

# FedSoft: Soft Clustered Federated Learning with Proximal Local Updating

Yichen Ruan, Carlee Joe-Wong

Carnegie Mellon University

yichenr@andrew.cmu.edu, cjoewong@andrew.cmu.edu

## Abstract

Traditionally, clustered federated learning groups clients with the same data distribution into a cluster, so that every client is uniquely associated with one data distribution and helps train a model for this distribution. We relax this hard association assumption to soft clustered federated learning, which allows every local dataset to follow a mixture of multiple source distributions. We propose **FedSoft**, which trains both locally personalized models and high-quality cluster models in this setting. **FedSoft** limits client workload by using proximal updates to require the completion of only one optimization task from a subset of clients in every communication round. We show, analytically and empirically, that **FedSoft** effectively exploits similarities between the source distributions to learn personalized and cluster models that perform well.

## 1 Introduction

Federated learning (FL) is an innovative privacy-preserving machine learning paradigm that distributes collaborative model training across participating user devices without users' sharing their raw training samples. In the widely used federated learning algorithm **FedAvg** (McMahan et al. 2017), clients jointly train a shared machine learning model by iteratively running local updates and synchronizing their intermediate local models with a central server. In spite of its success in applications such as next word prediction (Hard et al. 2018) and learning on electronic health records (Brisimi et al. 2018), FL is known to suffer from slow model training when clients' local data distributions are heterogeneous, or non-IID (non-independently and identically distributed). In response to this challenge, some recent works propose to bypass data heterogeneity by performing *local model personalization*. Instead of pursuing one universally applicable model shared by all clients, these algorithms' training objective is to create one model for each client that fits its local data. Personalization methods include local fine tuning (Sim, Zadrazil, and Beaufays 2019), model interpolation (Mansour et al. 2020), and multi-task learning (Smith et al. 2017). In this paper, we focus on an alternative approach: *clustered federated learning*, which we generalize to train both cluster and personalized models on realistic distributions of client data.

---

Copyright © 2022, Association for the Advancement of Artificial Intelligence ([www.aaai.org](http://www.aaai.org)). All rights reserved.

Clustered FL relaxes the assumption of FL that each client has an unique data distribution; instead, it allows different clients to share one data distribution, with fewer source data distributions than clients. The objective of clustered FL is to train one model for every distribution. In traditional clustered FL, a client can only be associated with one data distribution. We thus call this method *hard clustered federated learning*. Under the hard association assumption, the non-IID problem can be easily resolved: simply group clients with the same data distribution into one cluster, then conduct conventional FL on each cluster, within which the data distribution is now IID among clients. Unlike other personalization methods, clustered FL thus produces centrally available models that can be selectively migrated to new users that are unwilling, or unable, to engage in the subsequent local adaptation process (e.g. fine tuning) due to privacy concerns or resource limitations. This convenience in model adoption is particularly valuable for the current training-testing-deployment lifecycle of FL where deployment, rather than the training itself, is the end goal (Kairouz et al. 2019).

However, hard clustered FL faces two fundamental problems in practice. First, *multiple clients may be unlikely to possess identical data distributions*. In fact, the real-world user data is more likely to follow a *mixture* of multiple distributions (Marfoq et al. 2021). E.g., if each client is a mobile phone and we wish to model its user's content preferences, we might expect the clients to be clustered into adults and children. However, adult users may occasionally view children's content, and devices owned by teenagers (or shared by parents and children) may possess large fractions of data from both distributions. Similarly, content can be naturally grouped by users' interests (e.g., genres of movies), each of which may have a distinct distribution. Data from users with multiple interests then reflects a mixture of these distributions. Since the mixture ratios can vary for different clients, they may have different overall distributions even though the source distributions are identical. Clustering algorithms like the Gaussian mixture model (Reynolds 2009) use a similar rationale. Clients may then require models personalized to their distributions to make accurate predictions on their data, in addition to the cluster models used for new users.

Hard clustered FL's second challenge is that *it cannot effectively exploit similarities between different clusters*. Though FL clients may have non-IID data distributions, two

different distributions may still exhibit some similarity, as commonly assumed in personalization works (Smith et al. 2017). For example, young people may have more online slang terms in their chatting data, but all users (generally) follow the same basic grammar rules. Thus, the knowledge distilled through the training on one distribution could be transferred to accelerate the training of others. However, in most hard clustered FL algorithms, different cluster models are trained independently, making it difficult to leverage the potential structural likeness among distributions. Note that unlike in other personalization methods where the discussion of similarity is restricted to similarities between individual clients, here we focus on the broader similarities between source cluster distributions. Thus, we can gain better insight into the general data relationship rather than just the relationships between participating clients.

To overcome clustered FL’s first challenge of the hard association assumption, in this paper, we utilize *soft clustered federated learning*. In soft clustered FL, we suppose that the data of each client follows a mixture of multiple distributions. However, training cluster models using clients with mixed data raises two new challenges. First, *the workload of clients can explode*. When all the data of a client comes from the same distribution, as in hard clustered FL, it ideally only needs to contribute towards one training task: training that distribution’s model. However, in soft clustered FL, a client has multiple data sources. A natural extension of hard clustered FL is for the client to help train all cluster models whose distributions are included in its mixture (Marfoq et al. 2021). However, the workload of participating clients then grows linearly with the number of clusters, which can be large (though typically much smaller than the number of clients) for some applications. This multiplying of client workload can make soft clustered FL infeasible, considering the resource restrictions on typical FL user devices and the long convergence time for many FL models (McMahan et al. 2017). Second, *the training of cluster models and the local personalization are distinct*. In hard clustered FL, client models are the same as cluster models since a client is uniquely bound to one cluster. In soft clustered FL, local distributions differ from individual cluster distributions, and thus training cluster models does not directly help the local personalization. Complicating things further, these local distributions and their exact relationships to the cluster models are unknown *a priori*. Combining the training of cluster and personalized models is then challenging.

To solve these two challenges, and handle the second disadvantage of hard clustered FL discussed above, we utilize the proximal local updating trick, which is originally developed in **FedProx** (Li et al. 2018) to grant clients the use of different local solvers in FL. During the course of proximal local updating, instead of working on fitting the local model to the local dataset, each client optimizes a proximal local objective function that both carries local information and encodes knowledge from all cluster models. We name this proposed algorithm **FedSoft**.

In **FedSoft**, since the fingerprints of all clusters are integrated into one optimization objective, clients only need to solve *one single optimization problem*, for which the work-

load is almost the same as in conventional FL. In addition, by combining local data with cluster models in the local objective, clients can *perform local personalization on the fly*. Eventually, the server obtains collaboratively trained cluster models that can be readily applied to new users, and each participating client gets one personalized model as a byproduct. Proximal local updating allows a cluster to *utilize the knowledge of similar distributions*, overcoming the second disadvantage of the hard clustered FL. Intuitively, with all clusters present in the proximal objective, a client can take as reference training targets any cluster models whose distributions take up non-trivial fractions of its data. These component distributions, co-existing in the same dataset, are similar by nature. Thus, a personalized local model integrating all its component distributions can in turn be utilized by the component clusters to exploit their similarities.

Our *contributions* are: We design the **FedSoft** algorithm for efficient soft clustered FL. We establish a convergence guarantee that relates the algorithm’s performance to the divergence of different distributions, and validate the effectiveness of the learned cluster and personalized models in experiments under various mixture patterns. Our results show the training of cluster models converges linearly to a remaining error determined by the cluster heterogeneity, and that **FedSoft** can outperform existing FL implementations in both global cluster models for future users and personalized local models for participating clients.

## 2 Related Works

The training objective of hard clustered FL is to simultaneously identify the cluster partitions and train a model for each cluster. Existing works generally adopt an Expectation-Maximization (EM) like algorithm, which iteratively alternates between the cluster identification and model training. Based on how the partition structure is discovered, these algorithms can be classified into four types:

The first type leverages the distance between model parameters, e.g., Xie et al. (2021) propose to determine client association based on the distances between client models and server models. Similarly, Briggs et al. (2020) suggest to apply a distance-based hierarchical clustering algorithm directly on client models. The second type determines the partition structure based on the gradient information, e.g., the **CFL** (Sattler, Müller, and Samek 2020) algorithm splits clients into bi-partitions based on the cosine similarity of the client gradients, and then checks whether a partition is congruent (i.e., contains IID data) by examining the norm of gradients on its clients. Likewise, the **FedGroup** (Duan et al. 2020) algorithm quantifies the similarity among client gradients with the so-called Euclidean distance of decomposed cosine similarity metric, which decomposes the gradient into multiple directions using singular value decomposition. The third type utilizes the training loss, e.g., in **HyperCluster** (Mansour et al. 2020), each client is greedily assigned to the cluster whose model yields the lowest loss on its local data. A generalization guarantee is provided for this algorithm. Ghosh et al. (2020) propose a similar algorithm named **IFCA**, for which a convergence bound is established under the assumption of good initialization and all clients

having the same amount of data. The fourth type uses exogenous information about the data, e.g., Huang et al. (2019) and Qayyum et al. (2021) group patients into clusters respectively based on their electronic medical records and imaging modality. This information usually entails direct access to the user data and thus cannot be applied in the general case.

Recently, Marfoq et al. (2021) propose a multi-task learning framework similar to soft clustered FL that allows client data to follow a mixture of distributions. Their proposed **FedEM** algorithm adopts an EM algorithm and estimates the mixture coefficients based on the training loss. However, **FedEM** requires every client to perform a local update for each cluster in each round, which entails significantly more training time than conventional **FedAvg**. Their analysis moreover assumes a special form of the loss function with all distributions having the same marginal distribution, which is unrealistic. In contrast, **FedSoft** requires only a subset of clients to return gradients for only one optimization task in each round. Moreover, we show its convergence for generic data distributions and loss functions.

The proximal local updating procedure that we adopt incorporates a regularization term in the local objective, which is also used for model personalization outside clustered settings. Typical algorithms include **FedAMP** (Huang et al. 2021), which adds an attention-inducing function to the local objective, and **pFedMe** (Dinh, Tran, and Nguyen 2020), which formulates the regularization as Moreau envelopes.

### 3 Formulation and Algorithm

**Mixture of distributions.** Assume that each data point at each client is drawn from *one* of the  $S$  distinct data distributions  $\mathcal{P}_1, \dots, \mathcal{P}_S$ . Similar to general clustering problems, we take  $S$  as a hyperparameter determined a priori. Data points from all clients that follow the same distribution form a *cluster*. In soft clustered FL, a client may possess data from multiple clusters. Given a loss function  $l(w; x, y)$ , the (real) *cluster risk*  $F_s(w)$  is the expected loss for data following  $\mathcal{P}_s$ :

$$F_s(w) \triangleq \mathbb{E}_{(x,y) \sim \mathcal{P}_s} [l(w; x, y)] \quad (1)$$

We then wish to find  $S$  cluster models  $c_1^* \cdots c_S^*$  such that all cluster objectives are minimized simultaneously. These cluster models will be co-trained by all clients through coordination at the central server:

$$c_s^* = \operatorname{argmin}_w F_s(w), s = 1, \dots, S \quad (2)$$

Suppose a client  $k \in [N]$  with local dataset  $\mathcal{D}_k$  has  $|\mathcal{D}_k| = n_k$  data points, among which  $n_{ks}$  data points are sampled from distribution  $\mathcal{P}_s$ . The real risk of a client can thus be written as an average of the cluster risks:

$$\begin{aligned} f_k(w_k) &\triangleq \frac{1}{n_k} \mathbb{E} \left[ \sum_{s=1}^S \sum_{(x_k^i, y_k^i) \sim \mathcal{P}_s} l(w_k; x_k^i, y_k^i) \right] \\ &= \frac{1}{n_k} \sum_{s=1}^S n_{ks} F_s(w_k) = \sum_{s=1}^S u_{ks} F_s(w_k) \end{aligned} \quad (3)$$

Here we define  $u_{ks} \triangleq n_{ks}/n_k \in [0, 1]$  as the *importance weight* of cluster  $s$  to client  $k$ . In general,  $u_{ks}$ 's are unknown

in advance and the learning algorithm attempts to estimate their values during the learning iterations. It is worth noting that while we directly work on real risks, our formulation and analysis can be easily extended to empirical risks by introducing local-global divergences as in (Li et al. 2018).

**Proximal local updating.** Since  $f_k$  is a mixture of cluster risks, minimizing (3) alone does not help solve (2). Thus, we propose each client instead optimize a proximal form of (3):

$$h_k(w_k; c^t, u^t) \triangleq f_k(w_k) + \frac{\lambda}{2} \sum_{s=1}^S u_{ks}^t \|w_k - c_s^t\|^2 \quad (4)$$

Here  $\lambda$  is a hyperparameter and  $u_{ks}^t$  denotes the estimation of  $u_{ks}$  at time  $t$ . In the local updating step, every client searches for the optimal local model  $w_k^*$  that minimizes  $h_k$  given the current global estimation of cluster models  $\{c_s^t\}$ . As in (Li et al. 2018), clients may use any local solver to optimize  $h_k$ . This design of the proximal objective entails cluster models  $\{c_s^t\}$  be shared among all clients through the server, as is usual in clustered FL (Ghosh et al. 2020). We thus alternatively call  $\{c_s^t\}$  the *centers*.

The regularization term  $\frac{\lambda}{2} \sum_{s=1}^S u_{ks}^t \|w_k - c_s^t\|^2$  in the proximal objective serves as a reference point for the local model training. It allows clients to work on their own specific dataset while taking advantage of and being guided by the globally shared knowledge of the centers. The regularization is weighted by the importance weights  $u_{ks}$ , so that a client will pay more attention to distributions that have higher shares in its data. To see why compounded regularization helps identify individual centers, assume we have a perfect guess of  $u_{ks}^t \equiv u_{ks}$ . The minimization of (4) can then be decoupled as a series of sub-optimization problems  $h_k(w_k; c^t) = \sum_{s=1}^S u_{ks} (F_s(w_k) + \frac{\lambda}{2} \|w_k - c_s^t\|^2)$ . Thus, after  $h_k$  is minimized, the sub-problems corresponding to large  $u_{ks}$  will also be approximately solved. We can hence utilize the output local model  $w_k^{t*}$  to update these centers with large  $u_{ks}$ . Moreover,  $w_k^{t*}$  trained in this manner forges all its component distributions  $\{\mathcal{D}_s | u_{ks}^t \neq 0\}$ , which may share some common knowledge. Thus, the training of these clusters are bonded through the training of their common clients, exploiting similarities between the clusters.

The output model  $w_k^{t*}$  is itself a well personalized model that leverages both local client knowledge and the global cluster information. Marfoq et al. (2021) show that under certain conditions, the optimal client model for soft clustered FL is a mixture of the optimal cluster models. The same implication can also be captured by our proximal updating formulation. When  $\sum_s u_{ks}^t = 1$ , the gradient  $\nabla h_k$  is

$$\nabla_{w_k} h_k = \nabla f_k(w_k) + \lambda \left( w_k - \sum_{s=1}^S u_{ks}^t c_s^t \right) \quad (5)$$

which implies that  $w_k^*$  should be centered on  $\sum_s u_{ks} c_s^*$ . As a result, through the optimization of all  $h_k$ , not only will the server obtain the trained cluster models, but also each client will obtain a sufficiently personalized local model.

**Algorithm design.** We formally present **FedSoft** in Algorithm 1. The first step of the algorithm is to estimate the importance weights  $\{u_{ks}^t\}$  for each client  $k$  (lines 3-14). The

---

**Algorithm 1:** FedSoft

---

```

1 Input: Global epoch  $T$ , importance weights
   estimation interval  $\tau$ , number of clients  $N$ , client
   selection size  $K$ , counter smoother  $\sigma$ ;
2 for  $t = 0, \dots, T - 1$  do
3   if  $t \bmod \tau = 0$  then
4     Server sends centers  $\{c_s^t\}$  to all clients ;
5     for each client  $k$  do
6       for each data point  $(x_k^i, y_k^i)$  do
7          $j = \operatorname{argmin}_s l(c_s^t; x_k^i, y_k^i)$  ;
8          $n_{kj}^t = n_{kj}^t + 1$  ;
9       end
10      Send  $u_{ks}^t = \max\{\frac{n_{ks}^t}{n_k}, \sigma\}$  to server ;
11    end
12  else
13    Set  $u_{ks}^t = u_{ks}^{t-1}$  ;
14  end
15  Server computes  $v_{sk}^t$  as in (6) ;
16  Server selects  $S$  sets of clients  $Sel_s^t \subset [N]$  at
   random for each cluster, where  $|Sel_s^t| = K$ , and
   each client gets selected with probability  $v_{sk}^t$  ;
17  Selected clients download  $\{c_s^t\}$ , then compute
   and report  $w_k^{t+1} = \operatorname{argmin}_{w_k} h_k(w_k; c^t, u^t)$  ;
18  Server aggregates  $c_s^{t+1} = \frac{1}{K} \sum_{k \in Sel_s^t} w_k^{t+1}$  ;
19 end

```

---

algorithm obtains them by finding the center that yields the smallest loss value for every data point belonging to that client, and counting the number of points  $\{n_{ks}^t\}$  matched to every cluster  $s$ . If a client  $k$  has no samples matched to  $s$  ( $n_{ks}^t = 0$ ), the algorithm sets  $u_{ks}^t = \sigma$ , where  $0 < \sigma \ll 1$  is a pre-defined smoothing parameter.

Once the server receives the importance weights  $\{u_{ks}^t\}$ , it computes the *aggregation weights*  $v_{sk}^t$  as follows (line 15):

$$v_{sk}^t = \frac{u_{ks}^t n_k}{\sum_{k' \in Sel_s^t} u_{k's}^t n_{k'}} \quad (6)$$

i.e., a client that has higher importance weight on cluster  $s$  will be given higher aggregation weight, and vice versa. The introduction of the smoother  $\sigma$  avoids the situation where  $\sum_k u_{ks}^t = 0$  for some cluster, which could happen in the very beginning of the training when the center does not exhibit strength on any distributions. In that case,  $v_{sk}^t = \frac{1}{N}$ , i.e., the cluster will be updated in a manner that treats all clients equally. Otherwise, since  $\sigma$  is very small, a client with  $u_{ks}^t = \sigma$  will be assigned a  $v_{sk}^t \approx 0$ , and the aggregation weights of other clients will not be affected.

Though calculating and reporting  $\{u_{ks}^t\}$  is computationally trivial compared to the actual training procedure, sending centers to all clients may introduce large communication costs. **FedSoft** thus allows the estimations of  $u_{ks}$  to be used for up to  $\tau \geq 1$  iterations (line 3). In practice, a client can start computing  $u_{ks}^t$  for a cluster before it receives all other centers, the delay of transmission is thus tolerable.

Next, relevant clients run proximal local updates to find the minimizer  $w_k^{t+1}$  for the proximal objective  $h_k^t$ , which entails solving only one optimization problem (line 17). In the case when all clients participate, the cluster models are produced by aggregating all client models:  $c_s^{t+1} = \sum_{k=1}^N v_{sk}^t w_k^{t+1}$ . However, requiring full client participation is impractical in the federated setting. We thus use the client selection trick (McMahan et al. 2017) to reduce the training cost (lines 16). For each cluster  $s$ , the algorithm randomly selects a small subset of clients  $Sel_s^t$  to participate in the local updating at time  $t$ , where  $|Sel_s^t| = K < N$ .

Clustered FL generally entails more clients to be selected compared to conventional FL, to ensure the convergence of all cluster models. Since **FedSoft** clients can contribute to multiple centers, however, we select only  $|\cup_s Sel_s^t|$  clients instead of  $\sum_s |Sel_s^t| = SK$  clients in the usual clustered FL. For example, if each distribution has the same share in every client, then in expectation only  $N \left(1 - (1 - \frac{K}{N})^S\right)$  clients will be selected. This number equals  $2K - \frac{K^2}{N}$  when  $S = 2$ , i.e.,  $\frac{K^2}{N}$  clients are selected by both clusters.

Once the server receives the local models  $\{w_k^{t+1}\}$  for selected clients, it produces the next centers by simply averaging them (line 18). After completion, the algorithm yields trained cluster models  $\{c_s^T\}$  as outputs, and each client obtains a personalized local model  $w_k^T$  as a byproduct.

## 4 Convergence Analysis

In this section, we provide a convergence guarantee for **FedSoft**. First, note that we can rewrite (4) as follows:

$$h_k(w_k; c^t) = \sum_{s, u_{ks} \neq 0} u_{ks} h_{ks}(w_k; c_s^t) \quad (7)$$

$$h_{ks}(w_k; c_s^t) \triangleq F_s(w_k) + \frac{\lambda}{2} \frac{u_{ks}^t}{u_{ks}} \|w_k - c_s^t\|^2 \quad (8)$$

Here  $h_{ks}$  is only defined for  $u_{ks} \neq 0$ , and we call optimizing each  $h_{ks}$  a *sub-problem* for client  $k$ .

Our analysis relies on the following assumptions:

**Assumption 1.** ( $\gamma_0$ -inexact solution) *Each client produces a  $\gamma_0$ -inexact solution  $w_k^{t+1}$  for the local minimization of (4):*

$$\|\nabla h_k(w_k^{t+1}; c^t)\| \leq \gamma_0 \min_s \|\nabla F_s(c_s^t)\| \quad (9)$$

**Assumption 2.** ( $\beta$ -similarity of sub-problems) *The sub-problems  $h_{ks}$  of each client  $k$  have similar optimal points:*

$$\sum_{s'} u_{ks'} \|\nabla h_{ks'}(w_{ks}^*; c_s^t)\|^2 \leq \beta \|\nabla h_{ks}(c_s^t; c_s^t)\|^2, \forall s \quad (10)$$

for some  $\beta > 0$ , where  $w_{ks}^* = \operatorname{argmin}_{w_{ks}} h_{ks}(w_{ks}; c_s^t)$ .

**Assumption 3.** (*Strong convexity and smoothness*) *Cluster risks are  $\mu_F$  strongly convex and  $L_F$  smooth.*

**Assumption 4.** (*Bounded initial error*) *At a certain time of the training, all centers have bounded distance from their optimal points. We begin our analysis at that point:*

$$\|c_s^0 - c_s^*\| \leq (0.5 - \alpha_0) \sqrt{\mu_F/L_F} \delta, \forall s \quad (11)$$

where  $0 < \alpha_0 \leq 0.5$ .

Assumption 1 assumes significant progress is made on the proximal minimization of  $h_k$ , which is a natural extension from assumptions in **FedProx** (Li et al. 2018). Assumption 2 ensures the effectiveness of the joint optimization of  $h_k$ , i.e., solving one sub-problem can help identify the optimal points of others. Intuitively, if the sub-problems are highly divergent, we would not expect that solving them together would yield a universally good solution. This assumption quantifies our previous reasoning that different distributions co-existing in one local dataset have some similarities, which is the prerequisite for local models to converge and cluster models to be able to learn from each other. Assumption 3 is standard (Ghosh et al. 2020), and Assumption 4 is introduced by Ghosh et al. (2020) in order to bound the estimation error of  $u_{ks}^t$  (Lemma 1). