BiG-Fed: Bilevel Optimization Enhanced Graph-Aided Federated Learning

Pengwei Xing [1] Songtao Lu [2] Lingfei Wu [3] Han Yu [1]

Abstract

Federated learning (FL) has become a useful machine learning paradigm for training models with distributedly owned sensitive data. In this paper, we extend the non-independent and identical distributions (non-i.i.d.) FL literature with the Bilevel Optimization enhanced Graph-aided Federated Learning (BiG-Fed) approach. It is designed to support applications in which FL clients residing in a network topology collaboratively train heterogeneous models (e.g., traffic prediction based on sensor networks). BiG-Fed learns a shared outer level weight in the FL server to act as graph convolutional kernel learning in order to discover the relationships among clients. Within each client, there is an independent inner level weight for improving local task optimization. To the best of our knowledge, BiG-Fed is the first bilevel optimization technique to enable future federated learning approaches to cope with two nested optimization tasks at the FL server and FL clients, respectively.

1. Introduction

In recent years, federated learning (FL) (McMahan et al., 2017), which is a distributed learning paradigm by leveraging multiple computing resources over a network, has been experiencing rapid growth. In practice, each client in the network owns personalized data features, which requires the FL algorithm to be able to deal with non-i.i.d data distributions for individual learning tasks. Personalized federated learning (PFL) (Fallah et al., 2020; Dinh et al., 2020) have been proposed to deal with the heterogeneity of client models.

[1]Nanyang Technological University, Singapore [2]IBM Thomas J. Watson Research Center Yorktown Heights, United States [3]JD Silicon Valley Research Center, United States. Correspondence to: Pengwei Xing <pengwei.xing@ntu.edu.sg>, Songtao Lu <songtao@ibm.com>, Lingfei Wu <lingfei.wu@jd.com>, Han Yu <han.yu@ntu.edu.sg>.

This work was presented at the International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML 2021 (FL-ICML'21). This workshop does not have official proceedings and this paper is non-archival. Copyright 2021 by the author(s).

Typical PFL studies (Fallah et al., 2020; Khodak et al., 2019; Jiang et al., 2019) are based on meta-learning algorithms (e.g., using model-agnostic meta-learning (MAML) (Finn et al., 2017) to realize personalization of model training). These models focus on providing good initialization for the FL clients to learn new models by only using data feature in their own datasets. Such an approach does not leverage the connectivity information and prior domain knowledge included in a population of FL clients organized under a graph structure (e.g., distributed sensor networks).

Current FL research related to graph learning mostly focuses on aggregating multiple graph neural network (GNN) models. The scenario in which FL clients are organized into graph structures (e.g., in spatio-temporal traffic prediction (Pan et al., 2019; Li et al., 2017; Zhang et al., 2017; Liang et al., 2018)) has not been investigated. Under this setting, graph embedding techniques can capture inherent information over the topology of FL clients. Such insight offers a new way for enhancing PFL to solve non-i.i.d problem with graph learning.

To achieve this goal, a graph embedding based method is required to aid the feature learning process over a network. However, introducing a new learnable weight or additional task to quantify structural correction is a nested (couple) problem with respect to the graph information and the weights of local models, where the adjusted distributed local model will be a function of the newly introduced weights at the center (i.e. the FL server). Similarly, the local model is also a function of the center weight. Bilevel optimization is a kind of optimization to solving aforementioned embedded problem which have been successfully in various scenarios including reinforcement learning (Hong et al., 2020), hyperparameter optimization (Franceschi et al., 2018) and Stackelberg game (Roth et al., 2016). In this work, we formulate this class of problem as a bilevel optimization problem, which splits the adaption step into 1) the lower level problem optimized locally and 2) graphical link prediction as the higher level problem optimized at the center.

In this paper, we propose BiG-Fed, a bilevel optimization framework combining a graph embedding learning technique to conduct nested federated learning combining aided learning tasks at the center and generalized learning tasks for local clients. It can be applied to transform existing techniques for traffic prediction from directly aggregating spatio-temporal urban tensor data into the FL setting with privacy-preserving spatio-temporal representative as shown in Figure 1.

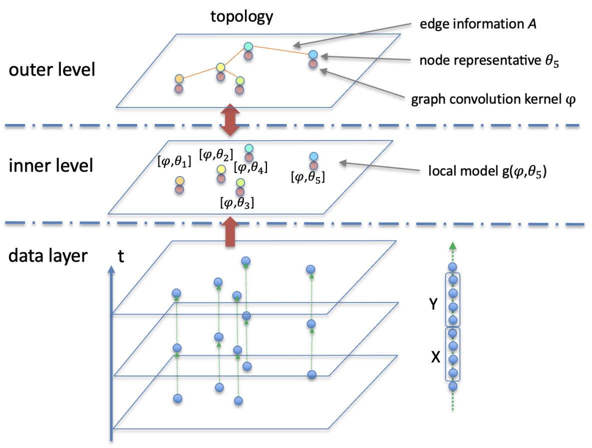

Figure 1. BiG-Fed for traffic prediction based on a spatio-temporal sensor networks.

