Clustered Graph Federated Personalized Learning

Francois Gauthier*, Vinay Chakravarthi Gogineni*, Stefan Werner*, Yih-Fang Huang†, Anthony Kuh‡
*Dept. of Electronic Systems, Norwegian University of Science and Technology, Norway
†Dept. of Electrical Engineering, University of Notre Dame, Notre Dame, IN, USA
‡Dept. of Electrical and Computer Engineering, University of Hawai'i, Hawai'i, USA
E-mails: {francois.gauthier, vinay.gogineni, stefan.werner}@ntnu.no, huang@nd.edu, kuh@hawaii.edu

2022 56th Asilomar Conference on Signals, Systems, and Computers | 978-1-6654-5906-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/IEEECONF56349.2022.10051979

***Abstract*—This paper proposes a graph federated learning approach wherein multiple servers collaborate to enhance personalized learning over clustered clients, essentially performing correlated learning tasks. In contrast to earlier approaches, relying on a cluster-dedicated server topology, the proposed graph federated multitask learning (GFedMt) framework adopts a more general setting wherein clients of the same cluster are distributed across servers. In order to address problems with unbalanced client distributions among servers and clusters as well as data shortage of isolated clients, servers perform intra-cluster and inter-cluster learning collaboratively through local interaction with neighboring servers. Clients use the alternating direction method of multipliers (ADMM) to learn their local models. Numerical simulations demonstrate the ability of the proposed method to ensure fast and accurate convergence when data is scarce.**

***Index Terms*—Personalized federated learning, graph federated learning, multi-server architecture, inter-cluster learning.**

I. Introduction

In the age of the internet of things (IoT) and cyber-physical systems (CPS), decentralized learning has gained considerable attention as aggregating the data from geographically dispersed edge devices at a single node is unpractical. Federated learning is a decentralized learning framework in which edge devices, also called clients, train a model on their local data without exposing it to others. The local models are aggregated by a central server into a global model that is shared with the clients [1], [2]. Several studies have examined certain practical implementation challenges associated with federated learning, including communication efficiency [3], [4], privacy [5], asynchronous behavior [6], noisy communication links [7], and byzantine attacks [8]. However, these works are primarily based on a single-server architecture, which is susceptible to communication and computational bottlenecks. Additionally, single-server architectures scale poorly when working with multiple geographically dispersed clients.

A few alternatives have been proposed in the literature to address the vulnerabilities associated with the single-server architecture. Among these, client-edge-server hierarchical learning [9] recommended the use of edge servers. In the aggregation process, the edge servers perform partial aggregation with their assigned clients and communicate these partially aggregated models with the cloud server that performs the final aggregation. However, the single cloud server remains vulnerable and can only accommodate up to a certain number of edge servers. Graph federated learning (GFL) [10] advocates utilizing a distributed network of servers. Following the partial aggregation with their associated clients, the servers exchange the partially aggregated models among themselves and then perform a final aggregation to produce their own global shared models. GFL offers greater scalability than client-edge-server hierarchical learning and single-server architectures due to its distributed structure.

Learning a universal global model for geographically dispersed clients is not suitable in many IoT or CPS applications; instead, multiple models need to be learned [11]. For instance, autonomous vehicles must maintain vehicle-specific models that describe their highly dynamic environment [12]. Personalized federated learning enables clients or groups of clients performing the same task (a cluster) to learn client- or cluster-specific models [13]. Cluster-specific models usually exhibit certain similarities [14]. For instance, an autonomous vehicle's environment is shared with other devices. The learning performance can be improved through inter-cluster learning by leveraging those similarities [14], [15]. In personalized federated learning, inter-cluster learning plays a crucial role when some clients or clusters lack data [16], [17]. In [18], although personalized federated learning has been extended to a multi-server architecture, all the clients associated with a server are assumed to learn the same model. Since neighboring clients may not solve the same learning task, this assumption greatly limits the range of potential applications. The general case where each server is assigned clients from various clusters is yet to be studied.

