Networked Personalized Federated Learning Using Reinforcement Learning

Francois Gauthier, Vinay Chakravarthi Gogineni, Stefan Werner
Dept. of Electronic Systems, Norwegian University of Science and Technology, Norway
E-mails: {francois.gauthier, vinay.gogineni, stefan.werner}@ntnu.no

***Abstract*—Personalized federated learning enables every edge device or group of edge devices within the distributed network to learn a device- or cluster-specific model tailored to their local needs. Data scarcity, however, makes it difficult to learn such individual models, resulting in performance degradation. Since the device- or cluster-specific tasks are distinct but often related, leveraging these similarities through inter-cluster learning alleviates data shortage and enhances learning performance. Although inter-cluster learning can boost performance, uncontrolled inter-cluster learning may lead to performance degradation due to over- or under-usage of local similarity enforcement. In light of this issue, an intelligent mechanism that performs inter-cluster learning based on device-specific needs is required. To this end, this paper proposes adopting reinforcement learning principles to control device-specific inter-cluster learning in real-time. We propose networked personalized federated learning using reinforcement learning (NPFed-RL) as a general framework and then demonstrate its feasibility by applying it to the ridge regression problem. We conduct numerical experiments to compare the proposed method with the state-of-the-art. The proposed method successfully controls device-specific parameters and offers better learning performance than existing solutions.**

***Index Terms*—Personalized federated learning, networked federated learning, distributed learning, inter-cluster learning.**

I. Introduction

Federated learning (FL) is a distributed learning paradigm that enables geographically dispersed edge devices, often called clients, to learn a global shared model on their locally stored data without revealing it [1]. FL received enormous attention due to the ability to handle system and statistical heterogeneity. System heterogeneity refers to the various computational and communication capacities of the participating devices [2]. Statistical heterogeneity implies that data are imbalanced and non-i.i.d. across devices [3]. Several challenges associated with the practical implementation of FL, such as communication efficiency [4], privacy preservation [5], byzantine attacks [6], and asynchronous behavior of devices and communication links [7], have been studied extensively in the literature. In contrast to single-server methodologies, multi-server architectures [8], [9] and fully distributed architectures [10] have also been studied recently. This paper deals with a fully distributed architecture.

In many practical applications, a network of clients must learn more than one model. Those models might be different, yet they exhibit similarities [11]. For instance, networked vehicles need to learn and maintain customized models of surrounding environments to plan trajectories [12]. Likewise, patient-specific models in healthcare are required for better diagnosis and treatment [13]. Personalized federated learning fulfills the requirements of these applications by allowing each client, or a group of clients (referred to as a cluster), to learn client- or cluster-specific models [14], [15]. Building personalized models for clusters is challenging, as they often have access to limited data. Inter-cluster learning, i.e., cooperation across the clusters, can alleviate data scarcity [16].

Although inter-cluster learning can improve learning accuracy and speed by exploiting similarities across personalized models, it can also have the opposite effects if not adequately controlled [17]. In particular, a fixed regularization parameter cannot effectively control inter-cluster learning for all clients since system heterogeneity and network topology affect local sensitivity to data shortages. For example, clients will benefit more from inter-cluster learning if neighbors, and clients within a few hops, mainly belong to different clusters, slowing down the propagation of cluster-specific information through the network. However, as consensus is reached within a group of clients from a given cluster, the need for inter-cluster learning diminishes. For these reasons, there is a need for an intelligent mechanism that controls client-specific inter-cluster learning parameters in real-time so that it is only used when it improves performance [18]. To that end, this paper develops a reinforcement learning-based mechanism that locally controls client-specific inter-cluster learning parameters on-the-fly.

Personalized distributed learning algorithms use a fixed inter-cluster learning parameter and suffer from the above-mentioned limitations [16], [17]. In [18], an ad-hoc rule has been proposed to disable inter-cluster learning if it becomes detrimental rather than adapting its impact according to client requirements. For this purpose, every model received from neighbors belonging to a different cluster is tested against the local training dataset. This process results in a substantial computational cost that grows with the network density, thus prohibitive for practical usage. Reinforcement learning has been used in distributed single-task learning [19], [20] and multi-task learning [21]–[23], but they have yet to focus on inter-cluster learning. In the field of evolutionary computing, reinforcement learning has been proven effective for parameter control [24], suggesting its potential for real-time control of inter-cluster learning parameters in personalized FL.

This paper proposes the NPFed-RL algorithm, where clients use the alternating direction method of multipliers (ADMM) and interact with neighbors to learn cluster-specific models. Inter-cluster learning is used to improve those models further.