Our contributions can be summarized as follows.

- We propose a novel paradigm to solve the nested federated learning problems via bilevel optimization. The meta learning-based and clustering-based federated learning approaches can be two specific forms of the proposed bilevel optimization-based FL paradigm.
- We enhance the Two-Timescale Stochastic Approximation (TTSA) Algorithm from (Hong et al., 2020) to achieve the purpose of converting a one-to-one bilevel optimization problem into a one-to-many federated learning problem by decoupling and chunking the inner level optimization problem. The proposed model also allows the node model to shift the computational burden of bilevel optimization to each local node.
- BiG-Fed can address the limitation that clustering-based FL methods must first specify the number of discrete clusters among data owners. It can learn node relations in the continuous space, which is more general for various types of graph structures (as shown in Figure 2). In addition, it can cope with the problem of graph topology changes during communication, benefiting from mature GNN techniques.

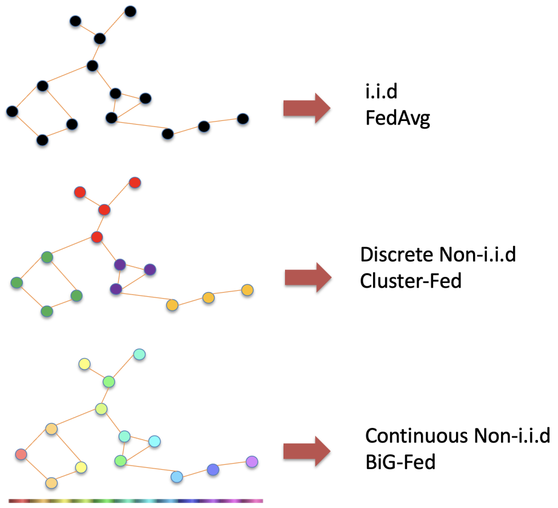

Figure 2. Comparison of different data distribution scenarios facing FL approaches.

2. Related Work

Recently, clustering-based PFL approaches (Xie et al., 2020; Sattler et al., 2020; Briggs et al., 2020) start to leverage the inherent relationships among FL clients to improve federated learning under non-i.i.d. settings. The work of (Xie et al., 2020) utilizes the classic Expectation-Maximization algorithm used in the K-means clustering technique to find K centers of the clusters by calculating distance between each model weight as multiple desired global model. Similarly, (Briggs et al., 2020) combined the hierarchical clustering algorithm with federated learning for multiple-center PFL. In (Sattler et al., 2020), the authors separates the clients into two clusters and sets a stopping criterion for deciding whether to aggregate or bisect all clients to each cluster, where this criterion is calculated after obtaining a converged solution by running the FL algorithm.

Although these works have obtained improved performance over MAML/FL based models, existing clustering-based FL approaches assume that the data distributions of the clients belong to multiple specific discrete centers, which means that they need to preset the number of clusters. Moreover, their assumption of no coupling between the central clustering task and the individual local tasks may not always hold. Specifically, methods like (Xie et al., 2020; Briggs et al., 2020) cannot prove that these two processes can achieve convergence if they are optimized independently or alternatively, unless these method can be implemented effectively under an explicitly decoupled setting like in (Sattler et al., 2020).

Laplacian matrix based PFL has also emerged for solving the problem of processing distribution correlation among the data points in the continuous space. The work of (Dinh et al., 2021) attempts to deal with the non-i.i.d issue of adopting the correlated structures among clients by employing graph Laplacian matrix as an additional regularization term in the objective function. Nevertheless, the Laplacian matrix of the graph is static. Thus, it is not able to handle cases in which the graph structure is dynamically changing over time (e.g., FL clients joining or leaving a federation during model training). We address these limitations in this work.

3. Methodology

The problem we consider in this paper is a specific form of the bilevel optimization problem under the federated settings with aided learning tasks at the center.