This paper proposes a generalized graph federated personalized learning framework in which the clients associated with each server can learn their cluster-specific models. The clients of a cluster can be spread across many servers, and isolated clients must rely on inter-cluster learning until cluster-specific information reaches them. In the proposed framework, clients use ADMM to learn a local model using their data. ADMM-based learning is usually preferred to (sub)gradient-based learning when the quantity of data is limited, as it allows for faster convergence [19]. Each server receives the local models from its clients and aggregate them into cluster-specific models. Since many clusters are represented at each server, the first iteration of inter-cluster learning is performed at the server. Afterward, the servers communicate among neighbors and share their cluster-specific models. Finally, the models received from neighboring servers are aggregated with the local models for identical clusters and used in the second

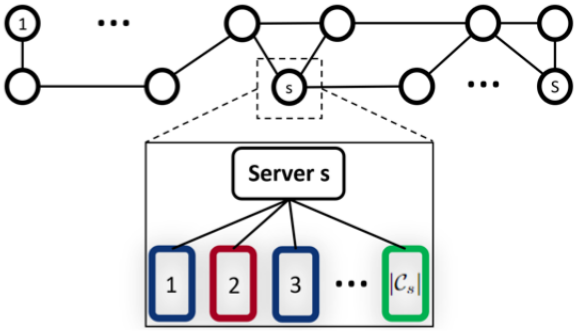


Fig. 1: Graph personalized federated learning, where servers are represented as circles and clients of the same color solve the same task.

iteration of inter-cluster learning. Numerical simulations are conducted to show that the proposed solution can ensure fast and accurate convergence to cluster-specific models when the client clusters are distributed among the servers.

II. Proposed method

We consider a multi-server architecture comprising S servers forming the set $\mathcal{S}$. The servers are connected as dictated by an undirected graph $\mathcal{G} = (\mathcal{S}, \mathcal{E})$, where the edge set $\mathcal{E}$ is such that $\mathcal{E}(s,t) = 1$ if the servers s and t are neighbors and 0 otherwise. A server s can only communicate with its neighboring servers whose set is denoted $\mathcal{N}_s$. The architecture is illustrated in Fig. 1. Each server $s \in \mathcal{S}$ is connected with its own set of clients, $\mathcal{C}_s$, so that $\sum_{s\in\mathcal{S}} \mathcal{C}_s = \mathcal{C}$. Each client $k \in \mathcal{C}_s$ learns a client-specific model $\mathbf{w}_{k,s}$. The clients associated with a server s are grouped into up to Q clusters. The clients that belong to the same cluster learn the same model, that is, $\mathbf{w}_{k,s} = \mathbf{w}_{q,s}, \forall k \in \mathcal{C}_s^q$. The models for different clusters are nonidentical, yet similar, i.e., $\mathbf{w}_{q,s} \sim \mathbf{w}_{r,s}, \forall r, s \in \{1,\dots,Q\}$.

We consider the regularized empirical risk minimization problem that is modified to take into account the correlation between the learning tasks. Each client $k \in \mathcal{C}$ has access to a data set $\mathcal{D}_k$, of cardinality $|\mathcal{D}_k| = D_k$, composed of a matrix $\mathbf{X}_k = [\mathbf{x}_{k,1},\dots,\mathbf{x}_{k,D_k}]^\mathsf{T}$ and a response vector $\mathbf{y}_k = [y_{k,1},\dots,y_{k,D_k}]^\mathsf{T}$. The optimization problem for a given cluster q can be expressed as

$$\min_{\mathbf{w}_q} \sum_{s\in\mathcal{S}} \sum_{k\in\mathcal{C}_s^q} \frac{1}{D_k} \sum_{i=1}^{D_k} \ell_k(\mathbf{x}_{k,i}, y_{k,i}; \mathbf{w}_q) + \lambda R(\mathbf{w}_q) + \tau \sum_{r\in\{1,\dots,Q\}\setminus q} ||\mathbf{w}_r - \mathbf{w}_q||_2^2, \quad (1)$$

where ℓ_k denotes the loss function of the task performed by client k, R denotes the regularizer function, $\lambda > 0$ is the regularization parameter, and $\mathbf{w}_q$ is an hypothetical global cluster-specific optimization variable. Further, $\tau > 0$ is the parameter that controls inter-cluster learning.