Each client uses reinforcement learning to adapt its inter-cluster learning parameter on-the-fly so that task similarity with neighboring clients is only leveraged when it is beneficial for local learning. Two reinforcement learning policies are developed and studied: a deterministic policy gradient and a stochastic actor-critic policy based on an estimate of the action-value function. The proposed policies are computationally inexpensive and improve the convergence properties of networked personalized federated learning. The proposed NPFed-RL will first be derived as a framework and then used to solve the ridge regression problem as proof of concept. Finally, numerical simulations will be conducted to demonstrate the performance of the proposed NPFed-RL and to observe the evolution of the inter-cluster learning parameters in the proposed method.

II. Networked Personalized Federated Learning

We consider a fully distributed network modeled as an undirected graph $\mathcal{G} = (\mathcal{C}, \mathcal{E})$, where $\mathcal{C}$ is the set of clients and $\mathcal{E}$ is the set of edges such that $\mathcal{E}(k,l) = 1$ if the clients k and l are neighbors and 0 otherwise. A client can only communicate with its neighbors, we denote $\mathcal{N}_k$ the set of the neighbors of client k. Further, clients are grouped into Q clusters, and the clients grouped in a cluster q, denoted by $\mathcal{C}_q$, for $q \in \{1, \dots, Q\}$, carry out the same task, i.e., they aim to learn the same model. For $r \neq q, (r,q) \in \{1, \dots, Q\}$, the tasks associated with cluster q and r are different, but similar. To warrant the use of inter-cluster learning, we take the following assumption on task similarity, where $\mathbf{w}_q^*$ denotes the optimal model for cluster q.

Assumption 1. $\left\|\mathbf{w}_q^* - \mathbf{w}_r^*\right\|_2^2 \leqslant \eta, \forall q, r \in \{1, \dots, Q\}$

Each client $k \in \mathcal{C}$ has access to a proprietary dataset $(\mathbf{X}_k, \mathbf{y}_k)$ composed of a matrix $\mathbf{X}_k = [\mathbf{x}_{k,1}, \dots, \mathbf{x}_{k,D_k}]^\mathsf{T}$ and a response vector $\mathbf{y}_k = [y_{k,1}, \dots, y_{k,D_k}]^\mathsf{T}$, where D_k is the number of data samples available to client k. The objective is to estimate each cluster-specific model as accurately as possible without leaking data. This leads to the following optimization problem for a given cluster q:

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

where ℓ_k denotes the loss function of the task performed by client k, R denotes the regularizer function, and $\lambda > 0$ is the regularization parameter. The term on the second line corresponds to the enforcement of similarity between the cluster-specific models, it is controlled by the global parameter τ. A larger τ enforces more similarity, leading to more similar cluster-specific models. This optimization problem uses a global model $\mathbf{w}_q$ for each cluster, this is not feasible in the proposed setting.

To circumvent this difficulty, the learning process must rely on the clients' models and enforce consensus among these models. To do so, the auxiliary variables $\mathbf{z}_k^l, \forall (k,l) \in \mathcal{C}_q : \mathcal{E}(k,l) = 1$ are introduced. The optimization problem for a given client k belonging to cluster q is then given by

$$\begin{aligned} \min_{\mathbf{w}_{k,q}} \quad & \frac{1}{D_k} \sum_{i=1}^{D_k} \ell_k(\mathbf{x}_{k,i}, y_{k,i}; \mathbf{w}_{k,q}) + \lambda R(\mathbf{w}_{k,q}) \\ & + \frac{\tau_k}{Q-1} \sum_{r \in \{1,\dots,Q\} \setminus \mathrm{q}} \left\|\hat{\mathbf{w}}_{k,r} - \mathbf{w}_{k,q}\right\|^2, \\ \text{s.t.} \quad & \mathbf{w}_{k,q} = \mathbf{z}_k^l, \mathbf{w}_{l,q} = \mathbf{z}_k^l; \forall (k,l) \in \mathcal{C}_q : \mathcal{E}(k,l) = 1, \end{aligned} \quad (2)$$

where $\mathbf{w}_{k,q}$ denotes the model of client k belonging to cluster q, and the constraints enforce intra-cluster consensus. The global parameter τ is replaced by client-specific parameters τ_k that control inter-cluster learning locally. $\hat{\mathbf{w}}_{k,r}$ denotes the best available estimate of the model for cluster r available at client k. This corresponds to the average of the models of the neighboring clients from the cluster in question, given by

$$\hat{\mathbf{w}}_{k,r} = \frac{1}{|\mathcal{N}_k \bigcap \mathcal{C}_r|} \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_r} \mathbf{w}_{l,r}. \quad (3)$$