3.1. Problem Formulation

Let $f(\cdot)$ denote the outer level objective function that can represent the center task in FL. Let n denote the number of sub-tasks of the inner level and $\boldsymbol{g}(\cdot) = [g_i(\cdot), i \in \{1, ..., n\}]$ represent a vector for a group of objective functions at the inner level for each FL client. Similarly, φ represents the outer decision weight and $\boldsymbol{\theta} = [\theta_i, i \in \{1, ..., n\}]$ denotes the inner decision weights, where $\boldsymbol{\theta}$ represents the vector form of n inner weight. Then, we can express the two coupled optimization problems in the following form:

$$\min_{\varphi \in \mathbb{R}^{d_1}} \ell(\varphi) = f(\varphi, [\theta_i^*(\varphi) | i \in \{1, ..., n\}])$$
$$\text{subject to} \quad \boldsymbol{\theta}^*(\varphi) \in \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^{nd_2}} \boldsymbol{g}(\varphi, \boldsymbol{\theta}). \tag{1}$$

It should be noted that each inner level sub-task of g_i in each client is mutually decoupled in the FL setting, though the outer task for f is coupled with the inner tasks $\boldsymbol{g}$. Thus, we can re-express Eq. (1) as follows:

$$\min_{\varphi \in \mathbb{R}^{d_1}} \ell(\varphi) = f(\varphi, [\theta_i^*(\varphi) | i \in \{1, ..., n\}])$$
$$\text{subject to} \quad \theta_i^*(\varphi) \in \arg\min_{\theta_i \in \mathbb{R}^{d_2}} g_i(\varphi, \theta_i), \forall i \in \{1, ..., n\}. \tag{2}$$

3.2. The Proposed BiG-Fed Approach

We first consider how to design the objective function of the inner and outer objective functions for specific scenarios of solving collaborative training problems with heterogeneous data distributions and topological information. The idea is that a positive correlation exists between the outer objective function and the inner objective function (i.e. the more the local inner level model tends to converge, the more the model parameters of each local node exhibit the corresponding topological structure relationship). When this is satisfied, by accelerating the convergence of the outer objective function, and consequently the convergence rate of the inner objective function coupled to it can also be accelerated. This enhances federated learning by leveraging the graph structure information at the center.

The Outer Learning Task Inspired by unsupervised graph embedding learning for link prediction (Hamilton et al., 2017), BiG-Fed leverages the connectivity of edges as a guiding information in the outer level task by mapping into the structural similarity of neighboring node models.

Firstly, let $\varphi \in \mathbb{R}^{d_1}$ denote the outer weight acting as a GNN weight with shape of d_1 and let $\boldsymbol{\theta} \in \mathbb{R}^{nd_2}$ denote the inner weight vector including n client models θ_i with a shape of d_2. $\boldsymbol{\theta}^*$ is the inner weight vector after one step of stochastic gradient descent (SGD) update. We use $\boldsymbol{\theta}^*$ to act as node representative instead of directly accessing the data to preserve data privacy.

φ denotes the shared weight for each node representative θ_i, which can be formulated as $(\mathbf{I} \otimes \varphi)\boldsymbol{\theta}^*$ via Kronecker product $\otimes$ with an identity matrix $\mathbf{I}$, and $\mathbf{A}$ is adjacency matrix. Then, let $\mathbf{H}$ denote the embedding matrix after neighborhood averaging, which can also be replaced by other message passing in GNN, as follows:

$$\mathbf{H} = \mathbf{A}(\mathbf{I} \otimes \varphi)\boldsymbol{\theta}^*. \tag{3}$$

Then, we can formulate outer objective function by cosine embedding loss corresponding to f in Eq. (2) as follows:

$$\varphi(\boldsymbol{\theta}^*) \in \arg\min_{\varphi \in \mathbb{R}^{d_1}} f(\varphi, \boldsymbol{\theta}^*) \triangleq$$
$$\frac{1}{n}\sum_{u=1}^{n} \sum_{v \in \mathcal{N}(u)} 1 - \cos(\mathbf{H}_u, \mathbf{H}_v)$$
$$+ \frac{1}{n}\sum_{u=1}^{n} \mathbb{E}_{v \sim P_u} \max(0, \cos(\mathbf{H}_u, \mathbf{H}_v)) \tag{4}$$

where u, v is index of graph node and $\cos(\mathbf{H}_u, \mathbf{H}_v)$ denotes

$$\cos(\mathbf{H}_u, \mathbf{H}_v) = \frac{\mathbf{H}_u^T \mathbf{H}_v}{\|\mathbf{H}_u\| \|\mathbf{H}_v\|}, \tag{5}$$

and $\mathcal{P}_u$ denotes the negative sampling distribution like (Hamilton et al., 2017) at node u.

The outer objective function for BiG-Fed in Eq. (4) utilize cosine embedding loss to respectively calculate node-pair embedding similarity of all linked edges and node-pair embedding dissimilarity of negative sample. Since the embedding derived from local model weight, this objective function can couple local optimization tasks with graphical link prediction tasks.