The aforementioned optimization problem relies on a single model per cluster. This is not feasible in the proposed multi-server architecture, where the servers are distributed on a graph and must communicate between neighbors to achieve consensus. In the proposed learning process, each server maintains local cluster-specific models, and consensus is enforced by the auxiliary variables $\{\mathbf{z}_{s,t}; \forall s,t : \mathcal{E}(s,t) = 1\}$. The optimization problem for a server s and cluster q is given by

$$\begin{aligned} \min_{\mathbf{w}_{q,s}} \quad & \sum_{k\in\mathcal{C}_s^q} \frac{1}{D_k} \sum_{i=1}^{D_k} \ell_k(\mathbf{x}_{k,i}, y_{k,i}; \mathbf{w}_{q,s}) + \lambda R(\mathbf{w}_{q,s}) \\ & + \tau \sum_{r\in\{1,\dots,Q\}\setminus q} \sum_{t\in\mathcal{N}_s} ||\mathbf{w}_{r,t} - \mathbf{w}_{q,s}||_2^2, \quad (2) \\ \text{s.t.} \quad & \mathbf{w}_{q,s} = \mathbf{z}_{s,t}, \mathbf{w}_{q,t} = \mathbf{z}_{s,t}; \mathcal{E}(s,t) = 1, \end{aligned}$$

where $\mathbf{w}_{q,s}$ denotes the model of server s for cluster q.

From (2), we can derive the augmented Lagrangian with the set of primal variables $\mathcal{V}_q = \{\mathbf{w}_{q,s}; s \in \mathcal{S}\}$, Lagrange multipliers $\mathcal{M} = (\{\mu_{s,t}\}, \{\gamma_{s,t}\})$, and auxiliary variables $\mathcal{Z} = \{\mathbf{z}_{s,t}\}$ as:

$$\begin{aligned} \mathcal{L}_{\rho,q}(\mathcal{V}_q, \mathcal{M}, \mathcal{Z}) = & \sum_{s\in\mathcal{S}} \Big(\sum_{k\in\mathcal{C}_s^q} \frac{\ell_k(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}_{q,s})}{D_k} + \lambda R(\mathbf{w}_{q,s}) \\ & + \tau \sum_{r\in\{1,\dots,Q\}\setminus q} \sum_{t\in\mathcal{N}_s} ||\mathbf{w}_{r,t} - \mathbf{w}_{q,s}||_2^2 \Big) \\ & + \sum_{s\in\mathcal{S}} \sum_{t\in\mathcal{N}_s} \Big(\mu_{s,t}^\mathsf{T}(\mathbf{w}_{q,s} - \mathbf{z}_{s,t}) + \gamma_{s,t}^\mathsf{T}(\mathbf{w}_{q,t} - \mathbf{z}_{s,t}) \Big) \\ & + \frac{\rho}{2} \sum_{s\in\mathcal{S}} \sum_{t\in\mathcal{N}_s} \Big(||\mathbf{w}_{q,s} - \mathbf{z}_{s,t}||_2^2 + ||\mathbf{w}_{q,t} - \mathbf{z}_{s,t}||_2^2 \Big) \quad (3) \end{aligned}$$

where ρ is the penalty parameter. Given that the Lagrange multipliers are initialized to zero, by using the Karush-Kuhn-Tucker conditions of optimality and setting $\gamma_s = 2\sum_{t\in\mathcal{N}_s} \gamma_{s,t}$, it can be shown that the Lagrange multipliers $\mu_{s,t}$ and the auxiliary variables $\mathcal{Z}$ are eliminated [20].

