge is related to all the local models, so the LL optimization variables would be coupled. One example is the multi-center FL case as shown in Eq. (10.10), where the kth center involved in the LL problem is calculated based on all $\boldsymbol{\theta}_i$s. Therefore, a more generalized bilevel optimization enhanced GAFL, or BiG-FL in short, is proposed in [25], by formulating the LL optimization problem as a competitive game as follows:

$$(\mathbb{P}3) \quad \min_{\boldsymbol{\varphi}} \ell(\boldsymbol{\varphi}) = f\left(\boldsymbol{\varphi}, \{\boldsymbol{\theta}_i^*(\boldsymbol{\varphi})\}\right) \tag{10.12a}$$

$$\text{such that } \boldsymbol{\theta}_i^*(\boldsymbol{\varphi}) = \arg\min_{\boldsymbol{\theta}_i} \mathbb{E}_{\xi \in \mathcal{D}_i} G_i\left(\boldsymbol{\varphi}, \boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}^*(\boldsymbol{\varphi}); \xi\right),\ \forall i \in [n], \tag{10.12b}$$

where $\boldsymbol{\theta}_{-i}^*(\boldsymbol{\varphi})$ denotes $\{\boldsymbol{\theta}_{-i}^*(\boldsymbol{\varphi}) | \boldsymbol{\theta}_j^*(\boldsymbol{\varphi}), j \neq i, \forall j \in \mathcal{N}_i\}$. From this model, we can see that the UL optimization problem is targeted at minimizing the globally shareable parameter $\boldsymbol{\varphi}$ while the LL problem is used for integrating the UL knowledge and adapting the local data distributions. In the following, we provide one way of applying the BiG-FL to a GNN-embedded FL system.

10.4.1 Applications of BiG-FL

Inspired by graph embedding learning for link prediction [7], BiG-FL can take the connectivity of FL clients/devices (i.e., edge information in the global graph structure) as a guide in the UL optimization process and maps this topology information to weigh the similarity of neighboring clients' models.

Let $\boldsymbol{\theta}^*(\boldsymbol{\varphi}) \in \mathbb{R}^{n\times d}$ denote the concatenation of all the LL optimal solutions. Constructing the embedding matrix $\mathbf{H}$ via linear message passing in GNNs yields:

$$\mathbf{H} = \mathbf{L}\boldsymbol{\theta}^*(\boldsymbol{\varphi})\boldsymbol{\varphi}, \tag{10.13}$$

where $\mathbf{L} \in \mathbb{R}^{n\times n}$ is a Laplacian matrix (e.g., $\mathbf{L} \triangleq \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}$, where $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\widetilde{\mathbf{D}}_{ii} = \sum_j \widetilde{\mathbf{A}}_{ij}$, which are commonly used in graph convolution networks (GCNs) [8]), $\boldsymbol{\theta}^*(\boldsymbol{\varphi})$ is learned from local models, and $\boldsymbol{\varphi} \in \mathbb{R}^{d\times d}$ denotes the UL parameters acting as the weights in GNN with a shape of $d \times d$.

UL loss function. With this graph-based representation, we can formulate the UL objective function using the cosine embedding loss as

$$\begin{aligned} f(\varphi, \boldsymbol{\theta}^*(\boldsymbol{\varphi})) \triangleq & \frac{1}{n}\sum_{i=1}^{n}\sum_{j\in\mathcal{N}_i}\left(1-\cos(\mathbf{H}_i,\mathbf{H}_j)\right) \\ & + \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{j\sim P_i}\max\left(0,\cos(\mathbf{H}_i,\mathbf{H}_j)\right) + \frac{\lambda}{2}\left\|\boldsymbol{\varphi}-\mathbf{I}\right\|^2, \end{aligned} \tag{10.14}$$

where i and j are indices of the nodes (i.e., FL clients/devices),

$$\cos(\mathbf{H}_i,\mathbf{H}_j) \triangleq \frac{\mathbf{H}_i^T\mathbf{H}_j}{\|\mathbf{H}_i\|\|\mathbf{H}_j\|}, \tag{10.15}$$

and $\mathcal{P}_i$ denotes the negative sampling distribution at client i [7].