Algorithm 1 BiG-Fed

```
 1: Initialize stepsize sequence $\{\alpha_k, \beta_k\}$, and $\varphi^0, \boldsymbol{\theta}^0$
 2: Initialize Jacobian matrix $\nabla^2_{\varphi^k, \boldsymbol{\theta}^k}\boldsymbol{G}$ and Hessian matrix $\nabla^2_{\boldsymbol{\theta}^k\boldsymbol{\theta}^k}\boldsymbol{G}$
 3: for round $k = 0, 1, 2..., K$ do
 4:     //At each FL Client:
 5:     for node $i = 0, 1, 2..., n$ do
 6:         Update $\theta_i^{k+1}$ by (9)
 7:         Calculate Jacobian block $\nabla_{\varphi^k\theta_i^k} G_i(\varphi^k, \theta_i^k; \xi)$, Hessian block $\nabla_{\theta_i^k\theta_i^k} G_i(\varphi^k, \theta_i^k; \xi)$
 8:         Send $\nabla_{\varphi^k\theta_i^k} G_i$, $\nabla_{\theta_i^k\theta_i^k} G_i$ and $\boldsymbol{\theta}^k$ to the FL Server
 9:     end for
10:     //At the FL Server:
11:     Construct $\nabla^2_{\varphi^k, \boldsymbol{\theta}^k}\boldsymbol{G}$ and $\nabla^2_{\boldsymbol{\theta}^k\boldsymbol{\theta}^k}\boldsymbol{G}$ by aggregating blocks from each node $i$
12:     Compute $\nabla_{\boldsymbol{\theta}^k} F(\varphi^k, \boldsymbol{\theta}^{k+1})$, $\nabla_{\boldsymbol{\varphi}^k} F(\varphi^k, \boldsymbol{\theta}^{k+1})$
13:     Update $\varphi^{k+1}$ by (10)
14: end for
```

The Inner Learning Task. The inner objective function of BiG-Fed corresponding to $\boldsymbol{g}$ in Eq. (2) can be formulated as follows:

$$\boldsymbol{\theta}^*(\varphi) \in \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^{nd_2}} \boldsymbol{g}(\varphi, \boldsymbol{\theta}) \tag{6}$$

where

$$\boldsymbol{g}(\varphi, \boldsymbol{\theta}) \triangleq \frac{1}{2}\mathbb{E}_{(\boldsymbol{X},\boldsymbol{Y})\sim\mathcal{D}}\|\boldsymbol{Y} - (\mathbf{I} \otimes \varphi)\boldsymbol{\theta}\boldsymbol{X}\|^2. \tag{7}$$

$\boldsymbol{X}$ and $\boldsymbol{Y}$ are the vectors denoting the data and the labels stored at the n FL clients, respectively, and $\mathcal{D}$ denotes the joint distribution of $\boldsymbol{X}$ and $\boldsymbol{Y}$. The corresponding decoupled form for each FL client i, $g_i(\cdot)$, can be expressed as:

$$\theta_i^*(\varphi) = \arg\min_{\theta_i \in \mathbb{R}^{d_2}} g_i(\varphi, \theta_i) \triangleq \frac{1}{2}\mathbb{E}_{(X_i,Y_i)\sim\mathcal{D}_i}\|Y_i - \varphi\theta_i X_i\|^2 \tag{8}$$

where X_i and Y_i are the data and the labels owned by each FL client i, respectively, and $\mathcal{D}_i$ denotes the joint distribution of X_i and Y_i.

The BiG-Fed Algorithm To take full advantage of the mutual decoupling of the inner level sub-tasks in Eq. (2), we extend the TTSA Algorithm in (Hong et al., 2020) from a one-to-one bilevel optimization algorithm into a one-to-many federated learning approach. The proposed BiG-Fed approach is shown in Algorithm 1. It is worth noting that:

- ► The training data for the outer level learning task consist of the connection information among the FL clients and the inner level model weights uploaded by the clients to the FL server. Thus, under the BiG-Fed framework, there is no exposure of any client's local data during the outer level learning task training process.
- ► Lines 5-9 in Algorithm 1 are executed by each FL client locally to update the inner level weight θ_i by solving Eq. (8) w.r.t distributed outer weight φ via one step SGD:

$$\theta_i^{k+1} = \theta_i^k - \beta_k m^{-1} \sum_{j=1}^{m} \nabla_{\theta_i^k} G_i(\varphi^k, \theta_i^k; \xi_j), \tag{9}$$

where $G_i(\varphi^k, \theta_i^k; \xi_j)$ denotes the i-th inner loss function of the j-th data sample (i.e. $\xi_j \triangleq (X_j, Y_j) \sim \mathcal{D}_j$). m stands for the mini-batch size. In addition, the Jacobian and the Hessian gradient blocks are calculated by each FL client.