It is possible to derive the augmented Lagrangian for a given cluster q with the set of primal variables $\mathcal{V}_q = \{\mathbf{w}_{k,q}\}$, Lagrange multipliers $\mathcal{M} = (\{\boldsymbol{\mu}_k^l\}, \{\boldsymbol{\gamma}_k^l\})$, and auxiliary variables $\mathcal{Z} = \{\mathbf{z}_k^l\}$ as

$$\begin{aligned} & \mathcal{L}_{\rho,q}(\mathcal{V}_q, \mathcal{M}, \mathcal{Z}) = \\ & \sum_{k \in \mathcal{C}_q} \left(\frac{1}{D_k} \sum_{i=1}^{D_k} \ell_k(\mathbf{x}_{k,i}, y_{k,i}; \mathbf{w}_{k,q}) + \frac{\lambda}{|\mathcal{C}_q|} R(\mathbf{w}_{k,q}) \right. \\ & \left. + \Big\| \frac{\tau_k}{Q-1} \sum_{r \in \{1,\dots,Q\} \setminus \mathrm{q}} (\hat{\mathbf{w}}_{k,r} - \mathbf{w}_{k,q}) \Big\|^2 \right) \\ & + \sum_{k \in \mathcal{C}_q} \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_q} \Big(\boldsymbol{\mu}_k^{l\mathsf{T}} (\mathbf{w}_{k,q} - \mathbf{z}_k^l) + \boldsymbol{\gamma}_k^{l\mathsf{T}} (\mathbf{w}_{l,q} - \mathbf{z}_k^l) \Big) \\ & + \frac{\rho}{2} \sum_{k \in \mathcal{C}_q} \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_q} \Big(\|\mathbf{w}_{k,q} - \mathbf{z}_k^l\|^2 + \|\mathbf{w}_{l,q} - \mathbf{z}_k^l\|^2 \Big), \end{aligned} \quad (4)$$

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 $\boldsymbol{\gamma}_k = 2 \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_q} \boldsymbol{\gamma}_k^l$, it can be shown that the Lagrange multipliers $\boldsymbol{\mu}_k^l$ and the auxiliary variables $\mathcal{Z}$ are eliminated [25]. From the Lagrangian, the local update steps of the ADMM for a given client k belonging to cluster q can be derived as:

- **Primal update**

$$\begin{aligned} \mathbf{w}_{k,q}^{(n)} = \arg\min_{\mathbf{w}} \; & \frac{1}{D_k} \ell_k(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}) + \frac{\lambda}{|\mathcal{C}_q|} R(\mathbf{w}) \\ & + \Big\| \frac{\tau_k}{Q-1} \sum_{r \in \{1,\dots,Q\} \setminus \mathrm{q}} (\hat{\mathbf{w}}_{k,r}^{(n-1)} - \mathbf{w}) \Big\|^2 \\ & + \mathbf{w}^\mathsf{T} \boldsymbol{\gamma}_{k,q}^{(n-1)} \\ & + \rho \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_q} \Big\| \mathbf{w} - \frac{\mathbf{w}_{k,q}^{(n-1)} + \mathbf{w}_{l,q}^{(n-1)}}{2} \Big\|^2, \end{aligned} \quad (5)$$

- **Dual update**

$$\boldsymbol{\gamma}_{k,q}^{(n)} = \boldsymbol{\gamma}_{k,q}^{(n-1)} + \rho \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_q} (\mathbf{w}_{l,q}^{(n)} - \mathbf{w}_{k,q}^{(n)}), \quad (6)$$

where the superscript (n) denotes the iteration number.

The choice of the inter-cluster learning parameters τ_k will greatly impact the performance of the proposed solution. A common and fixed value would fail to accommodate the heterogeneous data distribution and would not remain relevant throughout the various learning stages of the clients. To address this, we adopt in the next section the principles of reinforcement learning to control time-varying and client-specific inter-cluster learning parameters $\tau_k^{(n)}$.

III. Controlled Inter-Cluster Learning using Reinforcement Learning

Obtaining the optimal values for all $\tau_k^{(n)}$ would require prior knowledge of the network topology and data distribution as well as extensive computational power. Instead, each client k controls the real time evolution of the parameter $\tau_k^{(n)}$ using local data and information received from neighbors. Computationally inexpensive reinforcement learning is used for this purpose.

At any given moment, the state of the reinforcement learning process, at a client k in cluster q, is given by the primal variable, $\mathbf{w}_{k,q}^{(n)}$. The action a corresponds to the modification of the local inter-cluster learning parameter $\tau_k^{(n)}$. We denote $\mathbf{w}_{k,q}^{(n)}(\cdot)$ the function taking a value for the inter-cluster learning parameter and giving the corresponding alternative primal variable. The original primal variable in (5) corresponds to $\mathbf{w}_{k,q}^{(n)} = \mathbf{w}_{k,q}^{(n)}(\tau_k^{(n)})$. For an action a, the corresponding alternative primal variable is given by $\mathbf{w}_{k,q}^{(n)}(a)$.