From the above Lagrangian, it is possible to derive the local update steps of the ADMM for clients and servers. For a given client $k \in \mathcal{C}_q^s$, the primal and dual updates are given by:

- **Client primal update**

$$\begin{aligned} \mathbf{w}_k^{(n)} = & \arg\min_{\mathbf{w}} \frac{1}{D_k} \ell_k(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}) + \frac{\lambda}{|\mathcal{C}_s|} R(\mathbf{w}) \quad (4) \\ & - \Big\langle \chi_k^{(n-1)}, \mathbf{w} - \mathbf{w}_{q,s}^{(n-1)} \Big\rangle + \frac{\rho}{2} ||\mathbf{w} - \mathbf{w}_{q,s}^{(n-1)}||_2^2, \end{aligned}$$

- **Client dual update**

$$\chi_k^{(n)} = \chi_k^{(n-1)} + \rho(\mathbf{w}_{q,s}^{(n)} - \mathbf{w}_k^{(n)}), \quad (5)$$

where (n) denotes the current iteration.

Further, the local updates for a server $s \in \mathcal{S}$ are given by:

- **Server primal update**

$$\begin{aligned}\mathbf{w}_{q,s}^{(n)} =& \arg\min_{\mathbf{w}} \Bigg[\left\| \mathbf{w} - \hat{\mathbf{w}}_{q,s}^{k} \right\|_2^2 \\ &+ \tau \left\| \mathbf{w} - \frac{1}{Q-1} \sum_{r \in \{1,\ldots,Q\} \setminus q} \mathbf{w}_{r,s}^{(n-1)} \right\|_2^2 \\ &+ \tau \left\| \mathbf{w} - \frac{1}{Q-1} \sum_{r \in \{1,\ldots,Q\} \setminus q} \frac{1}{|\mathcal{N}_s|} \sum_{t \in \mathcal{N}_s} \mathbf{w}_{r,t}^{(n-1)} \right\|_2^2 \\ &+ \mathbf{w}^{\mathsf{T}} \gamma_{q,s}^{(n-1)} \\ &+ \rho \sum_{t \in \mathcal{N}_s} \left\| \mathbf{w} - \frac{\mathbf{w}_{q,s}^{(n-1)} - \mathbf{w}_{q,t}^{(n-1)}}{2} \right\|_2^2 \Bigg], \end{aligned} \tag{6}$$

- **Server dual update**

$$\gamma_{q,s}^{(n)} = \gamma_{q,s}^{(n-1)} + \rho \sum_{t \in \mathcal{N}_s} (\mathbf{w}_{q,t}^{(n)} - \mathbf{w}_{q,s}^{(n)}), \tag{7}$$

where $\hat{\mathbf{w}}_{q,s}^{k}$ is given by

$$\hat{\mathbf{w}}_{q,s}^{k} = \frac{1}{|\mathcal{C}_s^q|} \sum_{k \in \mathcal{C}_s^q} \mathbf{w}_k^{(n)} - \frac{1}{\rho |\mathcal{C}_s^q|} \sum_{k \in \mathcal{C}_s^q} \chi_k^{(n-1)}. \tag{8}$$

The above primal update performs aggregation, at the server inter-cluster learning, inter-server aggregation, and inter-server inter-cluster learning in a single operation. In addition to the high computational complexity, doing so limits the added value of inter-cluster learning since it has to rely on the estimates from the previous primal update ($\mathbf{w}_{r,s}^{(n-1)}$). To address this, the primal and dual updates at the servers are replaced with a multi-step update mechanism. In the proposed solution, the following updates are computed consecutively.

- **Aggregation**

$$\mathbf{w}_{q,s}^{(n)} = \frac{1}{|\mathcal{C}_s^q|} \sum_{k \in \mathcal{C}_s^q} \mathbf{w}_k^{(n)} - \frac{1}{\rho |\mathcal{C}_s^q|} \sum_{k \in \mathcal{C}_s^q} \chi_k^{(n-1)}. \tag{9}$$

- **Inter-cluster learning**

$$\mathbf{w}_{q,s}^{(n)} = \frac{\mathbf{w}_{q,s}^{(n)} + \tau \sum_{r \in \{1,\ldots,Q\} \setminus q} \mathbf{w}_{r,s}^{(n)}}{1 + \tau (Q-1)}. \tag{10}$$

- **Inter-server update**

$$\mathbf{w}_{q,s}^{(n)} = \frac{\mathbf{w}_{q,s}^{(n)} + \sum_{t \in \mathcal{N}_s} \mathbf{w}_{q,t}^{(n)}}{|\mathcal{N}_s| + 1}. \tag{11}$$