The UL objective function for BiG-Fed in Eq. (10.14) utilizes the cosine embedding loss to calculate both the client-pair embedding similarity of all linked clients and the client-pair embedding dissimilarity of negative samples. As the embedding is derived from the local model weights, Eq. (10.14) couples local learning tasks with the graphical link prediction task at the server side.

It is worth noting that the link prediction model learns the relative relationships among the local models. In addition, we expect that the optimized UL model can maintain the centroid of the overall weight distribution, so we initialize $\boldsymbol{\varphi}$ by the identity matrix $\mathbf{I}$, and add a regularity term on $\boldsymbol{\varphi}$ with the identity matrix $\mathbf{I}$ at the UL loss function.

LL loss function. The local LL learning tasks of BiG-FL at each device are finding the Nash equilibrium of the following problem:

$$\boldsymbol{\theta}_i^*(\boldsymbol{\varphi}) = \arg\min_{\boldsymbol{\theta}_i} g_i\left(\boldsymbol{\varphi}, \boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}^*(\boldsymbol{\varphi})\right), \forall i \in [n], \tag{10.16}$$

where $g_i(\boldsymbol{\varphi}, \boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}^*(\boldsymbol{\varphi})) \triangleq \mathbb{E}_{\xi\sim\mathcal{D}_i} G_i(\boldsymbol{\varphi}, \boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}^*(\boldsymbol{\varphi}); \xi)$. In this case, the UL weight $\boldsymbol{\varphi}$ is introduced into the LL learning task to penalize the distance between the aggregation of neighboring weights and the local one, as we assume that the neighboring clients share a similar latent space. Hence, for each client, the loss function $g_i(\cdot)$

includes both the supervised learning error and distance between the local model parameter and the centroid learned from the UL graph knowledge:

$$g_i\left(\boldsymbol{\varphi}, \boldsymbol{\theta}_i, \boldsymbol{\theta}^*_{-i}(\boldsymbol{\varphi})\right) \triangleq \frac{1}{2}\mathbb{E}_{(X_i,Y_i)\sim\mathcal{D}_i}\|Y_i - h_{\boldsymbol{\theta}_i}(X_i)\|^2 + \frac{\lambda}{2}R_i\left(\boldsymbol{\varphi}, \boldsymbol{\theta}_i, \boldsymbol{\theta}^*_{-i}(\boldsymbol{\varphi})\right) + \frac{\kappa}{2}\|\boldsymbol{\theta}_i\|^2, \forall i \in [n], \tag{10.17}$$

where $h_{\boldsymbol{\theta}_i}(\cdot) : X_i \to Y_i$ denotes the nonlinear mapping parametrized by a neural network with weight $\boldsymbol{\theta}_i$, X_i and Y_i respectively represent the data and labels owned by the ith client, $\mathcal{D}_i$ denotes the joint distribution of X_i and Y_i, regularization term $\kappa\|\boldsymbol{\theta}_i\|^2$ is used for stabilizing the LL learning process with $\kappa > 0$, and $R_i(\boldsymbol{\varphi}, \boldsymbol{\theta}_i, \boldsymbol{\theta}^*_{-i}(\boldsymbol{\varphi}))$ denotes the distance based regularization term,

$$R_i(\boldsymbol{\varphi}, \boldsymbol{\theta}_i, \boldsymbol{\theta}^*_{-i}(\boldsymbol{\varphi})) = \left\|(\mathbf{L}_{i,:} - \mathbf{L}_{i,i})\boldsymbol{\theta}^*(\boldsymbol{\varphi})\boldsymbol{\varphi} + \mathbf{L}_{i,i}\boldsymbol{\theta}_i\boldsymbol{\varphi} - \boldsymbol{\theta}_i\right\|^2. \tag{10.18}$$

Furthermore, $\mathbf{L}_{i,:}$ denotes the ith row of Laplacian matrix $\mathbf{L}$. Note that term $(\mathbf{L}_{i,:} - \mathbf{L}_{i,i})\boldsymbol{\theta}^*(\boldsymbol{\varphi})\boldsymbol{\varphi} + \mathbf{L}_{i,i}\boldsymbol{\theta}_i\boldsymbol{\varphi}$ is the weight embedding, where $(\mathbf{L}_{i,:} - \mathbf{L}_{i,i})\boldsymbol{\theta}^*(\boldsymbol{\varphi})\boldsymbol{\varphi}$ does not involve current node's weight $\boldsymbol{\theta}_i$ and can be approximated by the FL server in advance.

