Cooperative Personalized Bilevel Optimization over Random Directed Networks

Naoyuki Terashita
Hitachi, Ltd.
Tokyo, Japan
naoyuki.terashita.sk@hitachi.com

Satoshi Hara
Osaka University
Osaka, Japan
satohara@ar.sanken.osaka-u.ac.jp

ABSTRACT

This paper aims to enhance the predictive performance and communication robustness of decentralized bilevel optimization for personalized federated learning. To achieve better performance, we introduce a bilevel problem that encourages cooperation among agents; the goal of every agent is to minimize the total cost over all agents by updating its personalized outer-parameter. For robust communication, we solve the introduced problem over random directed networks. Our solver utilizes a decentralized computation algorithm for the gradient of the outer-parameter that runs over directed networks. Empirical results demonstrate that our approach outperforms baselines on personalization benchmarks and that it functions over simulated random directed networks.

KEYWORDS

Bilevel optimization, Hyper-gradient, Personalization, Federated learning, Distributed learning, Decentralized machine learning

1 INTRODUCTION

Recent studies have shown the effectiveness of *decentralized personalization* for federated learning (FL). Traditional FL [14] trains a globally shared, or *consensus*, model parameter on a central server. This scheme has two problems. First, studies have suggested that a consensus model may not perform well for situations in which the agents' data distributions are heterogeneous [19]. Furthermore, having a central server can lead to central point failure and bandwidth bottlenecks [1]. Decentralized personalization [13, 22, 25] is a promising approach to address both issues. In decentralized personalization, data heterogeneity is handled by training personally-tuned models for agents, and decentralized communication allows for peer-to-peer cooperation over communication, avoiding the previously mentioned practical issues of central server settings.

Recent studies [10, 11] demonstrated the effectiveness of gradient-based *bilevel optimization* in decentralized personalization. Bilevel optimization refers to an optimization problem (*outer-problem*) that embeds another optimization problem (*inner-problem*) as its constraint. Specifically, gradient-based bilevel optimization is considered to be effective when there are many parameters involved [4]. Prior works [10, 11] leverage the ability to handle many parameters in their personalization schemes that partition parameters of a deep neural network into consensus outer-parameters and personalized inner-parameters.

Current bottlenecks for improving personalization with decentralized bilevel optimization are the absence of cooperation among agents and the limited robustness of the available communication networks. Previous works [10, 11] let each agent perform a local gradient step to optimize its personalized parameter in the inner-problem. However, the local gradient of inner-cost ignores how perturbations to the parameter of each agent affect the performance of the others, lacking efficient cooperation among agents. To encourage cooperative optimization, it is preferable to allow agents to update their parameters such that the costs of other agents are minimized as well. While such cooperative updates are considered in another type of decentralized bilevel optimization [24], their algorithm is available only on *undirected networks*, which can be prone to the presence of failing agents and deadlocks [21].

Our contributions. This study proposes a decentralized bilevel optimization method for personalization that enables agent cooperation for enhanced performance, and that can run over a more robust communication scheme, namely, a *random directed network*.

Firstly, we formulate our problem setting called *Personalized Decentralized Bilevel Optimization* (*PDBO*) which optimizes an agent-wise personalized parameter as the outer-problem requiring agent cooperation; every agent optimizes its outer-parameter to minimize the total outer-cost over all agents.

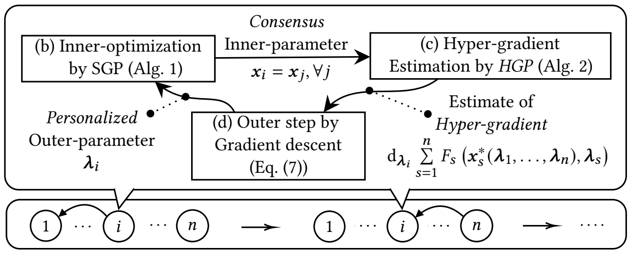
Secondly, to solve PDBO in a more robust communication setting, we investigate the decentralized computation algorithm for the gradient of the outer-parameter (*hyper-gradient*) that runs over random directed networks. Directed networks are known to be less prone to issues that arise in undirected networks [21]. We initially reveal that naively decentralizing a previous hyper-gradient computation algorithm fails due to the directionality of communication. Motivated by this negative result, we present *Hyper-gradient Push* (*HGP*), which is a modified yet unbiased version of the previous method that runs over directed networks.

Finally, we provide empirical evidence of the benefits of PDBO solved with HGP by showing superior performance on personalized FL benchmarks conducted on directed communications networks.

Paper Outline. The rest of the paper is organized as follows. Section 2 introduces the formulation of a random directed network (Fig. 1-a) on which agents communicate, as well as the stochastic gradient push (SGP) [1, 15] which solves the inner-problem (Fig. 1-b). Section 3 formulates PDBO and Section 4 proposes HGP that estimates hyper-gradient (Fig. 1-c) required to perform the outer-step of PDBO (Fig. 1-d). Finally, we present the experimental results in Section 5, followed by the conclusion in Section 6.

Notation. $\langle \boldsymbol{A} \rangle_{ij}$ denotes the i-th row and j-th column block of the matrix $\boldsymbol{A}$ and $\langle \boldsymbol{a} \rangle_i$ denotes the i-th block vector of the vector $\boldsymbol{a}$. For a function $f : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$, we denote its total and partial derivatives with respect to a vector $\boldsymbol{x} \in \mathbb{R}^{d_1}$ by $\mathrm{d}_{\boldsymbol{x}} f(\boldsymbol{x}) \in \mathbb{R}^{d_1 \times d_2}$

Appears in the 14th Workshop on Optimization and Learning in Multiagent Systems (OptLearnMAS-23), May 29, 2023, London, United Kingdom.