- **Inter-server inter-cluster learning**

$$\mathbf{w}_{q,s}^{(n)} = \frac{\mathbf{w}_{q,s}^{(n)} + \tau \sum_{r \in \{1,\ldots,Q\} \setminus q} \sum_{t \in \mathcal{N}_s} \mathbf{w}_{r,t}^{(n)}}{1 + \tau |\mathcal{N}_s| (Q-1)}. \tag{12}$$

The above decomposition has several advantages. First, it allows inter-cluster learning to rely on the last available estimates, both at the server and inter-server. Second, it increases the servers learning rate as ρ is discarded on the server-side. Finally, it enables the servers to share an intermediary result with the clients so that clients and servers computations happen in parallel.

III. Algorithm Derivation

To solve the problem (2), we propose the following multi-layer learning process. Consider a server $s \in \mathcal{S}$, and an iteration n of the algorithm. To begin, all clients perform ADMM-based batch learning on their locally available data. A client $k \in \mathcal{C}_s^q$ updates its local primal variable as follows

$$\begin{aligned}\mathbf{w}_k^{(n)} =& \arg\min_{\mathbf{w}} \frac{1}{D_k} \ell_k(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}) + \frac{\lambda}{|\mathcal{C}_s|} R(\mathbf{w}) \\ &- \left\langle \chi_c^{(n-1)}, \mathbf{w} - \mathbf{w}_{q,s}^{(n-1)} \right\rangle + \frac{\rho}{2} ||\mathbf{w} - \mathbf{w}_{q,s}^{(n-1)}||_2^2, \end{aligned} \tag{13}$$

where $\mathbf{w}_{q,s}^{(n-1)}$ is the existing model of server s for cluster q. Following the primal update, the clients share their primal and dual variables with their associated server.

In a second step, the servers aggregate the primal and dual variables received from their associated clients to compose their models. Server s updates its model for cluster q by performing the following aggregation

$$\mathbf{w}_{q,s}^{(n)} = \frac{1}{|\mathcal{C}_s^q|} \sum_{k \in \mathcal{C}_s^q} \mathbf{w}_k^{(n)} - \frac{1}{\rho |\mathcal{C}_s^q|} \sum_{k \in \mathcal{C}_s^q} \chi_k^{(n-1)}. \tag{14}$$

Further, the servers refine their cluster-specific models by performing inter-cluster learning. The purpose of this step is to leverage the similarities between the cluster-specific models to improve the learning performance. Server s refines its model for cluster q by computing

$$\mathbf{w}_{q,s}^{(n)} = \frac{\mathbf{w}_{q,s}^{(n)} + \tau \sum_{r \in \{1,\ldots,Q\} \setminus q} \mathbf{w}_{r,s}^{(n)}}{1 + \tau (Q-1)}. \tag{15}$$

Once a server has refined its cluster-specific models, they are shared with its associated clients for them to update their dual variables. A client $k \in \mathcal{C}_s^q$ receives the model $\mathbf{w}_{q,s}^{(n)}$ and updates its dual variable as

$$\chi_k^{(n)} = \chi_k^{(n-1)} + \rho (\mathbf{w}_{q,s}^{(n)} - \mathbf{w}_k^{(n)}). \tag{16}$$

Concurrently with the dual variable update at the clients, the servers share their models among neighbors to reach consensus. Server s receives the models of the servers in $\mathcal{N}_s$ and refine its model for cluster q by computing

$$\mathbf{w}_{q,s}^{(n)} = \frac{1}{|\mathcal{N}_s| + 1} (\mathbf{w}_{q,s}^{(n)} + \sum_{t \in \mathcal{N}_s} \mathbf{w}_{q,t}^{(n)}). \tag{17}$$

Since a server receives the models of its neighbors for all clusters, it can at no extra cost use the received models to perform another inter-cluster learning step. The server s refines its model for cluster q by using the models received from its neighbors for all other clusters with

$$\mathbf{w}_{q,s}^{(n)} = \frac{\mathbf{w}_{q,s}^{(n)} + \tau \sum_{r \in \{1,\ldots,Q\} \setminus q} \sum_{t \in \mathcal{N}_s} \mathbf{w}_{r,t}^{(n)}}{1 + \tau |\mathcal{N}_s| (Q-1)}. \tag{18}$$

This concludes the learning process of the proposed graph federated multitask learning (GFedMtl). The resulting algorithm is presented in Algorithm 1.