- ► Lines 11-13 in Algorithm 1 are executed at the FL server where the complete Jacobian and Hessian gradient matrices are constructed. In addition, the outer level weight φ is updated by solving Eq. (4) w.r.t the uploaded inner level weights $\boldsymbol{\theta}$ via one step SGD:

$$\begin{aligned}\varphi^{k+1} =& \varphi^k - \alpha_k m^{-1} \sum_{j=1}^{m} (\nabla_{\boldsymbol{\varphi}^k} F \\ &- \nabla^2_{\varphi^k, \boldsymbol{\theta}^k}\boldsymbol{G}(\nabla^2_{\boldsymbol{\theta}^k\boldsymbol{\theta}^k}\boldsymbol{G})^{-1}\nabla_{\boldsymbol{\theta}^k} F),\end{aligned} \tag{10}$$

where F is the short notation of $F(\varphi^k, [\theta_i^k(\varphi)|i \in \{1, ..., n\}]; j)$ which denotes the outer loss function of the j-th tail node of the negative sample for head node i (i.e. $j \sim \mathcal{P}_i$). $\boldsymbol{G}$ is the concatenation of $m^{-1}\sum_{j=1}^m \nabla_{\theta_i^k} G_i(\varphi^k, \theta_i^k; \xi_j)$.

- ► Eq. (14) and Eq. (15) in the appendix contain the details of constructing Hessian and Jacobian matrices through Hessian and Jacobian blocks uploaded from FL clients in Line 11 of Algorithm 1.

4. Analytical Evaluation

It will be seen from the following Lemma 1 that the computation of $\nabla^2_{\boldsymbol{\theta\theta}}\boldsymbol{g}(\boldsymbol{\theta}, \boldsymbol{\theta})$ and $\nabla^2_{\varphi\boldsymbol{\theta}}\boldsymbol{g}(\varphi, \boldsymbol{\theta})$ can be fully decoupled under the BiG-Fed framework for the update of the outer loop variable φ.

4.1. Algorithm Efficiency

Lemma 1. *With $g_i(\varphi, \theta_i)$ as expressed in Eq. (8), and given $\varphi \in \mathbb{R}^{d_1}$ and $\boldsymbol{\theta} \in \mathbb{R}^{nd_2}$, the computation of the Jacobian matrix $\nabla^2_{\varphi\boldsymbol{\theta}}\boldsymbol{g}(\varphi, \boldsymbol{\theta})$ and the Hessian matrix $\nabla^2_{\boldsymbol{\theta\theta}}\boldsymbol{g}(\varphi, \boldsymbol{\theta})$ can be fully decoupled as follows:*

$$\nabla^2_{\boldsymbol{\theta\theta}}\boldsymbol{g}(\boldsymbol{\theta}, \boldsymbol{\theta}) = diag\{\nabla^2_{\theta_1\theta_1}\boldsymbol{g}(\varphi, \theta_1), \ldots, \nabla^2_{\theta_n\theta_n}\boldsymbol{g}(\varphi, \theta_n)\}, \tag{11}$$

$$\nabla^2_{\varphi\boldsymbol{\theta}}\boldsymbol{g}(\varphi, \boldsymbol{\theta}) = [\nabla^2_{\varphi\theta_1}\boldsymbol{g}(\phi, \theta_1), \ldots, \nabla^2_{\varphi\theta_n}\boldsymbol{g}(\varphi, \theta_n)]^T. \tag{12}$$

Proof. Please see the appendix for more details. □

Remark 1. From Lemma 1, it can be observed that by leveraging the block structure of the lower level optimization problem, the most computational burden of updating the meta learner ϕ can be distributed among the FL clients.

Remark 2. When each local inner learning problem is strongly convex (i.e. $\nabla^2_{\boldsymbol{\theta}\boldsymbol{\theta}} \boldsymbol{g}(\theta_i, \theta_i) \succ 0, \forall i$), the Hessian matrix $\nabla^2_{\boldsymbol{\theta}\boldsymbol{\theta}} \boldsymbol{g}(\varphi, \boldsymbol{\theta})$ is invertible.

4.2. Convergence Rate Guarantees

By concatenating variable θ_i, it is easy to show that the algorithm can be written as the form of the stochastic bilevel optimization algorithms shown in (Hong et al., 2020) as either a single algorithm or (Ghadimi & Wang, 2018) as a double-looped one. Following the previous theoretical analysis results, we can also claim that under mild assumptions on boundedness and continuity of the inner and outer level loss functions BiG-Fed is able to find the first-order stationary points (FOSPs) of general weakly convex outer loop bilevel optimization problems with provable convergence guarantees.