Consensus
Inner-parameter
$x_i = x_j, \forall j$
(b) Inner-optimization by SGP (Alg. 1)
(c) Hyper-gradient Estimation by HGP (Alg. 2)
Personalized Outer-parameter λ_i
(d) Outer step by Gradient descent (Eq. (7))
Estimate of Hyper-gradient
$d_{\lambda_i} \sum_{s=1}^{n} F_s(x_s^*(\lambda_1,\ldots,\lambda_n),\lambda_s)$
1 ··· i ··· n
1 ··· i ··· n
····

(a) *Random Directed Network* (Section 2.1)

Figure 1: Overview of PDBO solved by HGP.

and $\partial_{\boldsymbol{x}} f(\boldsymbol{x}) \in \mathbb{R}^{d_1 \times d_2}$, respectively. Let $[n] = \{1, \ldots, n\}$, we use operations $\{x_i\}_{i\in[n]} = \{x_i \mid i \in [n]\}$ for some values x_i and $[\boldsymbol{x}_i]_{i=1}^n = [\boldsymbol{x}_1^\top \cdots \boldsymbol{x}_n^\top]^\top \in \mathbb{R}^{nd}$ for vectors $\boldsymbol{x}_i \in \mathbb{R}^d$.

2 PRELIMINARIES

This section provides the background of our study. Section 2.1 introduces a model of the random directed network and Section 2.2 provides a formulation of SGP that solves the FL over random directed networks.

2.1 Random Directed Communication Network

We model the random directed network following the previous work of push-sum [17]. We consider a random directed network consisting of n agents and communication edges that randomly realize at each time step. Let $\delta_{i\to j} \in \{0, 1\}$ be a random variable where $\delta_{i\to j} = 1$ denotes that there is a communication channel from the i-th agent to the j-th agent at a single time step, and $\delta_{i\to j} = 0$ otherwise. $\delta_{i\to j}$ are mutually independent for any (i, j) such that $i \neq j$, and $\delta_{i\to i} = 1$ for every $i \in [n]$. We assume if $\overline{\delta}_{j\to i} > 0$, then $\overline{\delta}_{i\to j} > 0$ and vice versa, implying the agents can communicate with each other in a sufficiently long time interval. We denote the set of sending edges of the i-th agent by $\delta_i = \{\delta_{i\to j}\}_{j\in[n]}$ and all the edges in the network by $\delta = \{\delta_i\}_{i\in[n]}$. We assume that realizations of δ are mutually independent between any different time steps.

2.2 Stochastic Gradient Push (SGP)

SGP [1, 15] is one of the most general decentralized solvers of the following FL problem [14]:

$$\boldsymbol{x}_i^\dagger = \mathop{\arg\min}_{\boldsymbol{x}_i;\, \boldsymbol{x}_i = \boldsymbol{x}_j, \forall j} \sum_{k\in[n]} \mathbb{E}_{\xi_k}\left[f_k(\boldsymbol{x}_k, \boldsymbol{\lambda}_k; \xi_k)\right],\ \forall i \in [n] \tag{1}$$

where, the i-th agent pursues the optimal parameter $\boldsymbol{x}_i^\dagger \in \mathbb{R}^{d_x}$, that makes *consensus* ($\boldsymbol{x}_i = \boldsymbol{x}_j, \forall j$) over all the agents, while minimizing its cost $f_i : \mathbb{R}^{d_x} \times \mathbb{R}^{d_\lambda} \to \mathbb{R}$. f_i is parameterized by the local instance $\xi_i \in \mathcal{X}$ (e.g., a tuple of inputs and their labels), which is a random variable following its own local distribution. We also allow f_i to take outer-parameter $\boldsymbol{\lambda}_i \in \mathbb{R}^{d_\lambda}$ as its argument.

Any i-th agent in SGP seeks $\boldsymbol{x}_i^\dagger$ by updating biased parameter $\boldsymbol{z}_i \in \mathbb{R}^{d_x}$ and debias weight $w_i \in \mathbb{R}$ in parallel to obtain its debiased parameter $\boldsymbol{x}_i = \boldsymbol{z}_i / w_i$. Let $\boldsymbol{\varphi}_i : \mathbb{R}^{d_x} \times \mathbb{R}^{d_\lambda} \to \mathbb{R}^{d_x}$ denote a local update function defined as

$$\boldsymbol{\varphi}_i(\boldsymbol{z}_i, \boldsymbol{\lambda}_i; \xi_i, w_i) = \boldsymbol{z}_i - \gamma_z \partial_{\frac{\boldsymbol{z}_i}{w_i}} f_i(\boldsymbol{z}_i / w_i, \boldsymbol{\lambda}_i; \xi_i), \tag{2}$$

where, $\gamma_z \in \mathbb{R}^+$ is the learning rate. We denote the independent copies of ξ_i and δ_i sampled at the t-th step as $\xi_i^{(t)}$ and $\delta_i^{(t)}$, respectively, and denote a weighted edge from j to i by

$$p_{j\to i}(\delta_j) = \frac{\delta_{j\to i}}{\sum_{k=1}^n \delta_{j\to k}}. \tag{3}$$

We then obtain the following SGP algorithm run by the i-th agent.

Algorithm 1: Stochastic Gradient Push [1]

```
1 $w_i^{(0)} \leftarrow 1,\quad \boldsymbol{z}_i^{(0)} \leftarrow \boldsymbol{x}_i^{(0)}$
2 foreach $t = 0$ to $T-1$ do
3     $\boldsymbol{z}_i^{(t+1)} \leftarrow \sum_{j=1}^n p_{j\to i}(\delta_j^{(t)}) \boldsymbol{\varphi}_j(\boldsymbol{z}_j^{(t)}, \boldsymbol{\lambda}_j; \xi_j^{(t)}, w_j^{(t)})$
4     $w_i^{(t+1)} \leftarrow \sum_{j=1}^n p_{j\to i}(\delta_j^{(t)}) w_j^{(t)}$
5 return $\boldsymbol{x}_i^{(T)} = \boldsymbol{z}_i^{(T)} / w_i^{(T)}$
```

3 PERSONALIZED DECENTRALIZED BILEVEL OPTIMIZATION (PDBO)

This section introduces PDBO, the personalization problem formulated as a bilevel problem. PDBO played by n agents is given as

$$\min_{\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_n} \sum_{s\in[n]} F_s(\boldsymbol{x}_s^\dagger(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_n), \boldsymbol{\lambda}_s), \tag{4a}$$

$$\text{s.t. } \boldsymbol{x}_i^\dagger \text{ satisfies Eq. (1)},\ \forall i \in [n], \tag{4b}$$

where the outer-problem (Eq. (4a)) lets the i-th agent to optimize its outer-parameter $\boldsymbol{\lambda}_i$ to minimize the sum of the outer-cost $F_i : \mathbb{R}^{d_x} \times \mathbb{R}^{d_\lambda} \to \mathbb{R}$ across all agents, encouraging agent cooperation. Note that outer-parameters are free from the consensus, allowing agents to optimize their own parameters for personalization.