The penalization on the difference between the current weight and the weight embedding in each LL task, together with the fact that the UL variable learns the relationship of each weight embedding simultaneously via $\boldsymbol{\varphi}$, allows us to leverage the graph information to improve the generalization of the model by integrating the prior knowledge of similarity among the nodes. Next, we will introduce an algorithm for solving Eq. (10.12).

10.4.2 Algorithm design for BiG-FL

Solving Eq. (10.17) exactly to get the optimal solution is not practical. Instead, following the FL algorithms, we apply SGD for several steps and obtain an approximate solution of the LL problem. To be more specific, the LL optimization variable is updated by

$$\boldsymbol{\theta}_i^{k+1} = \boldsymbol{\theta}_i^k - \frac{\beta^t}{m}\sum_{j=1}^{m}\nabla_{\boldsymbol{\theta}_i}G_i(\boldsymbol{\varphi}^t, \boldsymbol{\theta}_i^k, \boldsymbol{\theta}_{-i}^k; \xi_j), \quad \forall i \in [n], \tag{10.19}$$

where k denotes the index of the inner loop by initializing $\boldsymbol{\theta}_i^0 = \boldsymbol{\theta}_i^t$, β^t is the LL step size, and $\nabla G_i(\boldsymbol{\varphi}^t, \boldsymbol{\theta}_i^k, \boldsymbol{\theta}_{-i}^k; \xi_j)$ represents the gradient of the ith BiG-FL LL empirical loss function evaluated at the point $(\boldsymbol{\varphi}^t, \boldsymbol{\theta}_i^k, \boldsymbol{\theta}_{-i}^k)$ with the jth data sample (i.e., $\xi_j \triangleq (X_j, Y_j) \sim \mathcal{D}_j$). After consecutive τ steps of the inner loop update, we set $\boldsymbol{\theta}_i^{t+1} = \boldsymbol{\theta}_i^\tau$.

Algorithm 4 Bilevel optimization enhanced GFL (BiG-FL).

Initialize the UL and LL step sizes α^0, β^0, and model parameters $\boldsymbol{\varphi}^0$, $\{\boldsymbol{\theta}_i^0\}$
for round $t = 0, 1, 2, \ldots, T$ **do**
  for node $i = 0, 1, 2, \ldots, n$ **do**
    Receive $(\mathbf{L}_{i,:} - \mathbf{L}_{i,i})\boldsymbol{\theta}^t$, $\boldsymbol{\varphi}^t$, and $\mathbf{L}_{i,i}$ from FL Server
    Calculate the gradient of term $R_i(\boldsymbol{\varphi}^t, \boldsymbol{\theta}_i^t, \boldsymbol{\theta}_{-i}^*(\boldsymbol{\varphi}^t))$ by Eq. (10.18)
    Update $\boldsymbol{\theta}_i^{t+1}$ by Eq. (10.19) ▷ needed for Eq. (10.17)
    Send $\boldsymbol{\theta}_i^{t+1}$ to the FL Server
  end for
  Receive $\{\boldsymbol{\theta}_i^{t+1}\}$ from clients
  Complete message passing with Eq. (10.13) to generate $\mathbf{H}$
  Update $\boldsymbol{\varphi}^{t+1}$ by Eq. (10.20) ▷ needed for Eq. (10.14)
  Distribute $\boldsymbol{\varphi}^{t+1}$, $(\mathbf{L}_{i,:} - \mathbf{L}_{i,i})\boldsymbol{\theta}^{t+1}$, and $\mathbf{L}_{i,i}$ to each node
end for