Algorithm 1 GFedMtl

Initialization: $\mathbf{w}_k^{(0)}$ and $\mathbf{w}_{q,s}^{(0)}, \forall k, q, s$ set to $\mathbf{0}$
– Procedure at client k –
For $n = 1, 2, \ldots, N$
  Update $\mathbf{w}_k^{(n)}$ with (4)
  Share $\mathbf{w}_k^{(n)}$, receive $\mathbf{w}_{q,s}^{(n)}$
  Update $\chi_k^{(n)}$ with (5)
EndFor
– Procedure at server s –
For $n = 1, 2, \ldots, N$
  Update $\mathbf{w}_{q,s}^{(n)}$ with (9)
  Refine $\mathbf{w}_{q,s}^{(n)}$ with (10)
  Share $\mathbf{w}_{q,s}^{(n)}$, receive $\{\mathbf{w}_{r,t}^{(n)}; \forall t \in \mathcal{N}_s, \forall r\}$
  Aggregate $\mathbf{w}_{q,s}^{(n)}$ with (11)
  Refine $\mathbf{w}_{q,s}^{(n)}$ with (12)
  Share $\mathbf{w}_{q,s}^{(n)}$, receive $\{\mathbf{w}_k^{(n+1)}; \forall k \in \mathcal{C}_s\}$
EndFor

IV. Numerical Simulations

To demonstrate the performance of the proposed graph personalized federated learning, we have conducted simulations on the ridge regression problem. The loss function is given by $\ell_k(\mathbf{X}_k, \mathbf{y} = K; \mathbf{w}) = ||\mathbf{y}_k - \mathbf{X}_k\mathbf{w}||_2^2$ and the regularizer function by $R(\mathbf{w}) = ||\mathbf{w}||_2^2$.

We consider $|\mathcal{S}| = 10$ servers distributed over a graph, each is associated with $|\mathcal{C}_s| = 15; s \in \{1, \ldots, 10\}$ clients. Each set $\mathcal{C}_s$ comprises clients that may belong to $Q = 3$ different clusters. Every client aim to learn its cluster-specific model $\mathbf{w}_q$, generated by $\mathbf{w}_q = \mathbf{w}_0 + \sigma_q \mathbf{w}_0$. Where $\mathbf{w}_0$ is a randomly generated base model and $\sigma_q \in \mathcal{U}(-0.1, 0.1)$, $\mathcal{U}$ denoting the uniform distribution. Each client k possesses a training dataset composed of anywhere between 1 and 9 data samples of dimension 60. The training response vector for client k that belongs to cluster q is given by $\mathbf{y}_k = \mathbf{X}_k\mathbf{w}_q + \mathbf{n}_k$, where $\mathbf{n}_k \in \mathcal{N}(\mathbf{0}, \eta_k \mathbf{I}_{D_k})$ with η_k a client-specific noise variance. For each client, η_k is randomly selected so that the network's data is non i.i.d. for each cluster.

In addition to training data, each client possesses a testing dataset, independent but identically distributed to its training dataset. The testing data is aggregated for each cluster to compute an estimate of the optimal cluster model, $\bar{\mathbf{w}}_q$. The metric used to compare the performance of the algorithms is the mean-squared error (MSE). For each client belonging to cluster q, the MSE is computed with respect to $\bar{\mathbf{w}}_q$.

$$\text{Test MSE} = \frac{1}{|\mathcal{C}|} \sum_{k=1}^{|\mathcal{C}|} \frac{\|\mathbf{w}_k - \bar{\mathbf{w}}_q\|_2^2}{\|\bar{\mathbf{w}}_q\|_2^2}. \tag{19}$$

In the first experiment, we studied the performance of the following algorithms.