Reformulating the Inner-problem via Stationary Point. To solve PDBO by gradient methods, we rewrite the inner-problem (Eq. (4b)) to a stationary point equation of the inner-iteration [6].

We consider the stationary point of an iteration of SGP with respect to a concatenated biased parameter $\boldsymbol{z} = [\boldsymbol{z}_i]_{i=1}^n$. Let $\xi = \{\xi_i\}_{i\in[n]}$ be a set of instances of all agents, $\zeta = (\delta, \xi)$ be all randomness of the agents, and $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_i]_{i=1}^n$ be a concatenated outer-parameter. Recalling Alg. 1, we define a concatenated version of the SGP iteration $\boldsymbol{\psi}(\boldsymbol{z}, \boldsymbol{\lambda}; \zeta)$ as

$$\langle \boldsymbol{\psi}(\boldsymbol{z}, \boldsymbol{\lambda}; \zeta)\rangle_i = \sum_{j\in[n]} p_{j\to i}(\delta_j) \boldsymbol{\varphi}_j(\boldsymbol{z}_j, \boldsymbol{\lambda}_j; \xi_j, w_j^*),\ \forall i \in [n],$$

where, w_j^* is the j-th element of the stationary point $\boldsymbol{w}^* = \mathbb{E}[\boldsymbol{P}(\delta)]\boldsymbol{w}^*$. We finally introduce an assumption on $\boldsymbol{\psi}$.

Assumption 3.1. [1] $\mathbb{E}_\zeta[\boldsymbol{\psi}(\cdot, \boldsymbol{\lambda}; \zeta)]$ is a contraction $\forall \boldsymbol{\lambda} \in \mathbb{R}^{n d_\lambda}$.

Assumption 3.1 admits the following unique stationary point:

$$\boldsymbol{z}^*(\boldsymbol{\lambda}) = \mathbb{E}_\zeta[\boldsymbol{\psi}(\boldsymbol{z}^*(\boldsymbol{\lambda}), \boldsymbol{\lambda}; \zeta)]. \tag{5}$$

Letting $\boldsymbol{x}^*(\boldsymbol{z}) := [\boldsymbol{z}_i / w_i^*]_{i=1}^n$, we reformulate Eq. (4) as

$$\min_{\boldsymbol{\lambda}} \overline{F}(\boldsymbol{x}^*(\boldsymbol{z}^*(\boldsymbol{\lambda})), \boldsymbol{\lambda}), \quad \text{s.t. } \boldsymbol{z}^*(\boldsymbol{\lambda}) \text{ satisfies Eq. (5)}, \tag{6}$$

[1]We found that it is difficult to provide practical scenarios in which such an assumption holds true. Therefore, in Terashita and Hara [20], we propose a different hyper-gradient estimation method derived from the stationary point given by optimality condition in Eq. (1) rather than assuming Assumption 3.1.

where, $\overline{F}(\boldsymbol{x}, \boldsymbol{\lambda}) := \sum_{s=1}^{n} F_s(\boldsymbol{x}_s, \boldsymbol{\lambda}_s)$.

4 HYPER-GRADIENT ESTIMATION OVER RANDOM DIRECTED NETWORKS

To solve Eq. (6), agents need to compute the hyper-gradient $\mathrm{d}_{\boldsymbol{\lambda}}\overline{F} := \mathrm{d}_{\boldsymbol{\lambda}}\overline{F}(\boldsymbol{x}^*(\boldsymbol{z}^*(\boldsymbol{\lambda})), \boldsymbol{\lambda})$ using gradient descent. Our interest is to allow any i-th agent to perform the following local outer-step using the i-th block of the concatenated hyper-gradient $\mathrm{d}_{\boldsymbol{\lambda}_i}\overline{F} = \langle \mathrm{d}_{\boldsymbol{\lambda}}\overline{F} \rangle_i \in \mathbb{R}^{d_\lambda}$:

$$\boldsymbol{\lambda}_i \leftarrow \boldsymbol{\lambda}_i - \gamma_\lambda \mathrm{d}_{\boldsymbol{\lambda}_i}\overline{F}, \tag{7}$$

where, $\gamma_\lambda \in \mathbb{R}^+$ is the outer learning rate.

To this end, this section presents an empirical estimator of the hyper-gradient and its decentralized computation algorithm. Section 4.1 introduces the empirical estimator *Stochastic Recurrent Backpropagation (SRB)*, following previous hyper-gradient computation methods. Section 4.2 then provides our observation that a straightforward decentralization of SRB fails on directed networks. Finally, Section 4.3 proposes HGP, a decentralized algorithm that modifies SRB to run on random directed communication.

4.1 Empirical Estimation of the Hyper-gradient

In this section, we present the estimator of the hyper-gradient $\mathrm{d}_{\boldsymbol{\lambda}}\overline{F}$ by using approximate implicit differentiation and stochastic iteration techniques [5, 6].

Approximate Implicit Differentiation. Let $\partial_{\boldsymbol{z}}^*\boldsymbol{\psi}(\zeta)$ and $\partial_{\boldsymbol{\lambda}}^*\boldsymbol{\psi}(\zeta)$ be Jacobians of $\boldsymbol{\psi}(\boldsymbol{z}^*(\boldsymbol{\lambda}), \boldsymbol{\lambda}; \zeta)$ with respect to $\boldsymbol{z}^*(\boldsymbol{\lambda})$ and $\boldsymbol{\lambda}$, respectively. Assumption 3.1 ensures that $\boldsymbol{I} - \mathbb{E}[\partial_{\boldsymbol{z}}^*\boldsymbol{\psi}(\zeta)]$ is invertible and allows the Neumann series approximation $(\boldsymbol{I} - \mathbb{E}[\partial_{\boldsymbol{z}}^*\boldsymbol{\psi}(\zeta)])^{-1} \approx \sum_{m=0}^{M-1} \mathbb{E}[\partial_{\boldsymbol{z}}^*\boldsymbol{\psi}(\zeta)]^m$. Using these facts and implicit differentiation of Eq. (5), we obtain

$$\mathrm{d}_{\boldsymbol{\lambda}}\overline{F} \approx \mathbb{E}[\partial_{\boldsymbol{\lambda}}^*\boldsymbol{\psi}(\zeta)] \sum_{m=0}^{M-1} \mathbb{E}[\partial_{\boldsymbol{z}}^*\boldsymbol{\psi}(\zeta)]^m \partial_{\boldsymbol{z}}^*\overline{F} + \partial_{\boldsymbol{\lambda}}^*\overline{F}, \tag{8}$$

where $\partial_{\boldsymbol{z}}^*\overline{F}$ and $\partial_{\boldsymbol{\lambda}}^*\overline{F}$ are gradients of $\overline{F}(\boldsymbol{x}^*(\boldsymbol{z}^*(\boldsymbol{\lambda})), \boldsymbol{\lambda})$ with respect to $\boldsymbol{z}^*(\boldsymbol{\lambda})$ and $\boldsymbol{\lambda}$, respectively.