Similarly, the LL variable is updated by SGD with computing the hyper-gradient [5] as follows:

$$\boldsymbol{\varphi}^{t+1} = \boldsymbol{\varphi}^t - \frac{\alpha^t}{m}\sum_{j=1}^{m}\left(\nabla_{\boldsymbol{\varphi}} F_j^t - \nabla^2_{\boldsymbol{\varphi}\boldsymbol{\theta}} \boldsymbol{G}_j^t \left(\nabla^2_{\boldsymbol{\theta}\boldsymbol{\theta}} \boldsymbol{G}_j^t\right)^{-1} \nabla_{\boldsymbol{\theta}} F_j^t\right), \tag{10.20}$$

where α^t denotes the UL step size, $\nabla_{\varphi} F_j^t$ (the abbreviation of $\nabla_{\boldsymbol{\varphi}} F(\boldsymbol{\varphi}^t, \{\boldsymbol{\theta}_i^t\}; \xi_j)$) is the gradient of the UL empirical loss function evaluated at point $(\boldsymbol{\varphi}^t, \{\boldsymbol{\theta}_i^t\})$ with the jth tail node of the negative sample for head node i (i.e., $j \sim \mathcal{P}_i$), $\nabla^2_{\boldsymbol{\varphi}\boldsymbol{\theta}} \boldsymbol{G}_j^t$ and $(\nabla^2_{\boldsymbol{\theta}\boldsymbol{\theta}} \boldsymbol{G}_j^t)^{-1}$ respectively stand for the stochastic approximation of the Jacobian matrix of the LL loss function with respect to both $\boldsymbol{\varphi}$ and $\{\boldsymbol{\theta}_i\}$ and the inverse of Hessian matrix of the LL loss function. Here, we assume that the LL loss functions are strongly convex (e.g., if κ is sufficiently large), implying that $\nabla^2_{\boldsymbol{\theta}\boldsymbol{\theta}} \boldsymbol{G}_j^t$ is invertible. The complete implementation of BiG-FL is shown in Algorithm 4, where $\boldsymbol{\theta}$ is the concatenation of $\{\boldsymbol{\theta}_i, \forall i\}$.

The convergence guarantees of bilevel optimization algorithms have been investigated in recent works [5,12]. For the BiG-FL problem, it has been shown in [25] that when the UL and LL step sizes α^t and β^t are chosen on the order of $1/\sqrt{T}$, under the Lipschitz continuity of the UL loss function, as well as its Jacobian and Hessian, plus the strong convexity of the LL loss functions, Algorithm 4 needs $\mathcal{O}(1/\epsilon^4)$ iterations to reach the FOSPs of Eq. (10.12) in the sense that both the first-order stationarity of the UL optimization variable $\boldsymbol{\varphi}$, i.e., $\mathbb{E}\|\nabla \ell(\boldsymbol{\varphi}^t)\|^2$, and the distance between $\boldsymbol{\theta}_i^t$ and its optimal counterpart, i.e., $\mathbb{E}\|\boldsymbol{\theta}_i^t - \boldsymbol{\theta}_i^*(\boldsymbol{\varphi}^t)\|^2$, $\forall i$, are $\mathcal{O}(\epsilon)$, where the expectation is taken over both the randomness of data samples and the index of the iterations.

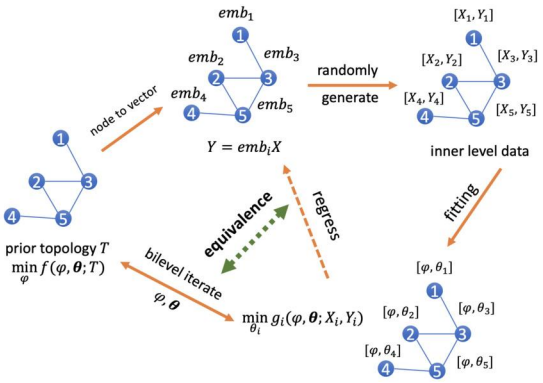

FIGURE 10.2

Homologous graph generation and data synthesis.

10.5 Numerical evaluation of GFL models

In this section, we provide numerical experiments for testing the performance of these GFL models and algorithms with a comparison with the benchmark FedAvg [18] on both synthetic and real data, which were partially shown in [25].