The state- and action-value functions correspond to the error on the local test dataset of the initial and alternative primal variables, respectively. For instance, the state-value function is given by

$$V_t(\mathbf{w}_{k,q}^{(n)}) = \frac{1}{D_k} \ell_k(\mathbf{X}_{k,t}, \mathbf{y}_{k,t}; \mathbf{w}_{k,q}^{(n)}) + \frac{\lambda}{|\mathcal{C}_q|} R(\mathbf{w}_{k,q}^{(n)}), \quad (7)$$

where $(\mathbf{X}_{k,t}, \mathbf{y}_{k,t})$ denotes the test dataset. To avoid overfitting, it is preferable not to use the test dataset in the reinforcement learning process. Instead, estimates of the state- and action-value functions are computed on a validation dataset $(\mathbf{X}_{k,v}, \mathbf{y}_{k,v})$. The estimate of the state-value function is given by $V_v(\mathbf{w}_{k,q}^{(n)})$, and the estimate of the action-value function by $V_v(\mathbf{w}_{k,q}^{(n)}(a))$.

Policy gradient [26], [27] and deterministic policy gradient [28] are among the most popular policies for continuous action reinforcement learning. They propose a gradient ascent alternative to the greedy maximization of the action-value function given by

$$\tau_k^{(n+1)} = \arg\max_a V_v(\mathbf{w}_{k,q}^{(n)}(a)). \quad (8)$$

In its simplest form, deterministic policy gradient relies on the gradient of the policy reward with respect to the policy parameter at the current state. In the proposed setting, this corresponds to the derivative of $V_v(\mathbf{w}_{k,q}^{(n)}(a))$ with respect to a taken at $\tau_k^{(n)}$, where the sign of the gradient is inverted since the reward corresponds to the error. The policy parameter update is given by

$$\tau_k^{(n+1)} - \tau_k^{(n)} \propto \frac{\partial V_v(\mathbf{w}_{k,q}^{(n)}(a))}{\partial a}, \quad (9)$$

where $\propto$ denotes proportionality. Given the primal update (5), the computation of this derivative is impossible in the general case. However, it can be possible when the loss and regularizer functions are known.

The second proposed policy is a stochastic actor-critic mechanism that takes a random action and compares the action-value function with the state-value function to decide on the next value of the policy parameter. This policy offers better policy parameter exploration than the deterministic policy gradient [29]. First, the policy proposes a random direction $\alpha \sim \mathcal{U}[\frac{-\nu_{\text{SAC}}}{2}, \frac{\nu_{\text{SAC}}}{2}]$ for the policy parameter $\tau_k^{(n)}$ so that $a = \tau_k^{(n)} + \alpha$. $\mathcal{U}(\cdot)$ denotes the uniform distribution, and ν_{SAC} is a hyper-parameter for the policy. The alternative primal variable $\mathbf{w}_{k,q}^{(n)}(a)$ is computed so that the action-value function can be compared with the state-value function. The policy parameter update is given by

$$\tau_k^{(n+1)} - \tau_k^{(n)} \propto -\text{sign}\big(V_v(\mathbf{w}_{k,q}^{(n)}(a)) - V_v(\mathbf{w}_{k,q}^{(n)})\big)\alpha.$$

IV. NPFed-RL for Ridge Regression

As a proof of concept, we use the proposed framework to solve the ridge regression problem. In ridge regression, the loss and regularizer functions used in (2) are given by

$$\ell(\mathbf{X}_k, \mathbf{y}_k, \mathbf{w}) = \|\mathbf{y}_k - \mathbf{X}_k \mathbf{w}\|^2, \qquad R(\mathbf{w}) = \|\mathbf{w}\|^2. \quad (10)$$

The primal variable update for ridge regression is obtained by substituting $\ell(\mathbf{X}_k, \mathbf{y}_k, \mathbf{w})$ and $R(\mathbf{w})$ with their above values in equation (5). Doing so, it is possible to compute the gradient of the term in the $\arg\min$ with respect to the primal variable $\mathbf{w}$, and, setting it to zero, we obtain

$$\begin{aligned}\mathbf{w}_{k,q}^{(n)}(\tau_k^{(n)}) = &\Big(\frac{\mathbf{X}_k^\intercal \mathbf{X}_k}{D_k} + (\frac{\lambda}{|\mathcal{C}_q|} + \tau_k^{(n)} + \rho|\mathcal{N}_k|)\mathbf{I}\Big)^{-1} \\ &\Big(\frac{\mathbf{X}_k^\intercal \mathbf{y}_k}{D_k} \frac{\tau_k^{(n)}}{(Q-1)} \sum_{r \in \{1,\dots,Q\} \setminus \mathrm{q}} \hat{\mathbf{w}}_{k,r}^{(n-1)} \\ &+ \frac{\rho}{2} \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_q} (\mathbf{w}_{k,q}^{(n-1)} + \mathbf{w}_{l,q}^{(n-1)}) - \frac{\boldsymbol{\gamma}_{k,q}^{(n-1)}}{2}\Big).\end{aligned} \quad (11)$$