Stochastic Recurrent Backpropagation (SRB). To avoid computation of full Jacobians in Eq. (8) and to estimate their expectation, we follow the stochastic iteration technique [5].

Let $\breve{\zeta}^{(m)}$ and $\widehat{\zeta}^{(m)}$ be independent copies of ζ for every $m \in \mathbb{N}$. By replacing the expected Jacobians in Eq. (8) with their estimate using $\breve{\zeta}^{(m)}$ and $\widehat{\zeta}^{(m)}$, we obtain the SRB that gives the unbiased estimation of Eq. (8) denoted by $\widehat{\mathrm{d}_{\boldsymbol{\lambda}}\overline{F}}$.

$$\boldsymbol{u}^{(0)} \leftarrow \partial_{\boldsymbol{z}}^*\overline{F}, \quad \boldsymbol{v}^{(0)} \leftarrow \partial_{\boldsymbol{\lambda}}^*\overline{F} \tag{9a}$$

for $m = 0, \ldots, M-1$

$$\left| \begin{cases} \boldsymbol{v}^{(m+1)} \leftarrow \partial_{\boldsymbol{\lambda}}^*\boldsymbol{\psi}(\breve{\zeta}^{(m)})\boldsymbol{u}^{(m)} + \boldsymbol{v}^{(m)}, \\ \boldsymbol{u}^{(m+1)} \leftarrow \partial_{\boldsymbol{z}}^*\boldsymbol{\psi}(\widehat{\zeta}^{(m)})\boldsymbol{u}^{(m)} \end{cases} \right. \tag{9b}$$

$$\widehat{\mathrm{d}_{\boldsymbol{\lambda}}\overline{F}} \leftarrow \boldsymbol{v}^{(M)}$$

Eq. (9b) only requires Jacobian-vector products $\partial_{\boldsymbol{\lambda}}^*\boldsymbol{\psi}(\zeta)\boldsymbol{u}^{(m)}$ and $\partial_{\boldsymbol{z}}^*\boldsymbol{\psi}(\zeta)\boldsymbol{u}^{(m)}$ leading $O(nd_{\boldsymbol{x}} + nd_{\boldsymbol{\lambda}})$ and $O(nd_{\boldsymbol{x}})$ in time, respectively.

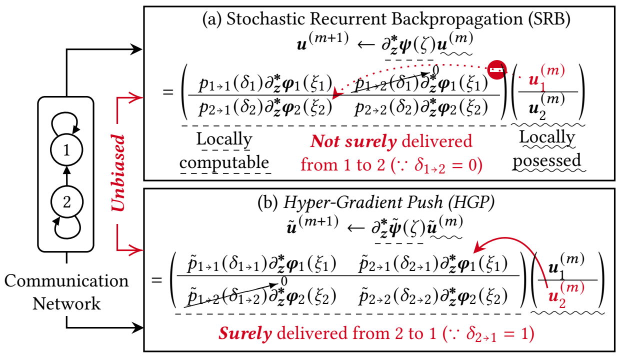

(a) Stochastic Recurrent Backpropagation (SRB)
$\boldsymbol{u}^{(m+1)} \leftarrow \partial_{\boldsymbol{z}}^*\boldsymbol{\psi}(\zeta)\boldsymbol{u}^{(m)}$
$= \begin{pmatrix} p_{1\to1}(\delta_1)\partial_{\boldsymbol{z}}^*\boldsymbol{\varphi}_1(\xi_1) & \cancelto{0}{p_{1\to2}(\delta_1)}\partial_{\boldsymbol{z}}^*\boldsymbol{\varphi}_1(\xi_1) \\ p_{2\to1}(\delta_2)\partial_{\boldsymbol{z}}^*\boldsymbol{\varphi}_2(\xi_2) & p_{2\to2}(\delta_2)\partial_{\boldsymbol{z}}^*\boldsymbol{\varphi}_2(\xi_2) \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1^{(m)} \\ \boldsymbol{u}_2^{(m)} \end{pmatrix}$
Locally computable
Not surely delivered from 1 to 2 ($\because \delta_{1\to2} = 0$)
Locally posessed
Unbiased
1
2
Communication Network
(b) *Hyper-Gradient Push (HGP)*
$\tilde{\boldsymbol{u}}^{(m+1)} \leftarrow \partial_{\boldsymbol{z}}^*\tilde{\boldsymbol{\psi}}(\zeta)\tilde{\boldsymbol{u}}^{(m)}$
$= \begin{pmatrix} \tilde{p}_{1\to1}(\delta_{1\to1})\partial_{\boldsymbol{z}}^*\boldsymbol{\varphi}_1(\xi_1) & \tilde{p}_{2\to1}(\delta_{2\to1})\partial_{\boldsymbol{z}}^*\boldsymbol{\varphi}_1(\xi_1) \\ \cancelto{0}{\tilde{p}_{1\to2}(\delta_{1\to2})}\partial_{\boldsymbol{z}}^*\boldsymbol{\varphi}_2(\xi_2) & \tilde{p}_{2\to2}(\delta_{2\to2})\partial_{\boldsymbol{z}}^*\boldsymbol{\varphi}_2(\xi_2) \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1^{(m)} \\ \boldsymbol{u}_2^{(m)} \end{pmatrix}$
Surely delivered from 2 to 1 ($\because \delta_{2\to1} = 1$)

Figure 2: An illustration of computation of $\boldsymbol{u}^{(m+1)}$ and $\widetilde{\boldsymbol{u}}^{(m+1)}$ by SRB (a) and HGP (b), respectively, with $n = 2$.