10.5.1 Results on synthetic data

To generate a heterogeneous network with respect to local model parameters, topology-based groundtruth weights (embedding) are generated based on a predefined graph, where inputs $\{X_i\}$ are randomly generated, and the outputs $\{Y_i\}$ are obtained by following our model in Eq. (10.13), which is further illustrated in Fig. 10.2. In order to evaluate the generalization performance of the models, we need to generate two homologous graphs so that graphs adopted in training and testing share the same distribution. To be specific, we generate two graphs that follow homologous distributions. Each of them has 50 nodes and consists of the same mixture of three standard normal distributions. The range of the centers of these distributions is $[-0.5, 0.5]$, and the standard deviation of all these distributions is 0.5. We use the four closest neighboring nodes to build the graph. The dimension of the generated samples is 64, which is used as the groundtruth weight (reshaped to 8×8) of the model for each client. For data synthesis, both training and testing sets are independently generated for the 50 LL learning tasks with uniformly distributed samples (100×8) in the range of $[-60, 60]$ as the input. The output is obtained by matrix multiplication of the input and the groundtruth weights. Parameters used in the synthetic data experiment are shown in Table 10.1.

The multi-layer perception followed by the sigmoid activation functions is used in the LL learning model, and the total number of layers of $\boldsymbol{\theta}_i$ is 3 with shapes of 8×16, 16×16, and 16×8, respectively. The UL task is link-prediction based on

Table 10.1 Synthetic experiment setting.

Common Mixture Distribution	Distribution Type	Gaussian
	Number of Center	3
	Range of Center	$[-0.5, 0.5]$
	Standard Deviation	0.5
	Dimension	64
Homologous Graph Generation	Number of Neighbor Connections	4
	Node Dimension	64
	Node Number of Train Graph	50
	Node Number of Test Graph	50
Node Data Synthesis	Shape of Ground Truth Weight	8×8
	Shape of Input	100×8
	Range of Input Distribution	$[-60, 60]$
	Shape of Input	8×100
Node (Lower Level) Model	Number of Layer	3
	Activation Function	Sigmoid
	Shape of Layers	[8×16, 16×16, 16×8]
Upper Level Model	Shape of GNN weight	512×512

negative sampling. For each head node, we randomly sample 10 tail nodes from all negative edges and use the DGL package with a GCN layer [22] to generate the graph embedding for learning the edge similarity. The GCN layer is used in the UL task, where $\boldsymbol{\varphi}$ of dimension 512×512 serves as the graph convolution kernel of the GCN layer. Each uploaded LL weight $\boldsymbol{\theta}_i$ acts as the node input representative, which is flattened into a shape of 1×512 and fitted into the cosine embedding loss function after message passing. As FeSEM requires prior knowledge about the number of clustering centers, we set $K = 3$ to reflect the setting of the three mixed Gaussian distributions.

The numerical results are shown in Fig. 10.3. It can be observed that the consensus-based models perform the worst in comparison with the cluster or graph embedding based ones, as there are three mixture Gaussian data samples over the graph. FeSEM improves the performance of classic FedAvg and DFL as the EM algorithm is able to find the three centers and tailor the model to fit different kinds of data distributions. BiG-FL shows the best performance among all the models in this case, which justifies the power of the UL learning task for extracting embedding through the graph information and transferring this similarity to the testing phrase.

10.5.2 Results on real-world data for NLP

A large lookup weight table related to the number of words in the corpus plays a central role in natural language processing (NLP) tasks. Each row of the table corre-

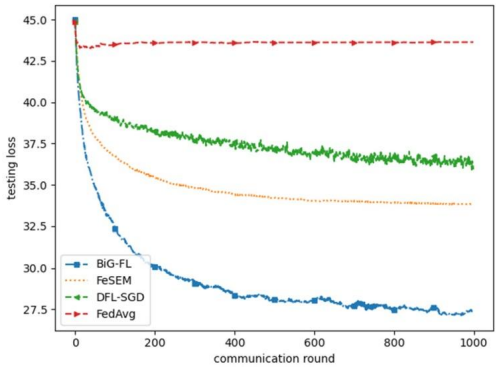

FIGURE 10.3

Performance comparison (local testing loss).

sponds to a word which is involved in the current task (i.e., word embedding). These word embeddings are trainable weights and are non-i.i.d. due to textual word data originating from different domains, and have different dimensions and interleaved weight spaces due to the differences in corpus words.

