Find Your Friends: Personalized Federated Learning with the Right Collaborators

Anonymous Author(s)
Affiliation
Address
email

Abstract

In the traditional federated learning setting, a central server coordinates a network of clients to train one global model. However, the global model may serve many clients poorly due to data heterogeneity. Moreover, there may not exist a trusted central party that can coordinate the clients to ensure that each of them can benefit from others. To address these concerns, we present a novel decentralized framework, *FedeRiCo*, where each client can learn as much or as little from other clients as is optimal for its local data distribution. Based on expectation-maximization, FedeRiCo estimates the utilities of other participants' models on each client's data so that everyone can select the right collaborators for learning. As a result, our algorithm outperforms other federated, personalized, and/or decentralized approaches on several benchmark datasets, being the *only* approach that consistently performs better than training with local data only.

1 Introduction

Federated learning (FL) [McMahan et al., 2017] offers a framework in which a single server-side model is collaboratively trained across decentralized datasets held by clients. It has been successfully deployed in practice for developing machine learning models without direct access to user data, which is essential in highly regulated industries such as banking and healthcare [Long et al., 2020, Sadilek et al., 2021]. For example, several hospitals that each collect patient data may want to merge their datasets for increased diversity and dataset size but are prohibited due to privacy regulations.

Traditional FL methods like Federated Averaging (FedAvg) [McMahan et al., 2017] can achieve noticeable improvement over local training when the participating clients' data is homogeneous. In practice, however, each client is likely to have a different data distribution from the others [Zhao et al., 2018, Adnan et al., 2022], making it difficult to learn a global model that works well for all participants. As a simple example, consider a scenario where each client seeks to fit a linear model to limited data on an interval of the sine curve, as shown in Fig. 1. This is analogous to the FL setting where several participating clients would like to collaborate, but each client only has access to data from its own data distribution. It is clear that no single linear model can adequately describe the entire joint dataset, so a global model learned by FedAvg performs poorly, as shown by the dotted line. Ideally, each client should benefit from collaboration by increasing the effective size and diversity of data, but in practice, forcing everyone to use the same global model without proper personalization can hurt performance on their own data distribution [Kulkarni et al., 2020, Tan et al., 2022].

To address this, we propose **Fede**rating with the **Ri**ght **Co**llaborators (FedeRiCo), a novel framework that enables each client to choose the *right collaborators*. This is illustrated in the right plot of Fig. 1: each client is able to correctly leverage information from the neighboring clients when it is beneficial to do so. The final personalized models fit the local distributions well, as demonstrated in the left plot.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

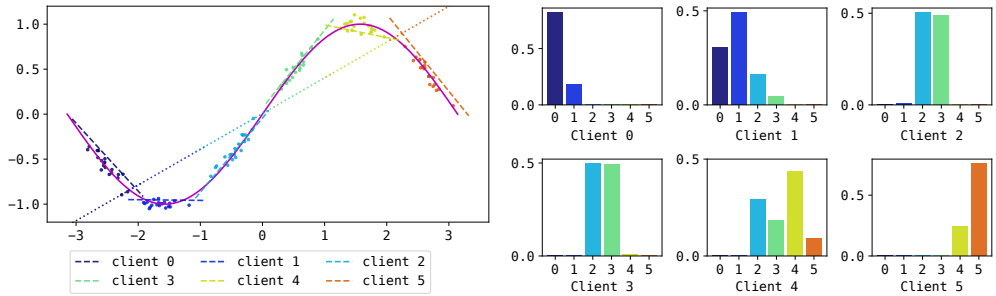


Figure 1: **Left:** Noisy data points generated for each client along a sine curve (solid magenta line) where the x-axis and y-axis correspond to input and output respectively. The corresponding model learned by FedAvg (dotted line) fails to adapt to the local data seen by each client, in contrast to the models learned by each client using our FedeRiCo (dashed lines). **Right:** The weights used by FedeRiCo to average participant outputs for each client. As the client index increases, the data is generated from successive intervals of the sine curve, and collaborator weights change accordingly.

More specifically, our FedeRiCo assumes that each client has an underlying data distribution, and exploits this structure among the clients' data. By selecting the most relevant neighbours, each client collaborates as much or as little as they need, and learns a personalized mixture model to fit the local data. Additionally, FedeRiCo achieves this in a fully decentralized manner that is not beholden to any central authority [Li et al., 2021a, Huang et al., 2021, Kalra et al., 2021].

2 Federated Learning with the Right Collaborators

In this section, we provide the problem formulation, followed by detailed derivations of our method. We consider a federated learning (FL) scenario with K clients. Let $[K] := \{1, 2, \ldots, K\}$ denote the set of positive integers up until K. Each client $i \in [K]$ consists of a local dataset $D_i = \{(\mathbf{x}_s^{(i)}, y_s^{(i)})\}_{s=1}^{n_i}$ where n_i is the number of examples for client i, and the input $\mathbf{x}_s \in \mathcal{X}$ and output $y_s \in \mathcal{Y}$ are drawn from a joint distribution $\mathcal{D}_i$ over the space $\mathcal{X} \times \mathcal{Y}$.

The goal of personalized FL is to find a prediction model $h_i : \mathcal{X} \mapsto \mathcal{Y}$ that can perform well on the local distribution $\mathcal{D}_i$ for each client. One of the main challenges in personalized FL is that if two clients i and j have vastly different data distributions, forcing them to collaborate is likely to result in worse performance compared to local training without collaboration. Our method, **Fede**rating with the **Ri**ght **Co**llaborators (FedeRiCo), is designed to address this problem so that each client can choose to collaborate or not, depending on their data distributions. FedeRiCo is a decentralized framework (i.e. without a central server). For illustration purposes, Section 2.1 first demonstrates how our algorithm works in a *hypothetical* all-to-all communication setting. This assumption is relaxed in Section 2.2, which describes how to use FedeRiCo with limited communication.

2.1 FedeRiCo with all-to-all communication

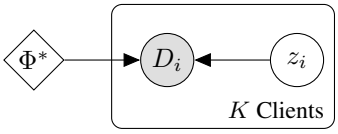


Figure 2: Graphical model

Note that every local distribution $\mathcal{D}_i$ can always be represented as a mixture of $\{\mathcal{D}_j\}_{j=1}^K$ with some client weights $\boldsymbol{\pi}_i = [\pi_{i1}, \ldots, \pi_{iK}] \in \Delta^K$, where Δ^K is the $(K-1)$-dimensional simplex[1]. Let z_i be the latent assignment variable of client i, and $\Pi := [\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_K]^\top$ be the prior $\Pi_{ij} = \Pr(z_i = j)$. Suppose that the conditional probability $p_i(y|\mathbf{x})$ satisfies $-\log p_i(y|\mathbf{x}) = \ell(h_{\boldsymbol{\phi}_i^*}(\mathbf{x}), y) + c$ for some parameters $\boldsymbol{\phi}_i^* \in \mathbb{R}^d$, loss function $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^+$, and normalization constant c. By using the stacked notation $\Phi^* = [\boldsymbol{\phi}_1^*, \ldots, \boldsymbol{\phi}_K^*] \in \mathbb{R}^{d \times K}$, Fig. 2 shows the graphical model of the local dataset generation. Our goal is to learn the parameters $\Theta := (\Phi, \Pi)$ by maximizing the log-likelihood:

$$f(\Theta) := \frac{1}{n} \log p(D; \Theta) = \frac{1}{n} \sum_{i=1}^{K} \log p(D_i; \Theta) = \frac{1}{n} \sum_{i=1}^{K} \log \sum_{z_i=1}^{K} p(D_i, z_i; \Theta). \tag{1}$$

[1]One-hot $\boldsymbol{\pi}_i$ is always feasible, but other mixing coefficients may exist.

where $D := \cup_i D_i$ and $n := \sum_i n_i$. One standard approach to optimization with latent variables is expectation maximization (EM) [Dempster et al., 1977]. The corresponding variational lower bound is given by (all detailed derivations of this section can be found in Appendix A)

$$\mathcal{L}(q, \Theta) := \frac{1}{n} \sum_i \mathbb{E}_{q(z_i)}[\log p(D_i, z_i; \Theta)] + C, \tag{2}$$

where C is a constant not depending on Θ. To obtain concrete objective functions suitable for optimization, we further assume that $p_i(x) = p(x), \forall i \in [K]$. Similar to Marfoq et al. [2021], this assumption is required due to technical reasons and can be relaxed if needed. With this assumption, we perform the following updates at each iteration t:

- **E-step:** For each client, find the best q, which is the posterior $p(z_i = j | D_i; \Theta^{(t-1)})$ given the current parameters $\Theta^{(t-1)}$:

$$w_{ij}^{(t)} := q^{(t)}(z_i = j) \propto \Pi_{ij}^{(t-1)} \exp\left[-\sum_{s=1}^{n_i} \ell\left(h_{\boldsymbol{\phi}_j^{(t-1)}}(\mathbf{x}_s^{(i)}),\ y_s^{(i)} \right) \right]. \tag{3}$$

- **M-step:** Given the posterior $q^{(t)}$ from the E-step, maximize $\mathcal{L}$ w.r.t. $\Theta = (\Phi, \Pi)$:

$$\Pi_{ij}^{(t)} = w_{ij}^{(t)} \qquad \text{and} \qquad \Phi^{(t)} \in \operatorname*{argmin}_{\Phi} \frac{1}{n} \sum_{i=1}^{K} \widehat{\mathcal{L}}_{w,i}(\Phi) \tag{4}$$

$$\text{where} \qquad \widehat{\mathcal{L}}_{w,i}(\Phi) := \sum_{j=1}^{K} w_{ij}^{(t)} \sum_{s=1}^{n_i} \ell\left(h_{\boldsymbol{\phi}_j}(\mathbf{x}_s^{(i)}),\ y_s^{(i)} \right). \tag{5}$$