4.2 SRB over Random Directed Networks

This section presents an impossibility result of decentralizing SRB due to the locality of Jacobian and gradient blocks, and the directionality of communication.

Locality of Jacobian and Gradient Blocks. Due to the locality of instances ξ_i and communication edges δ_i, any i-th agent can locally compute only specific blocks of $\partial_{\boldsymbol{z}}^*\overline{F}$, $\partial_{\boldsymbol{\lambda}}^*\overline{F}$, $\partial_{\boldsymbol{z}}^*\boldsymbol{\psi}(\zeta)$, and $\partial_{\boldsymbol{\lambda}}^*\boldsymbol{\psi}(\zeta)$ appearing in Eq. (9), that are

$$\begin{cases} \langle \partial_{\boldsymbol{z}}^*\overline{F} \rangle_i = \partial_{\boldsymbol{z}_i} F_i(\boldsymbol{z}_i^*(\boldsymbol{\lambda})/w_i^*, \boldsymbol{\lambda}_i) \\ \langle \partial_{\boldsymbol{\lambda}}^*\overline{F} \rangle_i = \partial_{\boldsymbol{\lambda}_i} F_i(\boldsymbol{z}_i^*(\boldsymbol{\lambda})/w_i^*, \boldsymbol{\lambda}_i) \end{cases} \tag{10}$$

$$\begin{cases} \langle \partial_{\boldsymbol{z}}^*\boldsymbol{\psi}(\zeta) \rangle_{ij} = p_{i\to j}(\delta_i)\partial_{\boldsymbol{z}_i}^*\boldsymbol{\varphi}_i(\xi_i), \forall j \in [n] \\ \langle \partial_{\boldsymbol{\lambda}}^*\boldsymbol{\psi}(\zeta) \rangle_{ij} = p_{i\to j}(\delta_i)\partial_{\boldsymbol{\lambda}_i}^*\boldsymbol{\varphi}_i(\xi_i), \forall j \in [n] \end{cases} \tag{11}$$

where, $\partial_{\boldsymbol{z}_i}^*\boldsymbol{\varphi}_i(\xi_i)$ and $\partial_{\boldsymbol{\lambda}_i}^*\boldsymbol{\varphi}_i(\xi_i)$ are Jacobians of $\boldsymbol{\varphi}_i(\boldsymbol{z}_i, \boldsymbol{\lambda}_i; \xi_i, w_i^*)$ with respect to $\boldsymbol{z}_i$ and $\boldsymbol{\lambda}_i$, respectively. We also remark the locality of hyper-gradient; the i-th agent only needs $\widehat{\mathrm{d}_{\boldsymbol{\lambda}_i}\overline{F}} = \langle \widehat{\mathrm{d}_{\boldsymbol{\lambda}}\overline{F}} \rangle_i \in \mathbb{R}^{d_\lambda}$ to perform the outer-step (Eq. (7)).

SRB Fails on Directed Networks. This paragraph presents our observation that SRB is surely commutable only when the communication networks are *undirected.*

We examine whether Eq. (9) can be computed over random directed networks for any agent and edge realizations. Let $\boldsymbol{u}_i^{(m)} = \langle \boldsymbol{u}^{(m)} \rangle_i \in \mathbb{R}^{d_x}$ and $\boldsymbol{v}_i^{(m)} = \langle \boldsymbol{v}^{(m)} \rangle_i \in \mathbb{R}^{d_\lambda}$ for all $i \in [n]$. Regarding Eq. (9a), only the i-th agent can obtain the initial vectors $\boldsymbol{u}_i^{(0)}$ and $\boldsymbol{v}_i^{(0)}$ from Eq. (10). According to the locality of the initial vectors, the i-th agent should be responsible for updating the $\boldsymbol{u}_i^{(m)}$ and $\boldsymbol{v}_i^{(m)}$ in Eq. (9b), obtaining the following local update iterations.

$$\begin{cases} \boldsymbol{v}_i^{(m+1)} \leftarrow \partial_{\boldsymbol{\lambda}_i}^*\boldsymbol{\varphi}_i(\breve{\xi}_i^{(m)})\sum_{j=1}^{n} p_{i\to j}(\breve{\delta}_i^{(m)})\boldsymbol{u}_j^{(m)} + \boldsymbol{v}_i^{(m)}, \\ \boldsymbol{u}_i^{(m+1)} \leftarrow \partial_{\boldsymbol{z}_i}^*\boldsymbol{\varphi}_i(\widehat{\xi}_i^{(m)})\sum_{j=1}^{n} p_{i\to j}(\widehat{\delta}_i^{(m)})\boldsymbol{u}_j^{(m)}. \end{cases}$$

To complete the iterations, the i-th agent must receive $\boldsymbol{u}_j^{(m)}$ from all the j-th agent such that $\delta_{i\to j} = 1$. However, receiving $\boldsymbol{u}_j^{(m)}$ is not surely possible as the realization of the communication channel from j to i (i.e., $\delta_{j\to i} = 1$) may not coincide with $\delta_{i\to j} = 1$ (Fig. 2-a). In other words, the decentralized computation of SRB is available only when the communications are undirected (i.e., $\delta_{i\to j} = \delta_{j\to i}$).

4.3 Hyper-Gradient Push (HGP)

We now propose HGP which enables any i-th agent to estimate $\mathrm{d}_{\boldsymbol{\lambda}_i}\overline{F}$ over random directed networks. HGP relaxes the undirected communication constraint by virtually reversing the direction of communication while preserving unbiasedness from SRB.

We first assume that the i-th agent knows the receiving frequency $\overline{\delta}_{j\to i} = \mathbb{E}_{\delta}[\delta_{j\to i}]$ and expected weighted edge $\overline{p}_{i\to j} = \mathbb{E}_{\delta}[p_{i\to j}(\delta_i)]$ for all j. In practice, agents can estimate them through SGP steps.