We adopt the datasets from [9] which consist of 16 different domains of review data. As shown in Table 10.2, the datasets differ in terms of review category, the average length of the review sentences, and the number of unique words. The goal of this task is to train a multi-domain next word prediction model. Here, we use the GloVe embedding vectors [19], which are commonly applied for embedding initialization to act as a global representation. We group the non-i.i.d. embeddings from multiple domains and project them into this global vector space. For the LL model, we use the classic neural probabilistic language model [1] for word embedding, where the embedding layer (i.e., a big lookup table) is with a dimension of 63144×100, a fully-connected hidden layer is with a dimension of 100×200 followed by a tanh function, and a decoder layer has a size of 200×63144. The length of the sliding window for the next word prediction task is 35 and the batch size is 100. The UL model is identical to the synthetic data case. We also set the dimension of the embedding layer for each client equal to the size of the total number of tokens for the purpose of alignment. If the current client contains a word, we set the corresponding row of the lookup table matrix of the embedding layer as trainable, and the row that does not have a corresponding word as 0. The total number of trainable rows per client is shown in the last column of Table 10.2.

The results are shown in Fig. 10.4. It can be observed that the testing losses are consistent with the synthetic dataset case. Although the testing curve of BiG-FL has some fluctuations at the early stage of the convergence, it achieves the lowest loss eventually. The reason might be due to the competition among the LL learning models, which affects the embedding search progress in the UL problem, and once the graph structure is learned, BiG-FL will showcase a stable convergence behavior. This example verifies the importance of leveraging the similarity among the word embed-

Table 10.2 Statistics of multi-domain datasets.

Dataset	Sentences	Avg Length	Unique/Total words
Books	2000	159	19,013/63,144
Electronics	1998	101	8799/63,144
DVD	2000	173	20,308/63,144
Kitchen	2000	89	9188/63,144
Apparel	2000	57	6655/63,144
Camera	1997	130	7657/63,144
Health	2000	81	8721/63,144
Music	2000	136	16,459/63,144
Toys	2000	90	8341/63,144
Video	2000	156	18,437/63,144
Baby	1900	104	8424/63,144
Magazine	1970	117	11,552/63,144
Software	1915	129	5788/63,144
Sports	2000	94	9816/63,144
IMDN	2000	269	25,195/63,144
MR	2000	21	7315/63,144

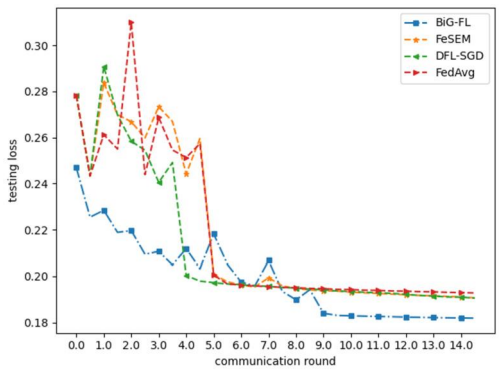

FIGURE 10.4

Real-data experiment results (testing loss).

dings in the multi-task FL problem and superiority of BiG-FL for solving this class of problems.

10.6 Summary

In this chapter, we introduced three major classes of the GFL framework, including decentralized FL, multi-center FL, and bilevel optimization enhanced FL. Each

FL model is mainly applicable for different practical FL scenarios, where DFL is designed for data samples collected in a decentralized way over a graph, multi-center FL is used for the clustered data structures, and BiG-FL is superior for integrating general topology information with adaptation to local learning tasks. Given the unique features of these GFL models, we further studied efficient gradient-based algorithms for solving the corresponding optimization problems in an FL way. It is concluded that the graph structure oriented FL systems can improve the classic FedAvg FL in terms of both learning efficiency and generalization performance.