The alternative primal variable $\mathbf{w}_{k,q}^{(n)}(a)$ for an action a can be computed in the same manner, by replacing $\tau_k^{(n)}$ with a in (11). Using this alternative primal variable and the values of

Algorithm 1 NPFed-RL for ridge regression

Initialization: $\mathbf{w}_{k,q}^{(0)}$ and $\boldsymbol{\gamma}_{k,q}^{(0)}$, $k \in \mathcal{C}$ are set to $\mathbf{0}$, the parameters $\tau_k^{(0)}$ are set to a given value within $(0,1)$.

Procedure at client k:

For $n = 1, 2, \ldots, N$

If $n > 1$

(DPG) τ_k is updated as in (13).

(SAC) τ_k is updated as in (14).

end If

Primal update: $\mathbf{w}_{k,q}^{(n)}$ takes the value in (11).

Client k shares $\mathbf{w}_{k,q}^{(n)}$ with its neighbors in $\mathcal{N}_k$.

Dual update:

$\boldsymbol{\gamma}_{k,q}^{(n)} = \boldsymbol{\gamma}_{k,q}^{(n-1)} + \rho \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_q} (\mathbf{w}_{l,q}^{(n)} - \mathbf{w}_{k,q}^{(n)})$.

the loss and regularizer function for ridge regression in (10), the action-value function can be expressed as

$$V_v(\mathbf{w}_{k,q}^{(n)}(a)) = \frac{1}{D_k} \left\| \mathbf{y}_{k,v} - \mathbf{X}_{k,v} \mathbf{w}_{k,q}^{(n)}(a) \right\|^2 + \frac{\lambda}{|\mathcal{C}_q|} \left\| \mathbf{w}_{k,q}^{(n)}(a) \right\|^2. \quad (12)$$

Using the expression for the primal variable (11) and the action-value function specific to ridge regression in (12), it is possible to compute the derivative of the policy reward with respect to the policy parameter $\partial V_v(\mathbf{w}_{k,q}^{(n)}(\tau_k^{(n)}))/\partial \tau_k^{(n)}$. The explicit derivation is not included because of space constraints. Given that the action-value function is to be minimized, the policy parameter update step for the *deterministic policy gradient*, which we refer to as (DPG), is given by:

$$\tau_k^{(n+1)} = \tau_k^{(n)} - \delta_{\text{DPG}} \frac{\partial V_v(\mathbf{w}_{k,q}^{(n)}(a))}{\partial a}, \quad (13)$$

where δ_{DPG} is the learning rate.

For a random direction α and corresponding action $a = \tau_k^{(n)} + \alpha$, using (12) for the action- and state-value functions, the update step of the policy parameter for the *stochastic actor-critic policy*, which we refer to as (SAC), is given by

$$\tau_k^{(n+1)} = \tau_k^{(n)} - \delta_{\text{SAC}} \text{sign}\big(V_v(\mathbf{w}_{k,q}^{(n)}(a)) - V_v(\mathbf{w}_{k,q}^{(n)})\big)\alpha, \quad (14)$$

where the sign is negated to minimize the action-value function and δ_{SAC} is the learning rate.

The resulting algorithms, referred to as Networked Personalized Federated learning using Reinforcement Learning (NPFed-RL) are summarized in Algorithm 1.

V. Numerical Simulations

We considered a distributed network composed of $|\mathcal{C}| = 30$ clients with an average of 6 neighbors per client. The clients are randomly grouped into $Q = 3$ clusters. The goal is to estimate cluster-specific tasks given by $\mathbf{w}_q = \mathbf{w}_0 + \delta_q \mathbf{w}_0$, with $\delta_q \sim \mathcal{U}(-0.5, 0.5)$. $\mathcal{U}$ denotes the uniform distribution, and $\mathbf{w}_0$ is a randomly chosen base model. Each client k possesses a training dataset $(\mathbf{X}_k, \mathbf{y}_k)$ where $\mathbf{X}_k \in \mathbb{R}^{D_k \times 60}$ and $\mathbf{y}_k \in \mathbb{R}^{D_k \times 1}$ with $D_k \sim \mathcal{U}(5, 35)$, as well as identically distributed testing and validation datasets. The data is generated as $\mathbf{y}_k = \mathbf{X}_k \mathbf{w}_q + \mathbf{n}_k$, with $\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \eta_k)$, where η_k is the client-specific noise variance and $\mathbf{w}_q$ is its cluster model. Finally, the Lagrangian penalty parameter is set to $\rho = 3$. We considered the mean squared error on the testing data set (Test MSE) as the performance metric for comparison of the algorithms. It is given by