Bear in mind that each client can only see its local data D_i in the federated setting. The E-step is easy to compute once the models from other clients $\boldsymbol{\phi}_j, j \neq i$ are available. $\Pi_{ij}^{(t)}$ is also easy to obtain as the posterior $w_{ij}^{(t)}$ is stored locally. However, $\Phi^{(t)}$ is trickier to compute since each client can potentially update Φ in different directions due to data heterogeneity amongst the clients. To stabilize optimization and avoid overfitting from client updates, we rely on small gradient steps in lieu of full optimization in each round. To compute $\Phi^{(t)}$ algorithmically, each client i:

1. Fixes $w_{ij}^{(t)}$ and computes the local gradient $\nabla \widehat{\mathcal{L}}_{w,i}(\Phi^{(t-1)})$ on local data D_i.
2. Broadcasts $\nabla \widehat{\mathcal{L}}_{w,i}(\Phi^{(t-1)})$ to and receives $\nabla \widehat{\mathcal{L}}_{w,j}(\Phi^{(t-1)})$ from other clients $j \neq i$. The models are updated based on the aggregated gradient with step size $\eta > 0$:

$$\Phi^{(t)} = \Phi^{(t-1)} - \eta \sum_{j=1}^{K} \nabla \widehat{\mathcal{L}}_{w,j}(\Phi^{(t-1)}). \tag{6}$$

Each client uses $\widehat{h}_i(\mathbf{x}) = \sum_j w_{ij}^{(t)} h_{\boldsymbol{\phi}_j^{(t)}}(\mathbf{x})$ for prediction after convergence.

Remark 1 The posterior $w_{ij}^{(t)}$ (or equivalently the prior in the next iteration $\Pi_{ij}^{(t)}$) reflects the importance of model $\boldsymbol{\phi}_j$ on the data D_i. When $w_{ij}^{(t)}$ is one-hot with a one in the ith position, client i can perform learning by itself without collaborating with others. When $w_{ij}^{(t)}$ is more diverse, client i can find the right collaborators with useful models $\boldsymbol{\phi}_j$. Such flexibility enables each client to make its own decision on whether or not to collaborate with others, hence the name of our algorithm.

Remark 2 Unlike prior work [Mansour et al., 2020, Marfoq et al., 2021], our assignment variable z and probability Π are on the client level; each client is assigned a mixture of distributions. When all clients share the same prior , our algorithm is similar to HypCluster [Mansour et al., 2020]. Marfoq et al. [2021] uses a similar formulation, but with assignment z on the instance level, applying to each individual data point. This can cause several issues at inference time, as assignments for new data points are unknown. We refer interested readers to Appendix B and Section 3 for further comparison.

Theoretical Convergence Under some regularity assumptions, our algorithm converges as follows:

Theorem 2.1. *[Convergence] Under Assumptions F.1-F.6, when the clients use SGD with learning rate $\eta = \frac{a_0}{\sqrt{T}}$, and after sufficient rounds T, the iterates of our algorithm satisfy*

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E} \|\nabla_{\Phi} f(\Phi^t, \Pi^t)\|_F^2 \leq \mathcal{O}\left(\frac{1}{\sqrt{T}} \right), \qquad \frac{1}{T} \sum_{t=1}^{T} \Delta_{\Pi} f(\Phi^t, \Pi^t) \leq \mathcal{O}\left(\frac{1}{T^{3/4}} \right), \tag{7}$$

where the expectation is over random batches and $\Delta_\Pi f(\Phi^t, \Pi^t) := f(\Phi^t, \Pi^t) - f(\Phi^t, \Pi^{t+1}) \geq 0$.

Due to space limitations, further details and the complete proof are deferred to Appendix F. The above theorem shows that the gradient w.r.t. the model parameters Φ and the improvement over the mixing coefficients Π becomes small as we increase the number of training rounds T.

2.2 Communication-efficient protocol

So far we have discussed how FedeRiCo works with all-to-all communication, which is impractical at scale. Here, we will propose several modifications to ensure communication efficiency. Specifically, we address both the E-step (3) and the M-step (4), which require joint information of all models Φ.

E-step For client i, the key quantity to compute (3) is the loss $\ell(\boldsymbol{\phi}_j^{(t-1)})$, or likelihood $p(D_i|z_i = j; \Phi^{(t-1)})$, of other clients' models $\boldsymbol{\phi}_j, j \neq i$. Since the models Φ are being updated slowly, one can expect that $\ell(\boldsymbol{\phi}_j^{(t-1)})$ will be close to the loss $\ell(\boldsymbol{\phi}_j^{(t-2)})$ of the previous iteration. Therefore, each client can maintain a list of losses for all the clients, in each round sample a subset of clients using a sampling scheme $\mathcal{S}$ (e.g., uniform sampling), and only update the losses of the chosen clients.

M-step To clearly see how Φ is updated in the M-step, we can focus on the update to a specific client's model $\boldsymbol{\phi}_i$. According to (5) and (6), the update to $\boldsymbol{\phi}_i$ is given by

$$-\eta \sum_{j=1}^{K} w_{ji}^{(t)} \sum_{s=1}^{n_j} \nabla_{\boldsymbol{\phi}_i} \ell\left(h_{\boldsymbol{\phi}_i}(\mathbf{x}_s^{(j)}),\ y_s^{(j)}\right). \tag{8}$$

Note that the aggregation is based on $w_{ji}^{(t)}$ instead of $w_{ij}^{(t)}$. Intuitively, this suggests $\boldsymbol{\phi}_i$ should be updated based on how the model is being used by *other clients* rather than how client i itself uses it. If $\boldsymbol{\phi}_i$ does not appear to be useful to all clients, i.e. $w_{ji}^{(t)} = 0,\ \forall j$, it does not get updated. Therefore, whenever client i is sampled by another client j using the sampling scheme $\mathcal{S}$, it will send $\boldsymbol{\phi}_i$ to j, and receives the gradient update $\mathbf{g}_{ij} := w_{ji}^{(t)} \sum_{s=1}^{n_j} \nabla_{\boldsymbol{\phi}_i} \ell\left(h_{\boldsymbol{\phi}_i}(\mathbf{x}_s^{(j)}),\ y_s^{(j)}\right)$ from client j. One issue here is that $\mathbf{g}_{ij}$ is governed by $w_{ji}^{(t)}$, which could be arbitrarily small, leading to no effective update to $\boldsymbol{\phi}_i$. We will show how this can be addressed by using an ϵ-greedy sampling scheme.

ϵ-greedy sampling scheme $\mathcal{S}$ In each round, each client uniformly samples clients with probability $\epsilon \in [0, 1]$ and samples clients with the highest posteriors otherwise. This allows a trade off between emphasizing updates from high-performing clients (small ϵ), versus receiving updates from clients uniformly to find potential collaborators (large ϵ). We show the effect of varying the number M of sampled clients per round and ϵ in experiments.

Tracking the losses for the posterior The final practical consideration is the computation of the posterior $w_{ij}^{(t)}$. From the E-step (3) and the M-step (4), one can see that $w_{ij}^{(t)}$ is the softmax of the negative accumulative loss $L_{ij}^{(t)} := \sum_{\tau=1}^{t-1} \ell_{ij}^{(\tau)}$ over rounds (see Appendix A for derivation). However, the accumulative loss can be sensitive to noise and initialization. If one of the models, say $\boldsymbol{\phi}_j$, performs slightly better than other models for client i at the beginning of training, then client i is likely to sample $\boldsymbol{\phi}_j$ more frequently, thus enforcing the use of $\boldsymbol{\phi}_j$ even when other better models exist. To address this, we instead keep track of the exponential moving average of the loss with a momentum parameter $\beta \in [0, 1)$, $\widehat{L}_{ij}^{(t)} = (1-\beta)\widehat{L}_{ij}^{(t-1)} + \beta l_{ij}^{(t)}$, and compute $w_{ij}^{(t)}$ using $\widehat{L}_{ij}^{(t)}$. This encourages clients to seek new collaborators rather than focusing on existing ones.

3 Experiments

3.1 Experimental settings

We conduct a range of experiments to evaluate the performance of our proposed FedeRiCo with multiple datasets. Additional experiment details and results can be found in Appendix D.

Datasets We evaluate different methods on image-classification tasks with the CIFAR-10, CIFAR-100 [Krizhevsky et al., 2009], and Office-Home [Venkateswara et al., 2017] datasets. Particularly, we consider a non-IID data partition among clients by first splitting data into several groups with disjoint

label sets. Each group acts as a distribution, and each client samples from one distribution to form its local data. For each client, we randomly divide the local data with a 80%-20% train-test split.

Baseline methods We compare FedeRiCo to FedAvg [McMahan et al., 2017], which trains a single global model across all clients, as well as several personalized FL approaches: FedAvg with local tuning (FedAvg+) [Jiang et al., 2019], Clustered FL [Sattler et al., 2020], FedEM [Marfoq et al., 2021][2], FedFomo [Zhang et al., 2021], as well as a local training baseline. We report the mean and standard deviation of the resulting accuracies across random data splits and training seeds. Unless specified otherwise, we use 3 neighbors with $\epsilon = 0.3$ and $\beta = 0.6$ for FedeRiCo in all experiments. For FedEM, we use 4 components, which is sufficient to accommodate different numbers of label groups. For FedFomo, we hold out 20% of the training data for client weight calculations. For FedAvg+, we follow Marfoq et al. [2021] and update the local model with 1 epoch of local training.

3.2 Performance comparison

Table 1: Accuracy (in percentage) with different number of data distributions. Best results in bold.