To be more specific, when the number of the inner loop updates is 1, BiG-Fed can converge to FOSPs in a rate of $\mathcal{O}(K^{-2/5})$ [Please see Theorem 2, (Hong et al., 2020)], where the step sizes of α_k and β_k are chosen in an order of $\mathcal{O}(K^{-3/5})$ and $\mathcal{O}(K^{-2/5})$, and K denotes the total number of iterations. When the number of the inner loop updates is $\mathcal{O}(\sqrt{k})$, then BiG-Fed can find FOSPs in a rate of $\mathcal{O}(K^{-1/2})$ [Please see Corollary 3.1, (Ghadimi & Wang, 2018)], where α_k is chosen in an order of $\mathcal{O}(K^{-1/2})$ and β_t is $\mathcal{O}(1/t)$ where t denotes the number of inner loops.

5. Experimental Evaluation

In order to evaluate our algorithm and eliminate other interference, we design a reverse function regression process as shown in Figure 3, in which we build 2 decision-weight ground truth functions exactly depending on a 2D-embedding generated from a real-world network. Then, we use the data generated from the ground truth function to reverse regress the functions with different distributions and graphical structure relationships.

5.1. Dataset

More specifically, we utilize the widely adopted Zachary karate club network from (Kunegis, 2013) which includes 156 edges, and a graph embedding tool (https://github.com/phanein/deepwalk) from (Perozzi et al., 2014) to generate 34 2-dimensional embedding vectors. We use one of the two dimensions to set the 34 different amplitudes, and the other dimension to set the value for the multiplier of the input variables forming 34 functions which are the ground truth of 34 time series prediction models

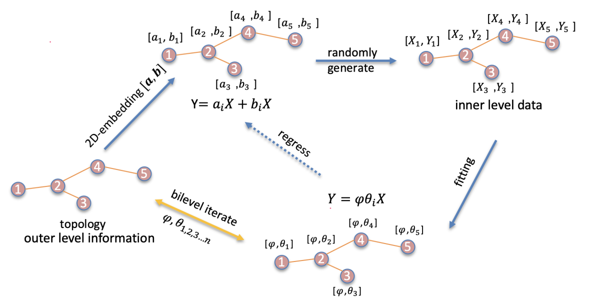

Figure 3. Experiment setup.

related to the Zachary karate club network topology. Then, with the 34 topology-related ground truth functions, we independently generate the train set and test set for the 34 inner level tasks by drawing a uniform distribution sample of 50 values in the range from -5 to 5 as the input and getting the output as labels from these ground truth functions.

Training data for the outer level include two parts. The first is the label of the edge connectivity in the form of a $[0, 1]$ vector with a dimension of 936×1 consisting of 156 connected edges and 5 negative sample for each node originated by a adjacency matrix from the Zachary karate club network. The second is an edge similarity vector corresponding to the aforementioned label vector, which is derived by the node representative matrix composed by local model weight uploaded from clients.

5.2. Model Setup

Inner Level Model The inner level model consists of multiple linear transformation layers. In our experiment, we set the total number of layers as 3 and select the middle layer as the outer weight φ with the shape of 8×8 which is distributed from FL server, and separate the decision weight θ_i into 2 sub-weights for the first layer and the last layer with the shape of 1×8 and 8×1, respectively, for accepting a 1-dimensional input and outputting a 1-dimensional predictive label.

Outer Level Model We use a graph convolutional network (GCN) layer (Wang et al., 2019) to generate the graph embedding for edge similarity learning. One GCN layer is used in the outer level task where the outer level weight φ acts as the graph convolutional kernel of the GCN layer with a dimension of 8×8, and each uploading inner node θ_i acts as the node input representative with a dimension of 8×2. The GCN outputs an embedding with a shape of 2×8, which we flatten into a 1×16 vector and fit it into the cosine embedding loss function.

5.3. Results and Discussion

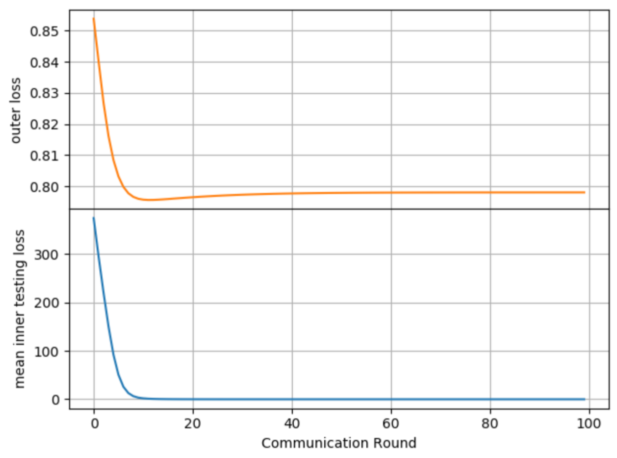

Figure 4. Positive correlation between inner level and outer level convergence.