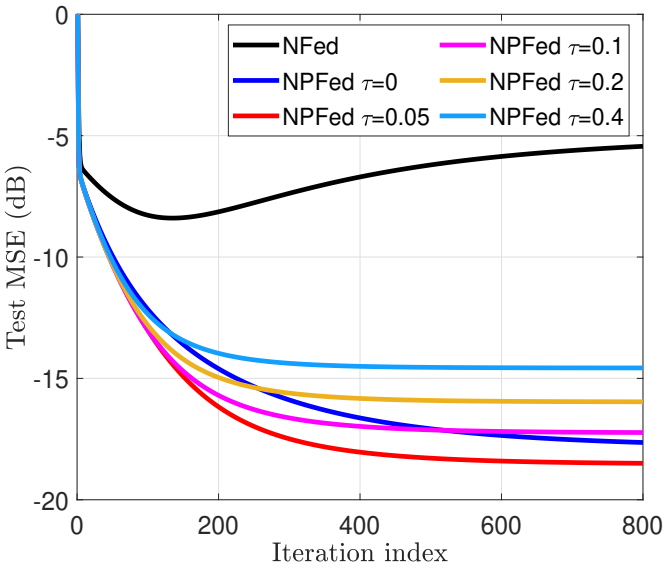

Fig. 1: Learning curves of the NFed and NPFed algorithms for various values of τ.

$$\text{Test MSE} = \frac{1}{|\mathcal{C}|} \sum_{k=1}^{|\mathcal{C}|} \frac{\|\mathbf{w}_{k,q} - \bar{\mathbf{w}}_q\|_2^2}{\|\bar{\mathbf{w}}_q\|}, \quad (15)$$

where $\{\mathbf{w}_{k,q}, k \in \mathcal{C}_q, q \in \{1, \ldots, Q\}\}$ are the models of the considered method and $\bar{\mathbf{w}}_q$ is the optimal model for the cluster. The simulation results presented in the following are obtained by averaging the results of 10 independent experiments. To ensure a fair comparison, the algorithms are tuned to have the same initial convergence rate in Fig. (2).

The first experiment studied the impact of the inter-cluster learning parameter τ on the learning behavior of conventional networked FL algorithms. For this purpose, we simulated the following algorithms:

- **NFed:** is the traditional networked FL that learns one universal model for the whole network.
- **NPFed:** is conventional personalized networked FL that learns cluster-specific models. The global parameter τ is fixed throughout the learning process, so that $\forall k \in \mathcal{C}, n > 0; \tau_k = \tau$. Inter-cluster learning is absent when $\tau = 0$.

The learning curves (i.e., Test MSE in dB vs. iteration index n) of the above algorithms are presented in Fig. 1. The figure shows that the NFed algorithm does not achieve satisfactory

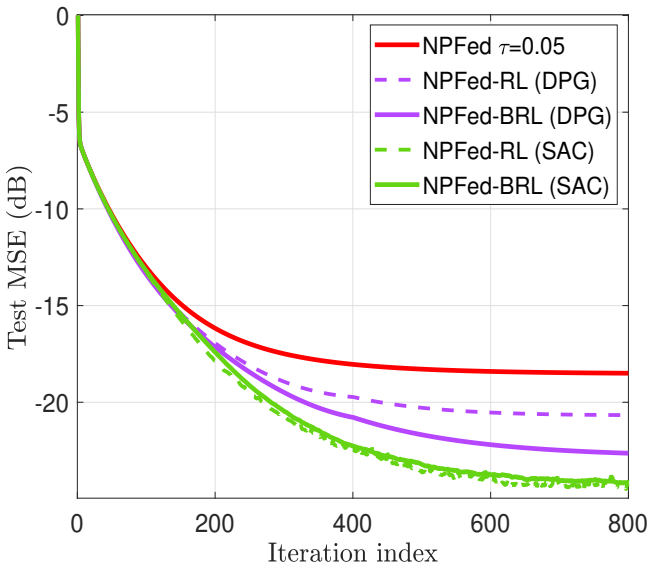

Fig. 2: Learning curves of the NPFed-RL and NPFed-BRL with different policies. Also plotted is the learning curve of NPFed for $\tau = 0.05$.