HGP no longer lets the i-th agent receive $\boldsymbol{u}_j^{(m)}$ from sending edges $\delta_{i\to j}$ but allows obtaining them through receiving edges $\delta_{j\to i}$ while debiasing $\boldsymbol{u}_j^{(m)}$ to maintain unbiassedness. More specifically, we replace $p_{i\to j}(\delta_i)$ in Eq. (11) with

$$\widetilde{p}_{j\to i}(\delta_{j\to i}) = \overline{p}_{i\to j}\frac{\delta_{j\to i}}{\overline{\delta}_{j\to i}} \tag{12}$$

to introduce matrices $\partial_{\boldsymbol{z}}^{*}\widetilde{\boldsymbol{\psi}}(\zeta) \in \mathbb{R}^{nd_{\boldsymbol{x}}\times nd_{\boldsymbol{x}}}$ and $\partial_{\boldsymbol{\lambda}}^{*}\widetilde{\boldsymbol{\psi}}(\zeta) \in \mathbb{R}^{nd_{\boldsymbol{\lambda}}\times nd_{\boldsymbol{x}}}$ such that for every $i, j \in [n]$,

$$\begin{aligned}\langle \partial_{\boldsymbol{z}}^{*}\widetilde{\boldsymbol{\psi}}(\zeta)\rangle_{ij} &= \widetilde{p}_{j\to i}(\delta_{j\to i})\partial_{z_i}^{*}\boldsymbol{\varphi}_i(\xi_i),\\ \langle \partial_{\boldsymbol{\lambda}}^{*}\widetilde{\boldsymbol{\psi}}(\zeta)\rangle_{ij} &= \widetilde{p}_{j\to i}(\delta_{j\to i})\partial_{\boldsymbol{\lambda}_i}^{*}\boldsymbol{\varphi}_i(\xi_i).\end{aligned}$$

By replacing $\partial_{\boldsymbol{z}}^{*}\boldsymbol{\psi}(\zeta)$ and $\partial_{\boldsymbol{\lambda}}^{*}\boldsymbol{\psi}(\zeta)$ in Eq. (9b) with $\partial_{\boldsymbol{z}}^{*}\widetilde{\boldsymbol{\psi}}(\zeta)$ and $\partial_{\boldsymbol{\lambda}}^{*}\widetilde{\boldsymbol{\psi}}(\zeta)$, respectively, we obtain HGP as follows.

Algorithm 2: Hyper-Gradient Push (HGP)

```
1 $\widetilde{\boldsymbol{u}}_i^{(0)} \leftarrow \partial_{z_i}F_i(z_i^*(\boldsymbol{\lambda})/w_i^*, \boldsymbol{\lambda}_i), \quad \widetilde{\boldsymbol{v}}_i^{(0)} \leftarrow \partial_{\boldsymbol{\lambda}_i}F_i(z_i^*(\boldsymbol{\lambda})/w_i^*, \boldsymbol{\lambda}_i)$
2 foreach $m = 0$ to $M-1$ do
3     $\widetilde{\boldsymbol{v}}_i^{(m+1)} \leftarrow \partial_{\boldsymbol{\lambda}_i}^{*}\boldsymbol{\varphi}_i(\check{\xi}_i^{(m)})\sum_{j=1}^{n}\widetilde{p}_{j\to i}(\check{\delta}_{j\to i}^{(m)})\widetilde{\boldsymbol{u}}_j^{(m)} + \widetilde{\boldsymbol{v}}_i^{(m)}$
4     $\widetilde{\boldsymbol{u}}_i^{(m+1)} \leftarrow \partial_{z_i}^{*}\boldsymbol{\varphi}_i(\widehat{\xi}_i^{(m)})\sum_{j=1}^{n}\widetilde{p}_{j\to i}(\widehat{\delta}_{j\to i}^{(m)})\widetilde{\boldsymbol{u}}_j^{(m)}$
5 return $\widetilde{\mathrm{d}_{\boldsymbol{\lambda}_i}\overline{F}} = \widetilde{\boldsymbol{v}}_i^{(M)}$
```

Alg. 2 is surely computable even on directed networks because the i-th agent needs to receive $\widetilde{\boldsymbol{u}}_j^{(m)}$ from the agents with $\delta_{j\to i} = 1$, which is always possible (Fig. 2-b).

We note that $\widetilde{\mathrm{d}_{\boldsymbol{\lambda}_i}\overline{F}}$ is unbiased from $\widehat{\mathrm{d}_{\boldsymbol{\lambda}_i}\overline{F}}$ because Eq. (12) and the independence of edge realization ensure $\langle \partial_{\boldsymbol{z}}^{*}\widetilde{\boldsymbol{\psi}}(\zeta)\rangle_{ij}$ and $\langle \partial_{\boldsymbol{\lambda}}^{*}\boldsymbol{\psi}(\zeta)\rangle_{ij}$ are unbiased. In addition, HGP enjoys low communication and time complexity; each agent only exchange $\widetilde{\boldsymbol{u}}_i^{(m)} \in \mathbb{R}^{d_{\boldsymbol{x}}}$ and compute Jacobian-vector products.

5 EXPERIMENTS

This section empirically validates the advantages of our PDBO with HGP: the agent cooperation and the ability to run on random directed networks. To do so, we applied and benchmarked PDBO on FL personalization tasks over simulated random directed networks. See Appendix A for the detailed settings.

Settings. We conducted personalization benchmarks on different tasks: handwritten character recognition (EMNIST [3]), natural image classification (CIFAR10 and CIFAR100 [8]), and language modeling (Shakespeare [2, 14]) on a simulated random directed communication network. For every task, we created heterogeneous datasets such that every agent has its unique data distribution. We simulated the random directed network by letting every edge $\delta_{j\to i}$ independently realize at probability $\overline{\delta}_{j\to i}$. For every i and j, $\overline{\delta}_{j\to i}$ was sampled from the uniform distribution over $[0.4, 0.8]$.

Table 1: Test accuracy on personalization benchmarks (average / bottom 10% percentile).