Relevance between Inner and Outer Levels Firstly, we conduct an experiment by freezing the update of the outer level function weight, and just calculating and recording the current loss value of the outer level once the local parameter is uploaded to the server for each round. Since the smaller the outer level loss is, the closer it is to the preset topology, we just need to observe whether this outer level loss decreases together with the convergence of the inner level. As shown in Figure 4, the two curves decrease simultaneously, demonstrating the effectiveness of the outer level loss function we have designed.

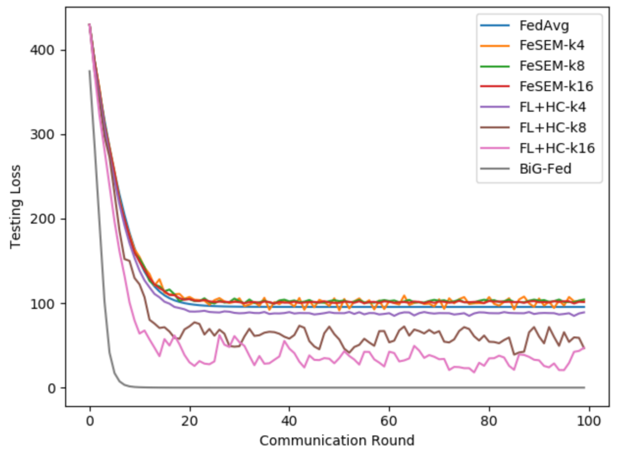

Figure 5. performance comparison of BiG-Fed with different baseline methods

Performance of BiG-Fed Since this research focuses on solving the non-i.i.d problem by adding a topological structure learner to the FL server, we compare the performance of BiG-Fed with clustering-based PFL approaches by considering the following baseline methods: a K-means cluster-based federated learning method called FeSEM (Xie et al., 2020) and an approach combining hierarchical clustering with federated learning called FL+HC (Briggs et al., 2020). The classic FedAvg (McMahan et al., 2017) is also included as a baseline.

Clustering-based FL can be considered as a special form of bilevel optimization FL. The outer level weights are the cluster center assignment labels for each node learned by the clustering algorithm, and this label is a function with respect to the local model uploaded from the inner level. Similarly, the inner level weights are essentially a function of this assignment label. Therefore, to make a fair comparison, We just keep the same shape of θ as the baseline local model parameters and use the respective clustering algorithm or FedAvg algorithm at the server. Since cluster-based methods FeSEM and FL+HC require prior knowledge about k (i.e., the number of clustering centers), we set a group baselines where $k = [4, 8, 16]$ for both FeSEM and FL+HC. As illustrated in Figure 5, the testing loss of BiG-Fed clearly drops faster than FedAvg and two groups of cluster-based methods. Moreover, benefiting from a bilevel optimization framework, BiG-Fed achieves the fastest convergence rate with the lowest testing loss.

6. Conclusions

In this paper, we propose a novel algorithm called BiG-Fed, which is a graph-aided federated bilevel optimization approach to solve the non-i.i.d problem when a graph structure exists among FL clients. We leverage prior topology information to collocate an optimization task in the center which is positively correlated with the current convergence of all local end models, thus enabling our algorithm to accelerate the convergence of the federated models. BiG-Fed is a convergence-guarantee algorithm enhancing the TTSA algorithm from a one-to-one bilevel optimization problem into a one-to-many bilevel FL problem. At the same time, utilizing the block structure of the lower level optimization problems, our algorithm can enable the computational burden of calculating high-order gradients to be distributed among the FL clients. Extensive experiments based on a real-world network demonstrate significant advantages of BiG-Fed over state-of-the-art related work.

References

Briggs, C., Fan, Z., and Andras, P. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *IJCNN*, pp. 1–9, 2020.

Dinh, C. T., Tran, N. H., and Nguyen, T. D. Personalized federated learning with moreau envelopes. *arXiv preprint arXiv:2006.08848*, 2020.

Dinh, C. T., Vu, T. T., Tran, N. H., Dao, M. N., and Zhang, H. Fedu: A unified framework for federated multi-task learning with laplacian regularization. *arXiv preprint arXiv:2102.07148*, 2021.

Fallah, A., Mokhtari, A., and Ozdaglar, A. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In *NeurIPS*, 2020.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pp. 1126–1135, 2017.

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, pp. 1568–1577. PMLR, 2018.