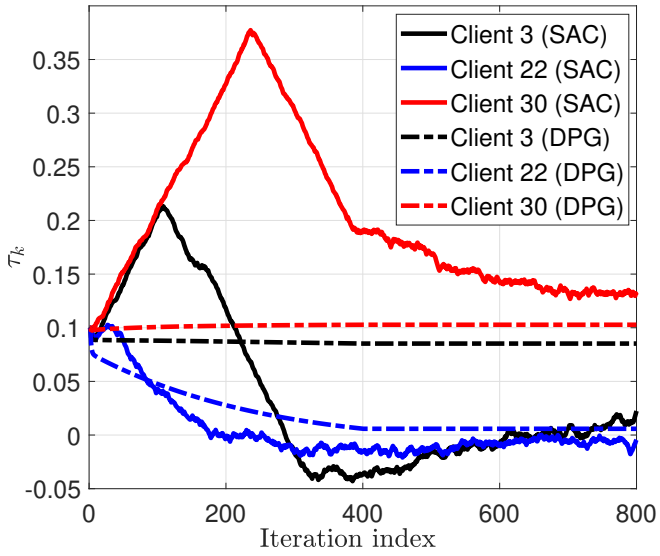

Fig. 3: Evolution of $\tau_k^{(n)}$ for the 3rd, 22nd, and 30th clients.

accuracy since it tries to learn a single universal model that cannot accommodate client-specific tasks. NPFed with $\tau = 0$ implies that each cluster independently builds its own model by relying solely on the cooperation among cluster members. Its performance can, therefore, be regarded as a benchmark for networked personalized federated learning. As the value of τ increases (e.g., $\tau = 0.05$), the NPFed performance improves, as it enforces similarity between the cluster-specific models. However, as more similarity is enforced between models for non-identical tasks, the steady-state accuracy decreases, as can be seen with NPFed $\tau = 0.1$, $\tau = 0.2$, and $\tau = 0.4$. This confirms that inter-cluster learning can be beneficial but leads to performance degradation when over-used.

In the second experiment, we demonstrated the effectiveness of the proposed NPFed-RL in the learning of personalized models. For this purpose, we simulated the following algorithms:

- **NPFed-RL (DPG):** is the proposed NPFed-RL using the deterministic policy gradient method. The policy hyper-parameter was set to $\delta_{\text{DPG}} = 0.001$, and its policy parameter update step is given in (13).
- **NPFed-BRL (DPG):** is the NPFed-RL using the policy gradient and batch reinforcement learning [30] with 3 epochs and $\delta_{\text{DPG}} = 0.003$.
- **NPFed-RL (SAC):** is the proposed NPFed-RL using the stochastic actor-critic policy. The policy hyper-parameters were set to $\nu_{\text{SAC}} = 0.05$ and $\delta_{\text{SAC}} = 0.1$. The policy parameter update step is given in (14).
- **NPFed-BRL (SAC):** is the NPFed-RL using the stochastic actor-critic policy and batch reinforcement learning [30] with 3 epochs, $\nu_{\text{SAC}} = 0.05$ and $\delta_{\text{SAC}} = 0.04$.

The learning curves of these algorithms are presented in Fig. 2. For comparison purposes, the learning curve of NPFed with $\tau = 0.05$ is also displayed. We see that all the versions of NPFed-RL exhibit better performance in initial learning speed and steady-state accuracy compared to NPFed operating with a fixed τ value. Since the clients have device-specific requirements, usage of a fixed universal τ is impractical. Whereas the proposed NFed-RL controls the amount of inter-cluster learning locally by learning the device-specific parameters $\tau_k^{(n)}$ in real time. Further, we also see that NPFed-RL (SAC) exhibites enhanced accuracy over NPFed-RL (DPG). The reason for this is that the stochastic nature of NPFed-RL (SAC) ensures sufficient exploration of the policy parameters τ_k, which is not the case for NPFed-RL (DPG). This stochastic nature also leads to extensive randomness in the convergence of NPFed-RL (SAC), batch reinforcement learning attenuates this issue as can be seen with NPFed-BRL (SAC). In the case of deterministic policy gradient, batch reinforcement learning increases the learning accuracy. It is important to note that batch reinforcement learning comes with a computational cost proportional to the number of epochs performed.