	EMNIST	CIFAR10	CIFAR100	Shakespeare
`SGP`	81.3 / 73.5	73.0 / 64.2	42.9 / 35.6	28.8 / 24.6
`Local`	65.6 / 51.0	67.2 / 46.5	39.1 / 30.6	27.0 / 20.9
`PDBO-MTL(local)`	<u>82.8</u> / 74.7	73.0 / 64.2	43.9 / 38.2	38.9 / 34.8
`PDBO-MTL`	<u>82.8</u> / <u>75.2</u>	<u>73.5</u> / <u>65.9</u>	<u>44.1</u> / <u>38.7</u>	<u>39.7</u> / <u>36.5</u>

We introduced a personalization method as special a case of PDBO, which we call `PDBO-MTL`. `PDBO-MTL` let each agent train an ensemble predictor that outputs weighted average predictions across base predictors. `PDBO-MTL` trained the parameters of $K \in \mathbb{N}$ base predictors as the inner-problem and optimized the outer-parameters $\boldsymbol{\lambda}_i \in \mathbb{R}^K$ to obtain an ensemble weight vector $\boldsymbol{\sigma}(\boldsymbol{\lambda}_i) \in [0,1]^K$, where $\boldsymbol{\sigma}$ is the softmax function. To solve PDBO, any i-th agent ran T SGP inner-iterations using the train dataset. Every agent then ran HGP with respect to the cross-entropy loss on the train dataset to estimate its hyper-gradient. The outer-optimization ran multiple outer-steps from the zeros initial outer-parameters $\mathbf{0}_{d_{\boldsymbol{\lambda}}}$. We reported the average test accuracy at an outer-step that gained the best validation accuracy.

We compared `PDBO-MTL` against three baselines. `SGP` and `Local` trained ensemble predictors with non-weighted averaged predictions by SGP and the local SGD, respectively. Another baseline, referred to as `PDBO-MTL(local)`, followed the `PDBO-MTL` but utilized the local outer-gradient $\partial_{\boldsymbol{\lambda}}^{*}\overline{F}$ as the hyper-gradient. Similarly to previous works [10, 11], `PDBO-MTL(local)` lacks agent cooperation as it disregards the hyper-gradient of the other agents' outer-costs.

Results. Table 1 shows the average test accuracy. Our `PDBO-MTL` outperformed the baselines over all tasks, except for EMNIST classification. The superior performance of `PDBO-MTL` compared to `PDBO-MTL(local)` verified that agent cooperation enhanced personalization performance. We also evaluated the accuracy of the bottom 10% percentile of the agents (Table 1). `PDBO-MTL` improved accuracy at the 10% percentile over the baselines in all tasks, confirming that the accuracy gained by our personalization was shared among all agents.

6 CONCLUSION

This paper proposed a decentralized bilevel optimization problem called PDBO, which optimizes an agent-wise personalized parameter as the outer-problem requiring agent cooperation; every agent is responsible for minimizing the total outer-cost over all agents. We then presented HGP which is a decentralized computation algorithm for the hyper-gradient that runs over random directed networks. Empirical results showed that PDBO solved with HGP outperforms baselines on personalized FL benchmarks conducted on simulated directed communications networks, demonstrating the advantages of our approach.

REFERENCES

[1] Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Mike Rabbat. 2019. Stochastic gradient push for distributed deep learning. In *International Conference on Machine Learning*.

[2] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097* (2018).

[3] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. 2017. EMNIST: Extending MNIST to handwritten letters. In *International Joint Conference on Neural Networks*.

[4] Justin Domke. 2012. Generic methods for optimization-based modeling. In *International Conference on Artificial Intelligence and Statistics*.

[5] Saeed Ghadimi and Mengdi Wang. 2018. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246* (2018).

[6] Riccardo Grazzi, Massimiliano Pontil, and Saverio Salzo. 2021. Convergence properties of stochastic hypergradients. In *International Conference on Artificial Intelligence and Statistics*.

[7] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

[8] A Krizhevsky. 2009. Learning Multiple Layers of Features from Tiny Images. *Master's thesis, University of Tront* (2009).

[9] Wei Li and Andrew McCallum. 2006. Pachinko allocation: DAG-structured mixture models of topic correlations. In *International Conference on Machine Learning*.

[10] Zhuqing Liu, Xin Zhang, Prashant Khanduri, Songtao Lu, and Jia Liu. 2022. INTERACT: Achieving Low Sample and Communication Complexities in Decentralized Bilevel Learning over Networks. *arXiv preprint arXiv:2207.13283* (2022).

[11] Songtao Lu, Xiaodong Cui, Mark S Squillante, Brian Kingsbury, and Lior Horesh. 2022. Decentralized Bilevel Optimization for Personalized Client Learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

[12] Sébastien Marcel and Yann Rodriguez. 2010. Torchvision the machine-vision package of torch. In *the 18th ACM International Conference on Multimedia*.

[13] Othmane Marfoq, Giovanni Neglia, Aurélien Bellet, Laetitia Kameni, and Richard Vidal. 2021. Federated multi-task learning under a mixture of distributions. In *Advances in Neural Information Processing Systems*.

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

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

[16] Sashank J Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and Hugh Brendan McMahan. 2020. Adaptive Federated Optimization. In *International Conference on Learning Representations*.

[17] Pouya Rezaienia, Bahman Gharesifard, Tamás Linder, and Behrouz Touri. 2019. Push-sum on random graphs: almost sure convergence and convergence rate. *IEEE Trans. Automat. Control* 65, 3 (2019), 1295–1302.

[18] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *The IEEE Conference on Computer Vision and Pattern Recognition*.

[19] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. 2022. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[20] Naoyuki Terashita and Satoshi Hara. 2023. Decentralized Hyper-Gradient Computation over Time-Varying Directed Networks. *arXiv preprint arXiv:2210.02129* (2023).

[21] Konstantinos I Tsianos, Sean Lawlor, and Michael G Rabbat. 2012. Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning. In *50th Annual Allerton Conference on Communication, Control, and Computing*.

[22] Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. 2017. Decentralized collaborative learning of personalized models over networks. In *International Conference on Artificial Intelligence and Statistics*.

[23] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. 2019. Federated Learning with Matched Averaging. In *International Conference on Learning Representations*.

[24] Shuoguang Yang, Xuezhou Zhang, and Mengdi Wang. 2022. Decentralized Gossip-Based Stochastic Bilevel Optimization over Communication Networks. *arXiv preprint arXiv:2206.10870* (2022).

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

A DETAILED EXPERIMENTAL SETTINGS

This section explains detailed settings of experiments in Section 5

A.1 Datasets and Models

We conducted personalization benchmarks on different tasks: handwritten character recognition (EMNIST [3]), natural image classification (CIFAR10 and CIFAR100 [8]), and language modeling (Shakespeare [2, 14]) on a simulated random directed communication network.

For EMNIST, we used 10% of the original dataset as in Marfoq et al. [13]. We generated a personalized version of EMNIST involving a combination of heterogeneous distributions in the following fashion. First, clusters are equally separated into three clusters which use mean and variance for the normalization of inputs, generating heterogeneous input distributions. For each cluster, we sampled mean and variance values independently from the uniform distribution with range $[0, 1]$. Second, instances are distributed across 100 agents using n-dimensional Dirichlet distribution of parameter $\alpha = 0.4$ for each label, modeling heterogeneous label distributions. Any agent splits the assigned dataset set of EMNIST by randomly selecting 20% of instances to generate the test dataset. The remaining instances are split into train and validation datasets with the ratio of $3 : 1$. We use the validation dataset only for the early stopping in outer-optimization of `PDBO-MTL` and `PDBO-MTL(Local)`. We adopted CNN with two convolution and two fully-connected layers for the base predictor as in Marfoq et al. [13].

For image classification on CIFAR10, we distributed samples with the same labels across agents according to a symmetric Dirichlet distribution with parameter 0.4, as in Marfoq et al. [13], Wang et al. [23], to create a federated version. We used 40% of the total data as the train and validation dataset in a 3:1 ratio and the rest as the test dataset. We also tested image classification using CIFAR100 exploiting the availability of "coarse" and "fine" labels, using a two-stage Pachinko allocation method [9] as in Marfoq et al. [13], Reddi et al. [16], to distribute 900, 300, and 1800 sized train, validation, test datasets to each agent, respectively. Pachinko allocation ran with the parameters adopted in Marfoq et al. [13]. For both CIFAR10 and CIFAR100, we set $n = 20$ and trained MobileNet-v2 [18], implemented in TorchVision[12], with an additional linear layer.

The Shakespeare dataset was naturally divided by assigning all lines from the same character to the same agent as in Marfoq et al. [13] and McMahan et al. [14]. From 728 characters, we randomly selected $n = 20$ characters and assigned each of them to an agent. We trained two stacked-LSTM layers with 256 hidden units followed by a densely-connected layer adopted in [13], to predict the next character from a sequence of 200 English characters as input. The model embeds 80 characters into a learnable 8-dimensional embedding space. For each agent, we used 80% of lines as the train and validation dataset in a 3:1 ratio and the rest as the test dataset. The lines are split from the beginning in the order train, validation, and test to simulate the practical time dependence between datasets.

A.2 Approaches

We introduced a personalization method as special a case of PDBO, which we call `PDBO-MTL`. `PDBO-MTL` is obtained by formulating FedEM [13] as PDBO to model the clustered input distributions. `PDBO-MTL` let each agent train an ensemble predictor that outputs weighted average predictions across multiple base predictors. `PDBO-MTL` trained the parameters of $K \in \mathbb{N}$ base predictors as the inner-problem and optimized the outer-parameters $\boldsymbol{\lambda}_i \in \mathbb{R}^K$ to obtain the ensemble weight vector, $\boldsymbol{\sigma}(\boldsymbol{\lambda}_i) \in [0, 1]^K$.

For baselines, due to the absence of personalization methods applicable to random directed communication networks, the local SGD (`Local`) and standard SGP [15] (`SGP`) were adopted. We trained ensemble models with uniform prediction weights for `Local` and `SGP` to fairly compare the performance difference between the baselines and our approach. This allows us to exclude architectural differences from the reasons for performance improvements. We also evaluated `PDBO-MTL(local)`, which follows the `PDBO-MTL` but utilized the local outer-gradient $\partial^*_{\boldsymbol{\lambda}}\overline{F}$ as the hyper-gradient. Similarly to previous works [10, 11], `PDBO-MTL(local)` lacks agent cooperation since it disregards the hyper-gradient of the other agents' outer-costs.

A.3 Training Procedure

For all approaches, we performed the inner-optimization (SGP or local SGD) with learning rates scheduled to be multiplied by 0.1 at the last 100 and 50 steps. For `PDBO-MTL`, `PDBO-MTL(local)`, and `SGP`, we adopt an alternative formulation of SGP [15], which performs a local gradient step after the communication, as it showed better performance in the inner-optimization. `PDBO-MTL` and `PDBO-MTL(local)` performed multiple outer-steps. For all $i \in [n]$ in the outer-problem, we ran Adam [7] iterations with $(\beta_1, \beta_2) = (0.90, 0.99)$ from the initial outer-parameters $\mathbf{0}_K$. Before every outer-step, every agent ran M HGP iterations to obtain its hyper-gradient. We also made a practical modification in HGP to sample $\partial^*_z\widetilde{\boldsymbol{\psi}}(\zeta)$ and $\partial^*_{\boldsymbol{\lambda}}\widetilde{\boldsymbol{\psi}}(\zeta)$ together using the same $\zeta^{(m)}$ at the m-th round, which leads the same length of the Neumann series with the half sampling costs of the original HGP, although they are no more unbiased. For `PDBO-MTL` and `PDBO-MTL(local)`, F_i is the average cross-entropy loss over the local train dataset of the i-th agent and L2 regularization loss of $\boldsymbol{\lambda}_i$ for all $i \in [n]$. We reported the mean test accuracy of an intermediate step that had maximum validation accuracy (i.e., early stopping). Hyper-parameters used for each approach are shown in Table 2.

Table 2: Hyperparameters that produced results in Table 1

Task	Method	Hyper-parameter							
		Num. of inner-steps	Inner L2 reg. decay	Inner learning rate	Batch-size	Num. of outer-steps	Outer learning rate	Outer L2 reg. decay	M
EMNIST	PDBO-MTL PDBO-MTL(Local)	600	0.001	0.5	128	5	1.0	0.01	200 0
	SGP Local	600	0.001	0.5	128	n/a	n/a	n/a	n/a
CIFAR10	PDBO-MTL PDBO-MTL(Local)	1200	0.001	0.1	128	10	1.0	0.01	20 0
	SGP Local	1200	0.001	0.1	128	n/a	n/a	n/a	n/a
CIFAR100	PDBO-MTL PDBO-MTL(Local)	1200	0.001	0.2	128	10	1.0	0.01	20 0
	SGP Local	1200	0.001	0.2	128	n/a	n/a	n/a	n/a
Shakespeare	PDBO-MTL PDBO-MTL(Local)	1200	0.001	10.0	128	10	1.0	0.01	20 0
	SGP Local	1200	0.001	10.0	128	n/a	n/a	n/a	n/a