- **GFed** is defined in [10] and corresponds to conventional federated learning adapted to the multi server architecture, it learns a single universal model.
- **GFedMtl** is the proposed algorithm that learns cluster-specific models. The parameter τ controls inter-cluster learning, if $\tau = 0$ the clusters do not exchange information with one another. If the edge set $\mathcal{E} = \mathbf{0}$, there is no cooperation between servers.

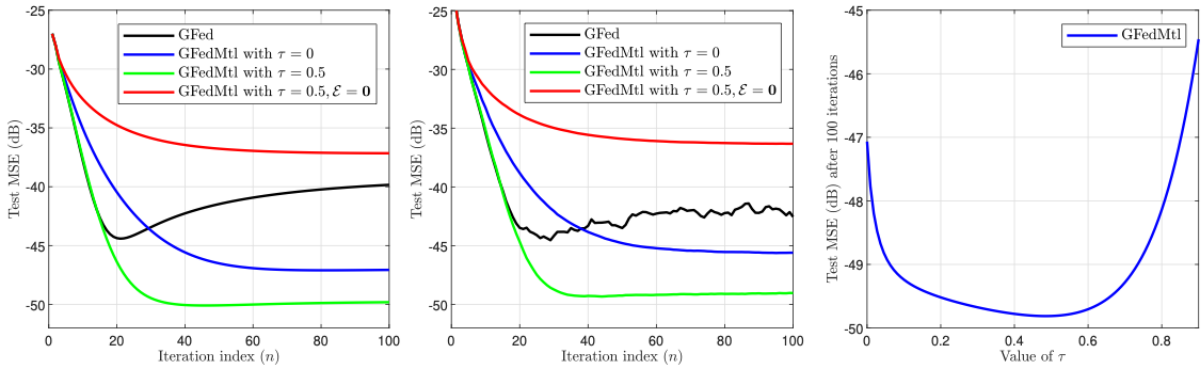
The learning curves (Test MSE in dB vs. iteration index n) are shown in Fig. 2 (a). The figure shows that GFed fails to achieve satisfactory accuracy, this is due to the fact that it tries to learn a single universal model that cannot accommodate the varied learning tasks. The proposed GFedMtl with $\tau = 0$ achieves better steady-state accuracy by learning cluster-specific models. However, it suffers from low convergence speed because isolated clients need wait for cluster-specific estimates to arrive after many hops. The proposed GFedMtl with $\tau = 0.5$ overcomes this issue and offers the same convergence speed as GFed. In addition to increasing convergence speed, the use of inter-cluster learning also increases the steady-state accuracy, allowing this method to outperform both GFed and GFedMtl with $\tau = 0$. For illustration purposes, the proposed GFedMtl with $\tau = 0.5$ and $\mathcal{E} = \mathbf{0}$ is displayed. Its poor performance shows that the clients associated with a given server do not have sufficient data to learn satisfactory models on their own, even when using inter-cluster learning. This confirms that the other algorithms successfully aggregate the information coming from other servers.

In the second experiment, we studied the impact of client scheduling on the performance of the aforementioned algorithms. Client scheduling is usually used to reduce the communication and computation burden on the clients. In this experiment, each server is allowed to perform partial aggregation with a subset of 9 randomly selected clients, the other hyper-parameters are untouched. From Fig. 2 (b), we see that although client scheduling has a negative impact on accuracy, this impact is minimal. The GFed algorithm using client scheduling suffers from extensive randomness as the proportion of clients from each cluster varies at each random selection, and this algorithm learns a single model.

It is important to note that the choice of the parameter τ, which controls inter-cluster learning, impacts the performance of the proposed GFedMtl. To illustrate this fact, the third experiment, displayed in Fig. 2 (c), shows the test MSE after 100 iterations vs. the value of τ. We recognize the test MSE attained by GFedMtl $\tau = 0$ and GFedMtl $\tau = 0.5$ in Fig. 2 (a). The figure shows that although most τ values offer an improvement over $\tau = 0$, there is an optimal value for this parameter, and over-using inter-cluster learning leads to performance degradation and should be avoided. There is a need for intelligent controlled inter-cluster learning that adapts the value of τ to the needs of the system automatically.