References

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, A neural probabilistic language model, Advances in Neural Information Processing Systems 13 (2000).
[2] Tsung-Hui Chang, Mingyi Hong, Hoi-To Wai, Xinwei Zhang, Songtao Lu, Distributed learning in the nonconvex world: from batch data to streaming and beyond, IEEE Signal Processing Magazine 37 (3) (2020) 26–38.
[3] Xiaodong Cui, Songtao Lu, Brian Kingsbury, Federated acoustic modeling for automatic speech recognition, in: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 6748–6752.
[4] Alireza Fallah, Aryan Mokhtari, Asuman Ozdaglar, Personalized federated learning with theoretical guarantees: a model-agnostic meta-learning approach, in: Proceedings of Advances in Neural Information Processing Systems (NeurIPS), 2020.
[5] Saeed Ghadimi, Mengdi Wang, Approximation methods for bilevel programming, arXiv: 1802.02246, 2018.
[6] Avishek Ghosh, Jichan Chung, Dong Yin, Kannan Ramchandran, An efficient framework for clustered federated learning, in: Proceedings of Advances in Neural Information Processing Systems (NeurIPS), 2020, pp. 19586–19597.
[7] Will Hamilton, Zhitao Ying, Jure Leskovec, Inductive representation learning on large graphs, in: Proceedings of Advances in Neural Information Processing Systems (NeurIPS), 2017.
[8] Thomas N. Kipf, Max Welling, Semi-supervised classification with graph convolutional networks, arXiv:1609.02907, 2016.
[9] Pengfei Liu, Xipeng Qiu, Xuanjing Huang, Adversarial multi-task learning for text classification, arXiv:1704.05742, 2017.
[10] Rui Liu, Han Yu, Federated graph neural networks: overview, techniques and challenges, arXiv:2202.07256, 2022.
[11] Wei Liu, Li Chen, Wenyi Zhang, Decentralized federated learning: balancing communication and computing costs, IEEE Transactions on Signal and Information Processing over Networks 8 (2022) 131–143.
[12] Songtao Lu, Xiaodong Cui, Mark S. Squillante, Brian Kingsbury, Lior Horesh, Decentralized bilevel optimization for personalized client learning, in: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2022, pp. 5543–5547.
[13] Songtao Lu, Jason D. Lee, Meisam Razaviyayn, Mingyi Hong, Linearized ADMM converges to second-order stationary points for non-convex problems, IEEE Transactions on Signal Processing 69 (2021) 4859–4874.

[14] Songtao Lu, Chai Wah Wu, Decentralized stochastic non-convex optimization over weakly connected time-varying digraphs, in: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 5770–5774.
[15] Songtao Lu, Siliang Zeng, Xiaodong Cui, Mark S. Squillante, Lior Horesh, Brian Kingsbury, Jia Liu, Mingyi Hong, A stochastic linearized augmented Lagrangian method for decentralized bilevel optimization, in: Proceedings of Advances in Neural Information Processing Systems (NeurIPS), 2022.
[16] Songtao Lu, Xinwei Zhang, Haoran Sun, Mingyi Hong, GNSD: A gradient-tracking based nonconvex stochastic algorithm for decentralized optimization, in: Proceedings of IEEE Data Science Workshop (DSW), 2019, pp. 315–321.
[17] Songtao Lu, Yawen Zhang, Yunlong Wang, Decentralized federated learning for electronic health records, in: Proceedings of the 54th Annual Conference on Information Sciences and Systems (CISS), 2020.
[18] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Aguera y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS), 2017, pp. 1273–1282.
[19] Jeffrey Pennington, Richard Socher, Christopher D. Manning, GloVe: global vectors for word representation, in: Proceedings of Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543.
[20] Haoran Sun, Songtao Lu, Mingyi Hong, Improving the sample and communication complexity for decentralized non-convex optimization: joint gradient estimation and tracking, in: Proceedings of International Conference on Machine Learning, 2020, pp. 9217–9228.
[21] Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, Ji Liu, D^2: decentralized training over decentralized data, in: Proceedings of International Conference on Machine Learning (ICML), 2018, pp. 4848–4856.
[22] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, Zheng Zhang, Deep graph library: a graph-centric, highly-performant package for graph neural networks, arXiv:1909.01315, 2019.
[23] Lin Xiao, Stephen Boyd, Sanjay Lall, A scheme for robust distributed sensor fusion based on average consensus, in: Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks, 2005, pp. 63–70.
[24] Ming Xie, Guodong Long, Tao Shen, Tianyi Zhou, Xianzhi Wang, Jing Jiang, Multi-center federated learning, arXiv:2005.01026, 2020.
[25] Pengwei Xing, Songtao Lu, Lingfei Wu, Han Yu, BiG-Fed: Bilevel optimization enhanced graph-aided federated learning, IEEE Transactions on Big Data (2022).