	CIFAR-10 # of distributions			CIFAR-100 # of distributions			Office-Home # of distributions		
Method	2	3	4	2	3	4	2	3	4
FedAvg	11.44 ± 3.28	11.73 ± 3.68	13.93 ± 5.74	21.28 ± 5.04	17.41 ± 3.27	18.36 ± 3.68	66.58 ± 1.88	53.36 ± 4.21	51.25 ± 4.37
FedAvg+	12.45 ± 8.46	29.86 ± 17.85	45.65 ± 21.61	29.95 ± 1.07	35.33 ± 1.77	36.17 ± 3.27	80.21 ± 0.68	81.88 ± 0.91	84.50 ± 1.37
Local Training	40.09 ± 2.84	55.27 ± 3.11	69.03 ± 7.05	16.60 ± 0.64	25.99 ± 2.38	31.05 ± 1.68	76.76 ± 0.23	83.30 ± 0.32	88.05 ± 0.44
Clustered FL	11.50 ± 3.65	15.24 ± 5.79	16.43 ± 5.17	20.93 ± 3.57	23.15 ± 7.04	15.15 ± 0.60	66.58 ± 1.88	53.36 ± 4.21	51.25 ± 4.37
FedEM	41.21 ± 10.83	55.08 ± 6.71	63.61 ± 9.93	26.25 ± 2.40	24.11 ± 7.36	19.23 ± 2.58	22.59 ± 1.95	28.72 ± 1.83	22.46 ± 3.99
FedFomo	42.24 ± 8.32	59.45 ± 5.57	71.05 ± 6.09	12.15 ± 0.57	20.49 ± 2.90	24.53 ± 2.77	78.61 ± 0.78	82.57 ± 0.24	87.86 ± 0.77
FedeRiCo	**56.61** ± 2.51	**69.76** ± 2.25	**78.22** ± 4.80	**30.95** ± 1.62	**39.19** ± 1.64	**41.41** ± 1.07	**83.56** ± 0.49	**90.28** ± 0.75	**93.76** ± 0.12

The performance of each FL method is shown in Table 1. Following the settings introduced by Marfoq et al. [2021], each client is evaluated on its own local testing data and the average accuracies weighted by local dataset sizes are reported. We observe that FedeRiCo has the best performance across all datasets and number of data distributions. Here, local training can be seen as an indicator to assess if other methods benefit from client collaboration as local training has no collaboration at all. We observe that our proposed FedeRiCo is the only method that consistently outperforms local training, meaning that FedeRiCo is the only method that consistently encourages effective client collaborations. Notably, both FedEM and FedFomo performs comparably well to FedeRiCo on CIFAR-10 but worse when the dataset becomes more complex like CIFAR-100. This indicates that building the right collaborations among clients becomes a harder problem for more complex datasets. Moreover, FedEM can become worse as the number of distributions increases, even worse than local training, showing that it is increasingly hard for clients to participate effectively under the FedEM framework for complex problems with more data distributions.

In addition, Clustered FL has similar performance to FedAvg, indicating that it is hard for Clustered FL to split into the right clusters. In Clustered FL [Sattler et al., 2020], every client starts in the same cluster and a cluster split only happens when the FL objective is close to a stationary point, i.e. the norm of averaged gradient update from all clients inside the cluster is small. Therefore, in a non-i.i.d setting like ours, the averaged gradient update might always be noisy and large, as clients with different distributions are pushing diverse updates to the clustered model. As a result, the cluster splitting rarely happens which makes clustered FL more like FedAvg.

3.3 Client collaboration

In this section, we investigate client collaboration by plotting the personalized client weights $w_{ij}^{(t)}$ of FedeRiCo over training. With different client data distributions, we show that FedeRiCo can assign more weight to clients from the same distribution. As shown in Fig. 3, we observe that clients with similar distributions collaborate to make the final predictions. For example, clients 3, 4 and 7 use a mixture of predictions from each other (in light blue) whereas client 0 only uses itself for prediction since it is the only client coming from distribution 0 (in dark blue) in this particular random split.

On the contrary, as shown in Fig. 4, even with 4 components, FedEM fails to use all of them for predictions for the 4 different data distributions. In fact, clients 2, 3, 4, 6 and 7 coming from two different distributions are using only the model of component 3 for prediction, whereas component

[2]We use implementations from `https://github.com/omarfoq/FedEM` for Clustered FL and FedEM.

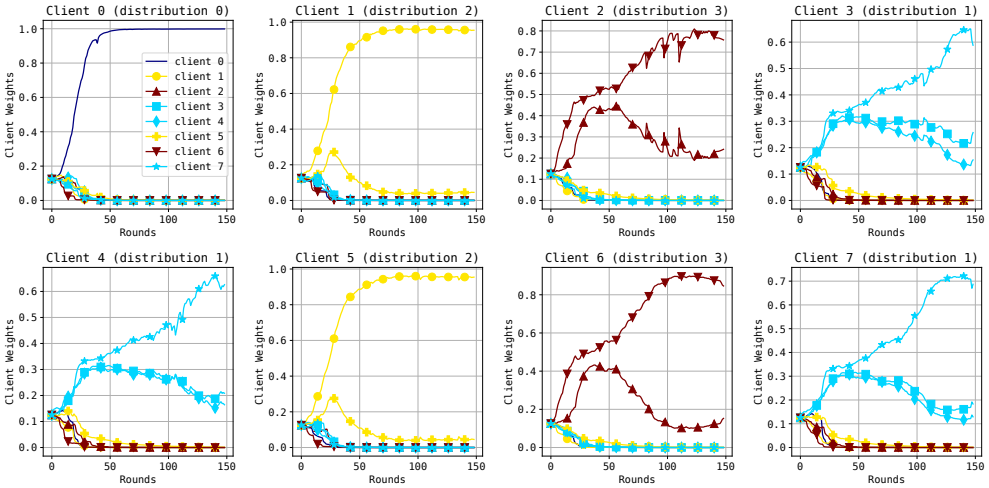


Figure 3: Client weights over time of FedeRiCo with CIFAR100 data and four different client distributions. Clients are color coded by their private data's distribution.

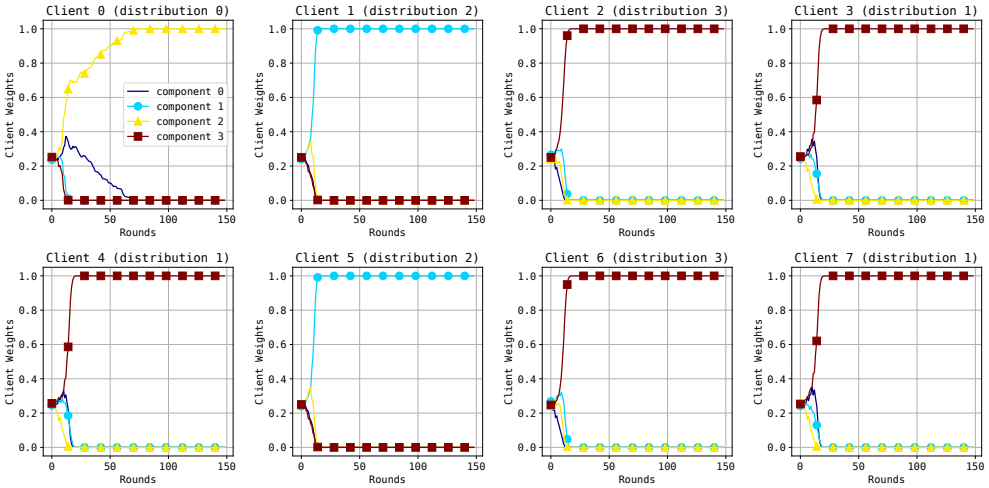


Figure 4: Component weights over training for FedEM with 4 components, on CIFAR100 data with 4 different client distributions. Clients are color coded by their private data's distribution.

0 is never used by any client. Based on this, we find FedeRiCo better encourages the clients to collaborate with other similar clients and less with different clients. Each client can collaborate as much or as little as they need. Additionally, since all the non-similar clients have a weight of (almost) 0, each client only needs a few models from their collaborators for prediction.

4 Conclusion and Future Work

In this paper, we proposed FedeRiCo, a novel framework for decentralized and personalized FL derived from expectation maximization for non-i.i.d client data. We evaluated FedeRiCo across 3 different datasets and demonstrated that FedeRiCo outperforms multiple existing personalized FL baselines by encouraging clients to collaborate with similar clients, i.e. the right collaborators.

While the decentralized FL scheme significantly reduces the risk of single point failure in the centralized FL setting, it also raises concerns about security risks with the absence of a mutually trusted central server. Thus, a promising direction is to incorporate trust mechanisms into the decentralized FL scheme Kairouz et al. [2019], such as blockchain frameworks Qin et al. [2022].

References

Durmus Alp Emre Acar, Yue Zhao, Ruizhao Zhu, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. Debiasing model updates for improving personalized federated training. In *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2021. URL https://proceedings.mlr.press/v139/acar21a.html.

Mohammed Adnan, Shivam Kalra, Jesse C. Cresswell, Graham W. Taylor, and Hamid R. Tizhoosh. Federated learning and differential privacy for medical image analysis. *Scientific reports*, 12(1): 1–10, 2022.

Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

Fengwen Chen, Guodong Long, Zonghan Wu, Tianyi Zhou, and Jing Jiang. Personalized federated learning with graph. *arXiv preprint arXiv:2203.00829*, 2022.

Luca Corinzia, Ami Beuret, and Joachim M Buhmann. Variational federated multi-task learning. *arXiv preprint arXiv:1906.06268*, 2019.

Sen Cui, Jian Liang, Weishen Pan, Kun Chen, Changshui Zhang, and Fei Wang. Collaboration equilibrium in federated learning, 2021. URL https://arxiv.org/abs/2108.07926.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1): 1–22, 1977.

Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning, 2020. URL https://arxiv.org/abs/2003.13461.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3557–3568. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/24389bfe4fe2eba8bf9aa9203a44cdad-Paper.pdf.