V. Conclusions

The proposed method handles the general case of graph personalized federated learning, where each server is assigned clients from several clusters. Estimates are exchanged between clients and servers and through the multi-server architecture



(a) (b) (c)

Fig. 2: Learning curves of GFed and the proposed GFedMtl with various settings: (a) without client scheduling, (b) with client scheduling. (c) Seady-state accuracy versus value of the inter-cluster learning parameter τ.

to ensure the aggregation of cluster-specific models. Inter-cluster learning is used at the server and inter-servers to speed up the learning process and increase learning accuracy. The ADMM learning steps are modified so that inter-cluster learning benefits from the last available estimates and to increase further the convergence speed. Numerical simulations show that the proposed method can ensure fast and accurate learning of personalized models when data is scarce and each cluster is spread across several servers.

ACKNOWLEDGEMENT

The Research Council of Norway supported this work.

REFERENCES

[1] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, Oct. 2016.

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," *Artificial Intel. Statis.*, pp. 1273–1282, Apr. 2017.

[4] V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Communication-Efficient Online Federated Learning Strategies for Kernel Regression," *IEEE Internet Things J.*, pp. 1–1, Nov. 2022.

[5] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 3454–3469, Apr. 2020.

[6] F. Gauthier, V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Resource-aware asynchronous online federated learning for nonlinear regression," *IEEE Int. Conf. Commun.*, pp. 2828–2833, May 2022.

[7] F. Ang, L. Chen, N. Zhao, Y. Chen, W. Wang, and F. R. Yu, "Robust federated learning with noisy communication," *IEEE Trans. Commun.*, vol. 68, no. 6, pp. 3452–3464, Mar. 2020.

[10] E. Rizk and A. H. Sayed, "A graph federated architecture with privacy preserving learning," *IEEE Int. Workshop Signal Process. Advances Wireless Commun.*, pp. 131–135, Sep. 2021.

[8] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2168–2181, Jul. 2020.

[9] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," *IEEE Int. Conf. Commun.*, pp. 1–6, Jun. 2020.

[11] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Trans. Neural Networks Learning Syst.*, Mar. 2022.

[12] B. Yang, X. Cao, K. Xiong, C. Yuen, Y. L. Guan, S. Leng, L. Qian, and Z. Han, "Edge intelligence for autonomous driving in 6G wireless system: Design challenges and solutions," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 40–47, Apr. 2021.

[13] V. C. Gogineni, S. Werner, F. Gauthier, Y.-F. Huang, and A. Kuh, "Personalized online federated learning for IoT/CPS: challenges and future directions," *IEEE Internet Things Mag.*, 2022.

[14] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks," *IEEE Trans. Signal Process.*, vol. 62, no. 16, pp. 4129–4144, Jun. 2014.

[15] V. C. Gogineni and M. Chakraborty, "Improving the performance of multitask diffusion APA via controlled inter-cluster cooperation," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 67, no. 3, pp. 903–912, Dec. 2019.

[16] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *Advances Neural Info. Process. Syst.*, vol. 33, pp. 19 586–19 597, 2020.

[17] D. Caldarola, M. Mancini, F. Galasso, M. Ciccone, E. Rodolà, and B. Caputo, "Cluster-driven graph federated learning over multiple domains," *Proc.IEEE Conf. Comput. Vision Pattern Recognit.*, pp. 2749–2758, 2021.

[18] V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Decentralized graph federated multitask learning for streaming data," *Annu. Conf. Inf. Sciences Syst.*, pp. 101–106, Mar. 2022.

[19] S. Zhou and G. Y. Li, "Communication-efficient ADMM-based federated learning," *arXiv preprint arXiv:2110.15318*, Oct. 2021.

[20] G. B. Giannakis, Q. Ling, G. Mateos, and I. D. Schizas, "Splitting Methods in Communication, Imaging, Science, and Engineering," in *Scientific Computation*. Springer Int. Publishing, Jan. 2017, pp. 461–497.