Finally, we illustrate the evolution of the inter/cluster learning parameters $\tau_k^{(n)}$ for NPFed-RL (SAC) and NPFed-RL (DPG) in Fig. 3. We selected three clients 22, 3, 30, having access to large, moderate, and small amounts of data, respectively. Therefore, these clients have different local requirements for inter-cluster learning. From Fig. 3, we observe that the evolution of $\tau_k^{(n)}$ is very smooth when using NPFed-RL (DPG) but fails to quickly adapt. On the other hand, the NPFed-RL (SAC) algorithm provides sufficient exploration of the policy parameter, ensuring that the $\tau_k^{(n)}$ parameters evolve quickly at the cost of extensive randomness (BRL helps to overcome this issue). Furthermore, we also see that $\tau_k^{(n)}$ evolves according to the needs of clients. Since client 30 has access to a small amount of data, $\tau_{30}^{(n)}$ increases linearly at first to enforce higher inter-cluster learning. After reaching near-convergence, the $\tau_{30}^{(n)}$ value decreases to reduce the amount of

inter-cluster learning to avoid its harmful effect. In contrast, since client 22 has access to a large amount of data, the $\tau_{22}^{(n)}$ parameter decreases almost immediately and stabilizes around 0 as inter-cluster learning is not valuable for this client. Finally, client 3 has access to an average amount of data. $\tau_{3}^{(n)}$ increases at first as similarity enforcement allows for faster initial convergence, but decreases afterwards and stabilizes around 0 to avoid performance degradation.

VI. Conclusions

Personalized federated learning suffers from data scarcity within clusters, this is alleviated by leveraging the similarity between the learning tasks. However, how much a specific client needs to rely on inter-cluster learning depends on its local data and the network topology. To accommodate the varied needs of the clients, we propose a networked personalized federated learning algorithm using reinforcement learning to control evolving client-specific inter-cluster learning parameters. Each local parameter is updated on-the-fly by the reinforcement learning process in a computationally inexpensive manner so that model similarity is enforced only as much as what is beneficial for local learning. Numerical simulations show that the proposed method has better learning performances than the state-of-the-art.

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 Proc. Mag.*, vol. 37, no. 3, pp. 50–60, May. 2020.

[3] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[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] 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.

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

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

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

[10] Y. Sarcheshmehpour, M. Leinonen, and A. Jung, "Federated learning from big data over networks," *in Proc. IEEE Int. Conf. Acoust., Speech and Signal Process.*, pp. 3055–3059, Jan. 2021.

[11] 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.

[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] S. Boll and J. Meyer, "Health-X dataLOFT: A Sovereign Federated Cloud for Personalized Health Care Services," *IEEE MultiMedia*, vol. 29, no. 1, pp. 136–140, May 2022.

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

[15] 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.

[16] 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.

[17] V. C. Gogineni and M. Chakraborty, "Diffusion affine projection algorithm for multitask networks," *in Proc. Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf.*, pp. 201–206, Nov. 2018.

[18] ——, "Improving the performance of multitask diffusion APA via controlled inter-cluster cooperation," *IEEE Trans. Circuits Sys. I: Reg. Papers*, vol. 67, no. 3, pp. 903–912, Dec. 2019.

[19] W. Lei, Y. Ye, M. Xiao, M. Skoglund, and Z. Han, "Adaptive stochastic ADMM for decentralized reinforcement learning in edge IoT," *IEEE Internet Things J.*, Jun. 2022.

[20] M. Krouka, A. Elgabli, C. B. Issaid, and M. Bennis, "Communication-efficient and federated multi-agent reinforcement learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 1, pp. 311–320, Nov. 2021.

[21] S. El Bsat, H. B. Ammar, and M. E. Taylor, "Scalable multitask policy gradient reinforcement learning," *in Proc. Thirty-first AAAI Conf. Artif. Intell.*, Feb. 2017.

[22] R. Tutunov, D. Kim, and H. Bou Ammar, "Distributed multitask reinforcement learning with quadratic convergence," *Advances Neural Inf. Process. Syst.*, vol. 31, 2018.

[23] Y. Ye, M. Xiao, and M. Skoglund, "Randomized neural networks based decentralized multi-task learning via hybrid multi-block ADMM," *IEEE Trans. Signal Process.*, vol. 69, pp. 2844–2857, May 2021.

[24] G. Karafotias, A. E. Eiben, and M. Hoogendoorn, "Generic parameter control with reinforcement learning," *in Proc. Annu. Conf. Genetic Evol. Comput.*, pp. 1319–1326, Jul. 2014.

[25] 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.

[26] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances Neural Inf. Process. Syst.*, vol. 12, 1999.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, Jul. 2017.

[28] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," *in Proc. Int. Conf. Mach. Learn.*, pp. 387–395, Jan. 2014.

[29] H. Wang, T. Zariphopoulou, and X. Y. Zhou, "Reinforcement Learning in Continuous Time and Space: A Stochastic Control Approach," *J. Mach. Learn. Res.*, vol. 21, no. 198, pp. 1–34, Jan. 2020.

[30] S. Lange, T. Gabel, and M. Riedmiller, "Batch reinforcement learning," *Reinforcement Learning*, pp. 45–73, Springer, 2012.