Ghadimi, S. and Wang, M. Approximation methods for bilevel programming. *arXiv:1802.02246*, 2018.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*, 2017.

Hong, M., Wai, H.-T., Wang, Z., and Yang, Z. A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic. *arXiv preprint arXiv:2007.05170*, 2020.

Jiang, Y., Konečnỳ, J., Rush, K., and Kannan, S. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

Khodak, M., Balcan, M.-F., and Talwalkar, A. Adaptive gradient-based meta-learning methods. *arXiv preprint arXiv:1906.02717*, 2019.

Kunegis, J. Konect: the koblenz network collection. In *WWW*, pp. 1343–1350, 2013.

Li, Y., Yu, R., Shahabi, C., and Liu, Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

Liang, Y., Ke, S., Zhang, J., Yi, X., and Zheng, Y. Geoman: Multi-level attention networks for geo-sensory time series prediction. In *IJCAI*, pp. 3428–3434, 2018.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, pp. 1273–1282, 2017.

Pan, Z., Liang, Y., Wang, W., Yu, Y., Zheng, Y., and Zhang, J. Urban traffic prediction from spatio-temporal data using deep meta learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1720–1730, 2019.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *KDD*, pp. 701–710, 2014.

Roth, A., Ullman, J., and Wu, Z. S. Watch and learn: Optimizing from revealed preferences feedback. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, pp. 949–962, 2016.

Sattler, F., Müller, K.-R., and Samek, W. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *arXiv preprint arXiv:1910.01991*, 2020.

Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., Xiao, T., He, T., Karypis, G., Li, J., and Zhang, Z. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

Xie, M., Long, G., Shen, T., Zhou, T., Wang, X., and Jiang, J. Multi-center federated learning. *arXiv preprint arXiv:2005.01026*, 2020.

Zhang, J., Zheng, Y., and Qi, D. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

A. Proof of Lemma 1

Proof. Consider $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_n]^T$ and $\boldsymbol{g}(\varphi, \boldsymbol{\theta}) = [g_1(\varphi, \theta_1), \ldots, g_n(\varphi, \theta_n)]^T$. Firstly, the gradient of $\boldsymbol{g}(\varphi, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ can be expressed as follows:

$$\frac{\partial \boldsymbol{g}(\varphi, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \left[\frac{\partial g_1(\varphi, \theta_1)}{\partial \theta_1}, \ldots, \frac{\partial g_n(\varphi, \theta_n)}{\partial \theta_n}\right]. \tag{13}$$

The corresponding Hessian matrix can be written as:

$$\nabla^2_{\boldsymbol{\theta}\boldsymbol{\theta}} \boldsymbol{g}(\boldsymbol{\theta}, \boldsymbol{\theta}) = \mathrm{diag}\{\nabla^2_{\theta_1\theta_1} \boldsymbol{g}(\varphi, \theta_1), \ldots, \nabla^2_{\theta_n\theta_n} \boldsymbol{g}(\varphi, \theta_n)\}. \tag{14}$$

Similarly, the Jacobian matrix $\nabla^2_{\varphi\boldsymbol{\theta}}$ can be expressed as:

$$\nabla^2_{\varphi\boldsymbol{\theta}} = \begin{bmatrix} \frac{\partial \nabla \boldsymbol{g}(\varphi,\boldsymbol{\theta})}{\partial \varphi_1} & \cdots & \frac{\partial \nabla \boldsymbol{g}(\varphi,\theta_1)}{\partial \varphi_m} \\ \vdots & & \vdots \\ \frac{\partial \nabla \boldsymbol{g}(\varphi,\boldsymbol{\theta})}{\partial \varphi_1} & \cdots & \frac{\partial \nabla \boldsymbol{g}(\varphi,\boldsymbol{\theta})}{\partial \varphi_m} \end{bmatrix} = \begin{bmatrix} \frac{\partial \nabla g_1(\varphi,\theta_1)}{\partial \varphi_1 \partial \theta_1} & \cdots & \frac{\partial \nabla g_1(\varphi,\theta_1)}{\partial \varphi_m \partial \theta_1} \\ \vdots & & \vdots \\ \frac{\partial \nabla g_n(\varphi,\theta_n)}{\partial \varphi_1 \partial \theta_n} & \cdots & \frac{\partial \nabla g_n(\varphi,\theta_n)}{\partial \varphi_m \partial \theta_n} \end{bmatrix} = \begin{bmatrix} \nabla^2_{\phi\theta_1} g_1(\phi, \theta_1) \\ \vdots \\ \nabla^2_{\phi\theta_n} g_n(\phi, \theta_n) \end{bmatrix}. \tag{15}$$

This completes the proof. □