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19586–19597. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/e32cc80bf07915058ce90722ee17bb71-Paper.pdf.

Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models, 2020. URL https://arxiv.org/abs/2002.05516.

Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35-9, pages 7865–7873, 2021.

Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning, 2019.

Shivam Kalra, Junfeng Wen, Jesse C. Cresswell, Maksims Volkovs, and Hamid R. Tizhoosh. Proxyfl: Decentralized federated learning through proxy model sharing. *arXiv preprint arXiv:2111.11343*, 2021.

Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. Adaptive gradient-based meta-learning methods. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/f4aa0dd960521e045ae2f20621fb4ee9-Paper.pdf`.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. Survey of personalization techniques for federated learning. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 794–797. IEEE, 2020.

Chengxi Li, Gang Li, and Pramod K Varshney. Decentralized federated learning via mutual knowledge transfer. *IEEE Internet of Things Journal*, 2021a.

Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, pages 6357–6368. PMLR, 2021b.

Guodong Long, Yue Tan, Jing Jiang, and Chengqi Zhang. Federated learning for open banking. In *Federated learning*, pages 240–254. Springer, 2020.

Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

Othmane Marfoq, Giovanni Neglia, Aurélien Bellet, Laetitia Kameni, and Richard Vidal. Federated multi-task learning under a mixture of distributions. *Advances in Neural Information Processing Systems*, 34, 2021.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

Zhen Qin, Shuiguang Deng, Xueqiang Yan, Schahram Dustdar, and Albert Y Zomaya. Secure and efficient decentralized federated learning with data representation protection. *arXiv preprint arXiv:2205.10568*, 2022.

Adam Sadilek, Luyang Liu, Dung Nguyen, Methun Kamruzzaman, Stylianos Serghiou, Benjamin Rader, Alex Ingerman, Stefan Mellem, Peter Kairouz, Elaine O. Nsoesie, Jamie MacFarlane, Anil Vullikanti, Madhav Marathe, Paul Eastham, John S. Brownstein, Blaise Aguera y. Arcas, Michael D. Howell, and John Hernandez. Privacy-first health research with federated learning. *npj Digital Medicine*, 4(1), September 2021. doi: 10.1038/s41746-021-00489-2. URL `https://doi.org/10.1038/s41746-021-00489-2`.

Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 32(8):3710–3722, 2020.

Tao Shen, Jie Zhang, Xinkang Jia, Fengda Zhang, Gang Huang, Pan Zhou, Kun Kuang, Fei Wu, and Chao Wu. Federated mutual learning. *arXiv preprint arXiv:2006.16765*, 2020.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in neural information processing systems*, 30, 2017.

Canh T Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33:21394–21405, 2020.

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

Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5018–5027, 2017.

Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M. Alvarez. Personalized federated learning with first order model optimization. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=ehJqJQk9cw`.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated Learning with Non-IID Data, 2018. URL `https://arxiv.org/abs/1806.00582`.

A Derivations

Variational lower bound Here we derive the variational lower bound Eq. (2) for the log-likelihood objective Eq. (1). For each $i \in [K]$,

$$\log \sum_{z_i=1}^{K} p(D_i, z_i; \Theta) = \log \sum_{z_i=1}^{K} q(z_i) \cdot \frac{p(D_i, z_i; \Theta)}{q(z_i)} \tag{9}$$

$$= \log \mathbb{E}_{q(z_i)} \left[\frac{p(D_i, z_i; \Theta)}{q(z_i)} \right] \tag{10}$$

$$\geq \mathbb{E}_{q(z_i)} \left[\log \frac{p(D_i, z_i; \Theta)}{q(z_i)} \right] \tag{11}$$

$$= \mathbb{E}_{q(z_i)} \left[\log p(D_i, z_i; \Theta) \right] - \mathbb{E}_{q(z_i)} [\log q(z_i)], \tag{12}$$

where q is an alternative distribution, the inequality is due to Jensen's Inequality and the last term $\mathbb{E}_{q(z_i)}[\log q(z_i)]$ is constant independent of the parameter Θ.

Derivations of the EM steps Given the assumptions in the main text about $p_i(y|\mathbf{x})$ and $p_i(\mathbf{x})$, we know that

$$-\log p(D_i | z_i = j; \Phi) = \sum_{s=1}^{n_i} \ell(h_{\boldsymbol{\phi}_j}(\mathbf{x}_s)^{(i)}, y_s^{(i)}) - \log p(\mathbf{x}_s^{(i)}) + c. \tag{13}$$

- **E-step:** Find the best q for each client given the current parameters $\Theta^{(t-1)}$:

$$w_{ij}^{(t)} := q^{(t)}(z_i = j) = p(z_i = j | D_i; \Theta^{(t-1)}) \tag{14}$$

$$= \frac{p(z_i = j | \Pi^{(t-1)}) \cdot p(D_i | z_i = j; \Phi^{(t-1)})}{\sum_{j'=1}^{K} p(z_i = j' | \Pi^{(t-1)}) \cdot p(D_i | z_i = j'; \Phi^{(t-1)})} \tag{15}$$

$$= \frac{\Pi_{ij}^{(t-1)} \cdot p(D_i | z_i = j; \Phi^{(t-1)})}{\sum_{j'=1}^{K} \Pi_{ij'}^{(t-1)} \cdot p(D_i | z_i = j'; \Phi^{(t-1)})} \tag{16}$$

$$\propto \Pi_{ij}^{(t-1)} \exp \left[-\sum_{s=1}^{n_i} \ell \left(h_{\boldsymbol{\phi}_j^{(t-1)}}(\mathbf{x}_s^{(i)}),\ y_s^{(i)} \right) \right]. \tag{17}$$

  Then the variational lower bound becomes

$$\mathcal{L}(q^{(t)}, \Theta) = \frac{1}{n} \sum_i \sum_j w_{ij}^{(t)} \cdot \log p(D_i, z_i = j; \Theta) + C \tag{18}$$

$$= \frac{1}{n} \sum_i \sum_j w_{ij}^{(t)} \cdot (\ \log p(z_i = j; \Pi) + \log p(D_i | z_i = j; \Phi)\ ) + C \tag{19}$$

$$= \frac{1}{n} \sum_i \sum_j w_{ij}^{(t)} \cdot (\ \log \Pi_{ij} + \log p(D_i | z_i = j; \Phi)\ ) + C. \tag{20}$$

- **M-step:** Given the posterior $w_{ij}^{(t)}$ from the E-step, we need to maximize $\mathcal{L}$ w.r.t. $\Theta = (\Phi, \Pi)$. For the priors Π, we can optimize each row i of Π individually since they are decoupled in Eq. (20). Note that each row of Π is also a probability distribution, so the optimum solution is given by $\Pi_{ij}^{(t)} = w_{ij}^{(t)}$. This is because the first term of Eq. (20) for each i is the negative cross entropy, which is maximized when Π_{ij} matches $w_{ij}^{(t)}$.
  Optimizing Eq. (20) w.r.t. Φ gives

$$\Phi^{(t)} \in \operatorname*{argmax}_{\Phi} \mathcal{L}(q^{(t)}, \Theta) = \operatorname*{argmin}_{\Phi} \frac{1}{n} \sum_{i=1}^{K} \sum_{j=1}^{K} w_{ij}^{(t)} \sum_{s=1}^{n_i} \ell \left(h_{\boldsymbol{\phi}_j}(\mathbf{x}_s^{(i)}),\ y_s^{(i)} \right). \tag{21}$$

Posterior and accumulative loss Here we show an alternative implementation for Eq. (3) using accumulative loss. To shorten notations, let $\ell_{ij}^{(t)} := \sum_{s=1}^{n_i} \ell \left(h_{\boldsymbol{\phi}_j^{(t)}}(\mathbf{x}_s^{(i)}),\ y_s^{(i)} \right)$. Combining Eq. (3)

and Eq. (4) gives

$$w_{ij}^{(t)} = p(z_i = j | D_i; \Theta^{(t-1)}) \tag{22}$$

$$\propto w_{ij}^{(t-1)} \exp\left[-\ell_{ij}^{(t-1)}\right] \tag{23}$$

$$\propto w_{ij}^{(t-2)} \exp\left[-\left(\ell_{ij}^{(t-2)} + \ell_{ij}^{(t-1)}\right)\right]. \tag{24}$$

We can see that it is accumulating the losses of previous models (e.g., $\boldsymbol{\phi}_j^{(t-2)}$, $\boldsymbol{\phi}_j^{(t-1)}$ and so on) inside the exponential. Therefore, assuming the uniform prior $\Pi_{ij}^{(0)} = 1/K, \forall j$, $w^{(t)}$ is the softmax transformation of the negative of the accumulative loss $L_{ij}^{(t)} := \sum_{\tau=1}^{t-1} \ell_{ij}^{(\tau)}$ up until round t.

B Related work for personalized FL

Meta-learning Federated learning can be interpreted as a meta-learning problem, where the goal is to extract a global meta-model based on data from several clients. This meta-model can be learned using, for instance, the well-known Federated Averaging (FedAvg) algorithm [McMahan et al., 2017], and personalization can then be achieved by locally fine-tuning the meta-model [Jiang et al., 2019]. Later studies explored methods to learn improved meta-models. Khodak et al. [2019] proposed ARUBA, a meta-learning algorithm based on online convex optimization, and demonstrates that it can improve upon FedAvg's performance. Per-FedAvg [Fallah et al., 2020] uses the Model Agnostic Meta-Learning (MAML) framework to build the initial meta-model. However, MAML requires computing or approximating the Hessian term and can therefore be computationally prohibitive. Acar et al. [2021] adopted gradient correction methods to explicitly de-bias the meta-model from the statistical heterogeneity of client data and achieved sample-efficient customization of the meta-model.

Model regularization / interpolation Several works improve personalization performance by regularizing the divergence between the global and local models [Hanzely and Richtárik, 2020, Li et al., 2021b, Huang et al., 2021]. Similarly, PFedMe [T Dinh et al., 2020] formulates personalization as a proximal regularization problem using Moreau envelopes. FML [Shen et al., 2020] adopts knowledge distillation to regularize the predictions between local and global models and handle model heterogeneity. In recent work, SFL Chen et al. [2022] also formulates the personalization as a bi-level optimization problem with an additional regularization term on the distance between local models and its neighbor models according to a connection graph. Specifically, SFL adopts GCN to represent the connection graph and learns the graph as part of the optimization to encourage useful client collaborations. Introduced in Mansour et al. [2020] as one of the three methods for achieving personalization in FL, model interpolation involves mixing a client's local model with a jointly trained global model to build personalized models for each client. Deng et al. [2020] further derives generalization bounds for mixtures of local and global models.

Multi-task learning Personalized FL naturally fits into the multi-task learning (MTL) framework. MOCHA [Smith et al., 2017] utilizes MTL to address both systematic and statistical heterogeneity but is restricted to simple convex models. VIRTUAL [Corinzia et al., 2019] is a federated MTL framework for non-convex models based on a hierarchical Bayesian network formed by the central server and the clients, and inference is performed using variational methods. SPO [Cui et al., 2021] applies Specific Pareto Optimization to identify the optimal collaborator sets and learn a hypernetwork for all clients. While also aiming to identify necessary collaborators, SPO adopts a centralized FL setting with clients jointly training the hypernetwork. In contrast, our work focuses on decentralized FL where clients aggregate updates from collaborators, and jointly make predictions with them.

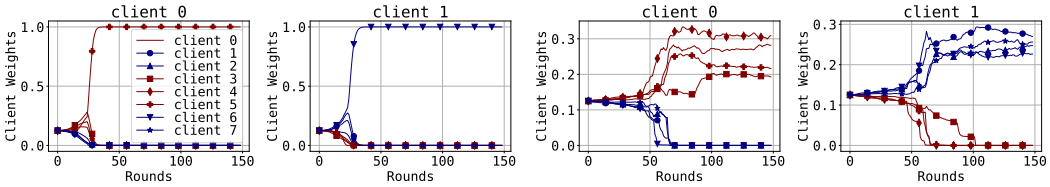
In a similar spirit to our work, Marfoq et al. [2021] assumes that the data distribution of each client is a mixture of several underlying distributions/components. Federated MTL is then formulated as a problem of modeling the underlying distributions using Federated Expectation-Maximization (FedEM). Clients jointly update a set of several component models, and each maintains a customized set of weights, corresponding to the mixing coefficients of the underlying distributions, for predictions. One shortcoming of FedEM is that it uses an instance-level weight assignment in training time but a client-level weight assignment in inference time. As a concrete example, consider a client consisting of a 20%/80% data mixture from distributions A and B. FedEM will learn two models, one for each distribution. Given a new data point at inference time, the client will always predict $0.2 \cdot \text{pred}_A + 0.8 \cdot \text{pred}_B$, *regardless of whether it came from distribution A or B*. This is caused by

the mismatched behaviour between training and inference time. On the contrary, FedeRiCo naturally considers a client-level weight assignment for both training and inference in a decentralized setting.

Other approaches Clustering-based approaches are also popular for personalized FL [Sattler et al., 2020, Ghosh et al., 2020, Mansour et al., 2020]. Such personalization lacks flexibility since each client can only collaborate with other clients within the same cluster. FedFomo [Zhang et al., 2021] interpolates the model updates of each client with those of other clients to improve local performance. FedPer [Arivazhagan et al., 2019] divides the neural network model into base and personalization layers. Base layers are trained jointly, whereas personalization layers are trained locally.

C Additional experimental results

C.1 Effect of using exponential moving average loss



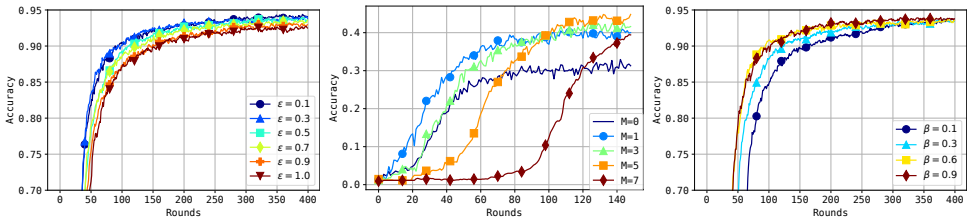
(a) Client weights with accumulative loss. (b) Client weights with exponential moving average.

Figure 5: Effect on client weights with different implementations. The client weights on CIFAR-10 with 2 different client distributions are reported.

Here, we visualize the effect of using the exponential moving average loss by plotting client weights with both accumulative loss and exponential moving average loss in Fig. 5[3]. We observe that with the accumulative loss in Fig. 5a, the client weights quickly converge to one-hot, while with the exponential moving average loss in Fig. 5b, the client weights are more distributed to similar clients. This corresponds to our expectation stated in Section 2.2: the clients using exponential moving average loss are expected to seek for more collaboration compared to using accumulative loss.

C.2 Hyperparameter sensitivity Study

In this section, we explore the effect of hyperparameters of our proposed FedeRiCo.



(a) ϵ for sampling. (b) # of sampled neighbors. (c) Momentum β.

Figure 6: Test accuracy with different hyperparameters.

Effect of ϵ-greedy sampling Here we show the effect of different ϵ values. Recall that each client deploys an ϵ-greedy selection strategy. The smaller the value of ϵ, the more greedy the client is in selecting the most relevant collaborators with high weights, leading to less exploration. Fig. 6a shows the accuracy along with training rounds with different ϵ values on the Office-Home dataset. One can see that there is a trade-off between exploration and exploitation. If ϵ is too high (e.g., $\epsilon = 1$, uniform sampling), then estimates of the likelihoods/losses are more accurate. However, some gradient updates will vanish because the client weight is close to zero (see Section 2.2), resulting in slow convergence. On the other hand, if ϵ is too small, the client may miss some important collaborators due to a lack of exploration. As a result, we use a moderate $\epsilon = 0.3$ in all of our experiments.

[3]We used uniform sampling for Fig. 5a ($\epsilon = 1$) as most of the client weights are 0 after a few rounds.

Effect of number of sampled neighbors We plot accuracy with number of neighbors $M \in \{0, 1, 3, 5, 7\}$ on CIFAR100 with 4 different client distributions, where $M = 0$ is similar to Local Training as no collaboration happens. As shown in Fig. 6b, when the number of neighbors increases, FedeRiCo converges more slowly as each client is receiving more updates on other client's models. While a smaller number of neighbors seems to have a lower final accuracy, we notice that even with $M = 1$, we still observe significant improvement compared to no collaboration. Therefore, we use $M = 3$ neighbors in our experiments as it has reasonable performance and communication cost.

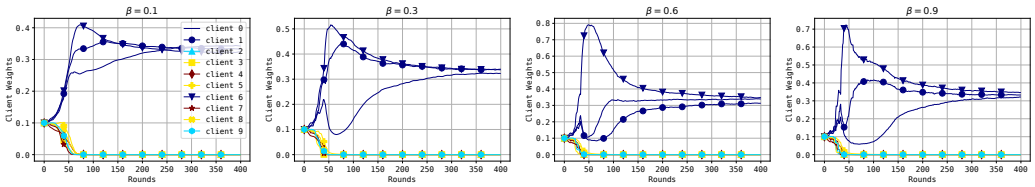


Figure 7: Client weights (client 0) with different momentum values β on the client weight update.

Effect of client weight momentum We plot the overall test accuracy of client 0 on the Office-Home dataset with 4 different data distributions over $\beta \in \{0.1, 0.3, 0.6, 0.9\}$ in Fig. 6c and similarly for the client weights in Fig. 7. With smaller β, as shown in Fig. 7, we observe a smoother update on the client weights, which is expected as the old tracking loss dominates the new one. Although various values produce similar final client weights, a bigger β can lead to more drastic changes in early training. However, one shouldn't pick a very small β just because it can produce smoother weights. As shown in Fig. 6c, the algorithm may converge more slowly with smaller β. Therefore, we use $\beta = 0.6$ as it encourages smoother updates and also maintains good convergence speed.

C.3 Client collaboration

Here we include more client weight plots of our proposed FedeRiCo on CIFAR100 with four client distributions using different data partition and training seeds. As shown in Fig. 8 and Fig. 9, clients from the same distribution collaborates has more client weights and more collaboration.

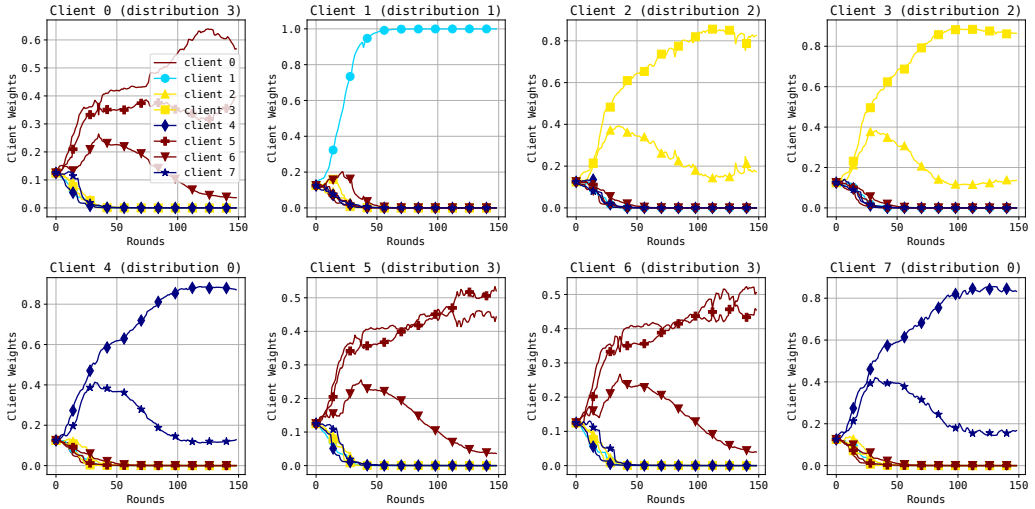


Figure 8: Client weights over time of FedeRiCo with CIFAR100 data and four different client distributions. Clients are color coded by their private data's distribution.

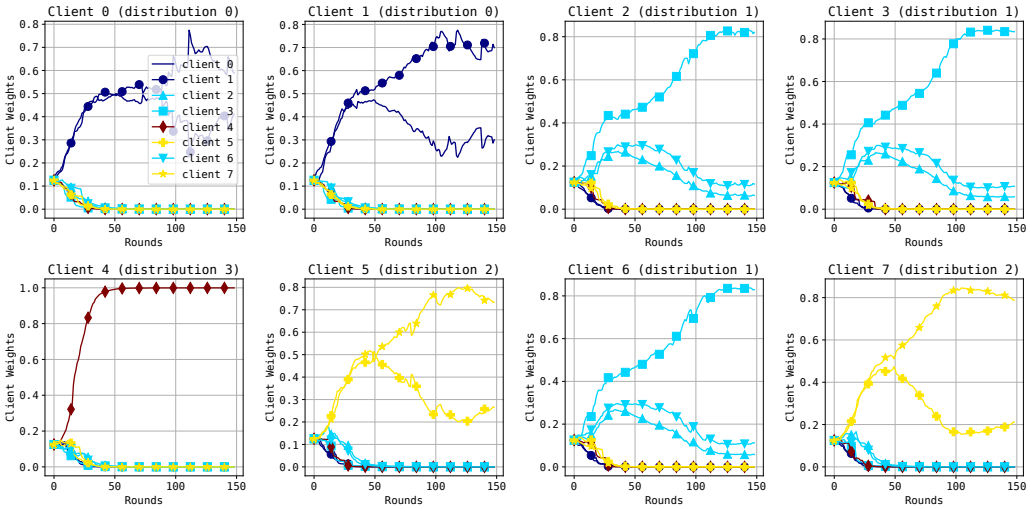


Figure 9: Client weights over time of FedeRiCo with CIFAR100 data and four different client distributions. Clients are color coded by their private data's distribution.

D Additional Experiment Details

Dataset To speed up training, we take 10%, and 15% of the training data from CIFAR-10, and CIFAR-100 respectively. For the Office-Home dataset, we merge images from all domains to get the training dataset, and use the features extracted from the penultimate layer of ResNet-18 pretrained on ImageNet.

Models and Methods For CIFAR-10, we use the CNN2 from Shen et al. [2020] with three 3x3 convolution layers (each with 128 channels followed with 2x2 max pooling and ReLu activation) and one FC layer. For CIFAR-100, we use ResNet-18 as in Marfoq et al. [2021]. For Office-Home, the model is an MLP with two hidden layers (1000 and 200 hidden units). The batch size is 50 for CIFAR, and 100 for Office-Home. For FedFomo, we use 5 local epochs in CIFAR-100 to adapt to the noisiness of training and 1 local epoch per communication round for all other experiments.

Settings CIFAR-10 results are reported across 5 different data splits and 3 different training seeds for each data split. CIFAR-100 and Office-Home results are reported across 3 different data splits with a different training seed for each split.

Training settings We optimize all models using Adam with learning rate 0.01. CIFAR experiments use 8 clients and 150 training rounds, while Office-Home experiments use 10 clients and 400 rounds.

Computational resources and software We summarize the computational resources used for the experiments in Table 2 and software versions in Table 3.

Table 2: Summary of computational resource

Operating System	Memory	CPU	GPU
Ubuntu 18.04.5	700GB	Intel(R) Xeon(R) Platinum 8168@2.70GHz	8 Tesla V100-SXM2

Table 3: Software versions

Python	Pytorch	mpi4py
3.9	1.9.0	3.1.2

E The FedeRiCo algorithm

Algorithm 1 describes our proposed FedeRiCo algorithm.

Algorithm 1: FedeRiCo: Federating with the Right Collaborators

Input: Client local datasets $\{D_i\}_{i=1}^K$, number of communication rounds r, number of neighbors M, ϵ-greedy sampling probability ϵ, momentum for exponential moving average loss tracking β, learning rate η.

Output: Client models $\{\boldsymbol{\phi}_i\}_{i=1}^K$ and client weights w_{ij}.

```
// Initialization
```

Randomly initialize $\{\boldsymbol{\phi}_i\}_{i=1}^K$;

for *client* C_i *in* $\{C_i\}_{i=1}^K$ **do**

  Initialize $\widehat{L}_{ij}^{(0)} = 0, \ell_{ij}^{(0)} = 0, w_{ij}^{(0)} = \frac{1}{K}$;

end

for *iterations* $t = 1 \dots T$ **do**

  for *client* C_i *in* $\{C_i\}_{i=1}^K$ **do**

    Sample M neighbors of this round B^t according to ϵ-greedy selection w.r.t. $w_{ij}^{(t-1)}$;

    Send $\boldsymbol{\phi}_i$ to other clients that sampled C_i;

    Receive $\boldsymbol{\phi}_j$ from sampled neighbors B^t;

    `// E-step`

    $\ell_{ij}^{(t)} = \ell_{ij}^{(t-1)}$; `// Keep the loss from previous round`

    for *b in* B^t **do**

      $\ell_{ib}^{(t)} = \sum_{s=1}^{n_i} \ell\left(h_{\boldsymbol{\phi}_b^{(t)}}(\mathbf{x}_s^{(i)}),\ y_s^{(i)}\right)$; `// Update the sampled ones`

    end

    $\widehat{L}_{ij}^{(t)} = (1-\beta)\widehat{L}_{ij}^{(t-1)} + \beta\ell_{ij}^{(t)}$; `// Update exponential moving averages`

    $w_{ij}^{(t)} = \frac{\exp(-\widehat{L}_{ij}^{(t)})}{\sum_{j'=1}^{K}\exp(-\widehat{L}_{ij'}^{(t)})}$;

    `// M-step`

    for C_b *in* B^t **do**

      `// Could also do multiple gradient steps instead`

      Compute and send $\mathbf{g}_{bi} = w_{ib}^{(t)} \nabla_{\boldsymbol{\phi}_b} \sum_{s=1}^{n_i} \ell\left(h_{\boldsymbol{\phi}_b}(\mathbf{x}_s^{(i)}),\ y_s^{(i)}\right)$ to C_b;

    end

    for C_j *that sampled* C_i **do**

      Receive $\mathbf{g}_{ij} = w_{ji}^{(t)} \sum_{s=1}^{n_j} \nabla_{\boldsymbol{\phi}_i} \ell\left(h_{\boldsymbol{\phi}_i}(\mathbf{x}_s^{(j)}),\ y_s^{(j)}\right)$;

    end

    $\boldsymbol{\phi}_i^t = \boldsymbol{\phi}_i^{(t-1)} - \eta \sum_j \mathbf{g}_{ij}$; `// Or any other gradient-based method`

  end

end

F Convergence Proof

We adapt assumptions 2 to 7 of Marfoq et al. [2021] to our setting as follows:

Assumption F.1. $\forall i \in [K], p_i(x) = p(x)$.

Assumption F.2. *The conditional probability $p_i(y|x)$ satisfies*

$$-\log p_i(y|x) = \ell(h_{\boldsymbol{\phi}_i^*}(x), y) + c, \tag{25}$$

for some parameters $\boldsymbol{\phi}_i^* \in \mathbb{R}^d$, loss function $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^+$ and normalization constant c.

Let $f(\Phi, \Pi) := \frac{1}{n} \log p(D; \Phi, \Pi)$ be the log-likelihood objective as in Eq. (1).

Assumption F.3. *f is bounded below by $f^* \in \mathbb{R}$.*

Assumption F.4 (Smoothness and bounded gradient)**.** *For all x, y, the function $\boldsymbol{\phi} \mapsto \ell(h_{\boldsymbol{\phi}}(x), y)$ is L-smooth, twice continuously differentiable and has bounded gradient: there exists $B < \infty$ such that $\|\nabla_{\boldsymbol{\phi}} \ell(h_{\boldsymbol{\phi}}(x), y)\| \leq B$.*

Assumption F.5 (Unbiased gradients and bounded variance)**.** *Each client $i \in [K]$ can sample a random batch ξ and compute an unbiased estimator $\mathbf{g}_i(\boldsymbol{\phi}, \xi)$ of the local gradient with bounded variance, i.e., $\mathbb{E}_\xi[\mathbf{g}_i(\boldsymbol{\phi}, \xi)] = \frac{1}{n_i} \sum_{s=1}^{n_i} \nabla \ell(h_{\boldsymbol{\phi}}(\mathbf{x}_i^{(s)}), y_i^{(s)})$ and $\mathbb{E}_\xi \|\mathbf{g}_t(\boldsymbol{\phi}, \xi) - \frac{1}{n_i} \sum_{s=1}^{n_i} \nabla \ell(h_{\boldsymbol{\phi}}(\mathbf{x}_i^{(s)}), y_i^{(s)})\| \leq \sigma^2$.*

Assumption F.6 (Bounded dissimilarity)**.** *There exist β and G such that any set of weights $\boldsymbol{\gamma} \in \Delta^K$:*

$$\sum_{i=1}^{K} \frac{n_i}{n} \left\| \frac{1}{n_i} \sum_{s=1}^{n_i} \sum_{j=1}^{K} \gamma_j \nabla \ell(h_{\boldsymbol{\phi}}(\mathbf{x}_i^{(s)}), y_i^{(s)}) \right\|^2 \leq G^2 + \beta^2 \left\| \frac{1}{n} \sum_{i=1}^{K} \sum_{s=1}^{n_i} \sum_{j=1}^{K} \gamma_j \nabla \ell(h_{\boldsymbol{\phi}}(\mathbf{x}_i^{(s)}), y_i^{(s)}) \right\|^2. \tag{26}$$

Theorem 2.1. *[Convergence] Under Assumptions F.1-F.6, when the clients use SGD with learning rate $\eta = \frac{a_0}{\sqrt{T}}$, and after sufficient rounds T, the iterates of our algorithm satisfy*

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E} \|\nabla_\Phi f(\Phi^t, \Pi^t)\|_F^2 \leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right), \qquad \frac{1}{T} \sum_{t=1}^{T} \Delta_\Pi f(\Phi^t, \Pi^t) \leq \mathcal{O}\left(\frac{1}{T^{3/4}}\right), \tag{7}$$

where the expectation is over random batches and $\Delta_\Pi f(\Phi^t, \Pi^t) := f(\Phi^t, \Pi^t) - f(\Phi^t, \Pi^{t+1}) \geq 0$.

Proof: At a high level, we apply the generic convergence result from Marfoq et al. [2021, Thm.3.2'] for the proof. Whereas other conditions can be easily verified, we need to find *partial first-order surrogates* [Marfoq et al., 2021, Def.1] g_i and g for f_i and f, respectively, where

$$f_i(\Theta) = f_i(\Phi, \pi_i) := -\frac{1}{n_i} \log p(D_i|\Phi, \pi_i) = -\frac{1}{n_i} \sum_{s=1}^{n_i} \log p(x_i^{(s)}, y_i^{(s)}|\Phi, \pi_i), \tag{27}$$

is the local objective function. In the following, we will verify that

$$g_i^{(t)}(\Phi, \Pi) := g_i^{(t)}(\Phi, \boldsymbol{\pi}_i) \tag{28}$$

$$:= \frac{1}{n_i} \sum_{s=1}^{n_i} \sum_{j=1}^{K} q_j^{(t)} \left[\ell\left(h_{\boldsymbol{\phi}_j}(x_i^{(s)}), y_i^{(s)}\right) - \log p_j(x_i^{(s)}) - \log \pi_{ij} + \log q_j^{(t)} - c \right], \tag{29}$$

$$g^{(t)}(\Phi, \Pi) := \sum_{i=1}^{K} \frac{n_i}{n} g_i^{(t)}(\Phi, \boldsymbol{\pi}_i), \tag{30}$$

satisfy the three conditions of partial first-order surrogates near $(\Phi^{(t-1)}, \Pi^{(t-1)})$: (similarly defined for $g^{(t)}$ and f)

1. $g_i^{(t)}(\Phi, \Pi) \geq f_i(\Phi, \Pi), \forall t, \Phi, \Pi$;
2. $r_i^{(t)}(\Phi, \Pi) := g_i^{(t)}(\Phi, \Pi) - f_i(\Phi, \Pi)$ is differentiable and $\widetilde{L}$-smooth w.r.t. Φ (for some $\widetilde{L} < \infty$). Moreover, $r_i^{(t)}(\Phi^{(t-1)}, \Pi^{(t-1)}) = 0$ and $\nabla_\Phi r_i(\Phi^{(t-1)}, \Pi^{(t-1)}) = \mathbf{0}$;

3. $g_i^{(t)}(\Phi, \Pi^{(t-1)}) - g_i(\Phi, \Pi) = \mathsf{d}(\Pi^{(t-1)}, \Pi)$ for all Φ and $\Pi \in \operatorname{argmin}_{\Pi'} g(\Phi, \Pi')$ where d is non-negative and $\mathsf{d}(\Pi, \Pi') = 0$ iff $\Pi = \Pi'$.

To simplify notations, define the following (the dependency on round t is ignored when it is clear from context)

$$q_j := q_i(z_i = j), \tag{31}$$

$$\mathcal{L}_j := \sum_{s=1}^{n_i} \ell\left(h_{\boldsymbol{\phi}_j}(x_i^{(s)}), y_i^{(s)}\right), \tag{32}$$

$$\gamma_j := p_i(z_i = j | D_i, \Phi, \boldsymbol{\pi}_i). \tag{33}$$

(1) To start verifying the first condition,

$$g_i(\Phi, \boldsymbol{\pi}_i) = \frac{1}{n_i} \sum_{s=1}^{n_i} \sum_{j=1}^{K} q_j \left[\ell\left(h_{\boldsymbol{\phi}_j}(x_i^{(s)}), y_i^{(s)}\right) - \log p_j(x_i^{(s)}) - \log \pi_{ij} + \log q_j - c\right] \tag{34}$$

$$= \frac{1}{n_i} \sum_{s=1}^{n_i} \sum_{j} q_j \left[-\log\left(p_j(y_i^{(s)} | x_i^{(s)}, \boldsymbol{\phi}_j) \cdot p_j(x_i^{(s)}) \cdot p_i(z_i = j)\right) + \log q_j\right] \tag{35}$$

$$= \frac{1}{n_i} \sum_{s=1}^{n_i} \sum_{j} q_j \left[-\log p_i\left(x_i^{(s)}, y_i^{(s)}, z_i = j \middle| \Phi, \boldsymbol{\pi}_i\right) + \log q_j\right] \tag{36}$$

$$= \frac{1}{n_i} \sum_{j} q_j \left[-\log p_i\left(D_i, z_i = j | \Phi, \boldsymbol{\pi}_i\right) + \log q_j\right]. \tag{37}$$

Then

$$r_i(\Phi, \boldsymbol{\pi}_i) = g_i(\Phi, \boldsymbol{\pi}_i) - f_i(\Phi, \boldsymbol{\pi}_i) \tag{38}$$

$$= \frac{1}{n_i} \mathcal{KL}\left(q(\cdot) \parallel p_t(\cdot | D_i, \Phi, \boldsymbol{\pi}_i) \right), \tag{39}$$

where $\mathcal{KL}$ is the KL-divergence. This verifies the first condition of partial first-order surrogates since the KL-divergence is non-negative.

(2) Now we verify the second condition. Note that r_t is twice continuously differentiable due to Assumption F.4. With Assumption F.1

$$\gamma_j = p_i(z_i = j | D_i, \Phi, \boldsymbol{\pi}_i) = \frac{\exp\left[-\mathcal{L}_{j'} + \log \pi_{ij}\right]}{\sum_{j'} \exp\left[-\mathcal{L}_{j'} + \log \pi_{ij'}\right]}, \tag{40}$$

$$\nabla_{\boldsymbol{\phi}_{j'}} \gamma_j = \begin{cases} (-\gamma_j + \gamma_j^2) \nabla \mathcal{L}_j & \text{if } j' = j \\ \gamma_j \gamma_{j'} \nabla \mathcal{L}_{j'} & \text{if } j' \neq j, \end{cases} \tag{41}$$

where $\nabla \mathcal{L}_j$ is shorthand for $\nabla_{\boldsymbol{\phi}_j} \mathcal{L}_j$. Then

$$\nabla_{\boldsymbol{\phi}_{j'}} r_i = \frac{1}{n_i} \nabla_{\boldsymbol{\phi}_{j'}} \sum_j (-q_j \log \gamma_j) \qquad \text{Definition of } \mathcal{KL} \tag{42}$$

$$= \frac{1}{n_i} \sum_j \left(-\frac{q_j}{\gamma_j} \nabla_{\boldsymbol{\phi}_{j'}} \gamma_j\right) \tag{43}$$

$$= \frac{1}{n_i} \left[q_{j'}(1 - \gamma_{j'}) - \sum_{j \neq j'} q_j \gamma_{j'}\right] \nabla \mathcal{L}_{j'} \qquad \text{When } j = j' \text{ vs } j \neq j' \tag{44}$$

$$= \frac{1}{n_i} \left[q_{j'}(1 - \gamma_{j'}) - (1 - q_{j'}) \gamma_{j'}\right] \nabla \mathcal{L}_{j'} \qquad \sum_j q_j = 1 \tag{45}$$

$$= \frac{1}{n_i} (q_{j'} - \gamma_{j'}) \nabla \mathcal{L}_{j'}. \tag{46}$$

The Hessian of r_i, $\mathbf{H}(r_i) \in \mathbb{R}^{dK \times dK}$ w.r.t. Φ, is a block matrix, with blocks given by

$$\Big(\mathbf{H}(r_t)\Big)_{j,j'} = \begin{cases} \frac{1}{n_i}\left[(q_j - \gamma_j)\mathbf{H}(\mathcal{L}_j) + (\gamma_j - \gamma_j^2)(\nabla\mathcal{L}_j)(\nabla\mathcal{L}_j)^\top\right] \\ -\frac{1}{n_i}\gamma_j\gamma_{j'}(\nabla\mathcal{L}_j)(\nabla\mathcal{L}_{j'})^\top \qquad \text{when } j \neq j', \end{cases} \tag{47}$$

where $\mathbf{H}(\mathcal{L}_j) \in \mathbb{R}^{d\times d}$ is the Hessian of $\mathcal{L}_{\boldsymbol{\phi}_j}(D_t)$ w.r.t. $\boldsymbol{\phi}_j$. Introduce block matrices $\widetilde{\mathbf{H}}, \widehat{\mathbf{H}} \in \mathbb{R}^{dK\times dK}$ as

$$\begin{aligned} \widetilde{\mathbf{H}}_{j,j'} &= \begin{cases} \frac{1}{n_i}(\gamma_j - \gamma_j^2)(\nabla\mathcal{L}_j)(\nabla\mathcal{L}_j)^\top \\ -\frac{1}{n_i}\gamma_j\gamma_{j'}(\nabla\mathcal{L}_j)(\nabla\mathcal{L}_{j'})^\top \qquad \text{when } j \neq j', \end{cases} \\ \widehat{\mathbf{H}}_{j,j'} &= \begin{cases} \frac{1}{n_i}(q_j - \gamma_j)\mathbf{H}(\mathcal{L}_j) \\ \mathbf{0} \qquad \text{when } j \neq j'. \end{cases} \end{aligned} \tag{48}$$

Since $q_j, \gamma_j \in [0,1]$ and ℓ is L-smooth by Assumption F.4, we have $-L \cdot I_{dK} \preccurlyeq \widehat{\mathbf{H}} \preccurlyeq L \cdot I_{dK}$. Using Lemma F.7 (see below), we have $\mathbf{0} \preccurlyeq \widetilde{\mathbf{H}} \preccurlyeq B^2 \cdot I_{dK}$ (note that $\nabla\mathcal{L}_j$ is the sum of n_i individual gradients and $\mathbf{H}(r_t)$ has $1/n_i$). As a result, $-\widetilde{L} \cdot I_{dK} \preccurlyeq \mathbf{H}(r_t) \preccurlyeq \widetilde{L} \cdot I_{dK}$ (where $\widetilde{L} = L + B^2 < \infty$) and therefore r_t is $\widetilde{L}$-smooth.

Finally, $q_j^{(t)} = p_i(z_i = j | D_i, \Phi^{(t-1)}, \boldsymbol{\pi}_i^{(t-1)}), \forall t > 0$ by the algorithm, which means

$$r_i^{(t)}(\Phi^{(t-1)}, \Pi^{(t-1)}) = r_i^{(t)}(\Phi^{(t-1)}, \boldsymbol{\pi}_i^{(t-1)}) = 0. \tag{49}$$

Additionally, from Eq. (39) we know that $r_i^{(t)}(\Phi, \boldsymbol{\pi}_i)$ is a (non-negative) KL-divergence for all Φ, Π. Recall that $r_i^{(t)}$ is differentiable. It follows that $\Phi^{(t-1)}$ is a minimizer of the function $\{\Phi \mapsto r_i^{(t)}(\Phi, \boldsymbol{\pi}_i^{(t-1)})\}$ and

$$\nabla_\Phi r_i^{(t)}(\Phi^{(t-1)}, \boldsymbol{\pi}_i^{(t-1)}) = \mathbf{0}. \tag{50}$$

This verifies the second condition of the partial first-order surrogate.

(3) Note that $\boldsymbol{\pi}_i^{(t)} = \operatorname{argmin}_{\boldsymbol{\pi}} g_i^{(t)}(\Phi, \boldsymbol{\pi})$ due to the choice of $q_i^{(t)}$ by the algorithm. Then for any $\boldsymbol{\pi}_i$ and $i \in [K]$,

$$\begin{aligned} g_i^{(t)}(\Phi, \boldsymbol{\pi}_i) - g_i^{(t)}(\Phi, \boldsymbol{\pi}_i^{(t)}) &= \sum_j q_j^{(t)}(\log \pi_{ij}^{(t)} - \log \pi_{ij}) \\ &= \sum_j \pi_{ij}^{(t)}(\log \pi_{ij}^{(t)} - \log \pi_{ij}) \\ &= \mathcal{KL}(\boldsymbol{\pi}_i^{(t)} \| \boldsymbol{\pi}_i), \end{aligned} \tag{51}$$

which is non-negative and equals zero iff $\boldsymbol{\pi}_i^{(t)} = \boldsymbol{\pi}_i$. This verifies the third condition of partial first-order surrogate.

At last, g, f are convex combinations of $\{g_i\}_{i=1}^K, \{f_i\}_{i=1}^K$, respectively, thus the same properties hold between g and f. This completes the proof. ■

Lemma F.7. *Suppose* $\mathbf{g}_1, \ldots, \mathbf{g}_K \in \mathbb{R}^d$ *and* $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_K) \in \Delta^K$. *The block matrix* $\mathbf{H} \in \mathbb{R}^{dK}$*:*

$$\mathbf{H}_{j,j'} = \begin{cases} (\gamma_j - \gamma_j^2)\mathbf{g}_j\mathbf{g}_j^\top \\ -\gamma_j\gamma_{j'}\mathbf{g}_j\mathbf{g}_{j'}^\top \qquad \textit{when } j \neq j', \end{cases} \tag{52}$$

is positive semi-definite (PSD). If in addition $\|\mathbf{g}_j\| \leq B < \infty, \forall j \in [K]$*, then* $\mathbf{H} \preccurlyeq B^2 \cdot I_{dK}$

Proof: Let $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_K] \in \mathbb{R}^{dK}$, then

$$\begin{aligned}
\mathbf{x}^\top \mathbf{H} \mathbf{x} &= \sum_{j,j'=1}^{K} \mathbf{x}_j^\top \mathbf{H}_{j,j'} \mathbf{x}_j && (53)\\
&= \sum_{j=1}^{K} \left(\mathbf{x}_j^\top \mathbf{H}_{j,j} \mathbf{x}_j + \sum_{j' \neq j} \mathbf{x}_j^\top \mathbf{H}_{j,j'} \mathbf{x}_{j'} \right) && (54)\\
&= \sum_{j=1}^{K} (\gamma_j - \gamma_j)^2 \cdot (\mathbf{x}_j^\top \mathbf{g}_j)^2 - \sum_{j=1}^{K} \left(\sum_{j' \neq j} \gamma_j \gamma_{j'} \cdot (\mathbf{x}_j^\top \mathbf{g}_j) \cdot (\mathbf{x}_{j'}^\top \mathbf{g}_{j'}) \right) && (55)\\
&= \sum_{j=1}^{K} \gamma_j (1 - \gamma_j) \cdot (\mathbf{x}_j^\top \mathbf{g}_j)^2 - \sum_{j=1}^{K} \left(\gamma_j (\mathbf{x}_j^\top \mathbf{g}_j) \cdot \sum_{j' \neq j} \gamma_{j'} \cdot (\mathbf{x}_{j'}^\top \mathbf{g}_{j'}) \right) && (56)\\
&= \sum_{j=1}^{K} \gamma_j \left(\sum_{j' \neq j} \gamma_{j'} \right) \cdot (\mathbf{x}_j^\top \mathbf{g}_j)^2 - \sum_{j=1}^{K} \left(\gamma_j (\mathbf{x}_j^\top \mathbf{g}_j) \cdot \sum_{j' \neq j} \gamma_{j'} \cdot (\mathbf{x}_{j'}^\top \mathbf{g}_{j'}) \right) && (57)\\
&= \sum_{j=1}^{K} \gamma_j (\mathbf{x}_j^\top \mathbf{g}_j) \cdot \sum_{j' \neq j} \gamma_{j'} \left(\mathbf{x}_j^\top \mathbf{g}_j - \mathbf{x}_{j'}^\top \mathbf{g}_{j'} \right) && (58)\\
&= \sum_{j=1}^{K} \gamma_j (\mathbf{x}_j^\top \mathbf{g}_j) \cdot \sum_{j'=1}^{K} \gamma_{j'} \left(\mathbf{x}_j^\top \mathbf{g}_j - \mathbf{x}_{j'}^\top \mathbf{g}_{j'} \right) && (59)\\
&= \sum_{j=1}^{K} \gamma_j (\mathbf{x}_j^\top \mathbf{g}_j)^2 - \left(\sum_{j=1}^{K} \gamma_j \mathbf{x}_j^\top \mathbf{g}_j \right)^2 && (60)\\
&= \mathbb{E}_{j \sim \boldsymbol{\gamma}}[(\mathbf{x}_j^\top \mathbf{g}_j)^2] - \left(\mathbb{E}_{j \sim \boldsymbol{\gamma}}[\mathbf{x}_j^\top \mathbf{g}_j] \right)^2 && (61)\\
&= \mathbb{V}_{j \sim \boldsymbol{\gamma}}[\mathbf{x}_j^\top \mathbf{g}_j] \geq 0, && (62)
\end{aligned}$$

where we have repeatedly applied $\sum \gamma_j = 1$ and $\mathbb{E}, \mathbb{V}$ denote expectation and variance, treating $\mathbf{x}_j^\top \mathbf{g}_j$ as a random variable. As a result, $\mathbf{H}$ is PSD.

Suppose in addition $\|\mathbf{g}_j\| \leq B < \infty, \forall j \in [K]$. Using the Cauchy-Schwarz inequality, we have

$$-B \cdot \|\mathbf{x}_j\| \leq -\|\mathbf{x}_j\| \cdot \|\mathbf{g}_j\| \leq \mathbf{x}_j^\top \mathbf{g}_j \leq \|\mathbf{x}_j\| \cdot \|\mathbf{g}_j\| \leq B \cdot \|\mathbf{x}_j\|. \tag{63}$$

Since $\|\mathbf{x}_j\| \leq \|\mathbf{x}\|, \forall j \in [K]$, we have

$$-B \cdot \|\mathbf{x}\| \leq \mathbf{x}_j^\top \mathbf{g}_j \leq B \cdot \|\mathbf{x}\|. \tag{64}$$

Finally, with the Popoviciu's inequality on variances, we have

$$\mathbf{x}^\top \mathbf{H} \mathbf{x} = \mathbb{V}_{j \sim \boldsymbol{\gamma}}[\mathbf{x}_j^\top \mathbf{g}_j] \leq \frac{1}{4} (B \cdot \|\mathbf{x}\| + B \cdot \|\mathbf{x}\|)^2 = B^2 \|\mathbf{x}\|^2, \tag{65}$$

which means $\mathbf{H} \preccurlyeq B^2 I_{dK}$. ■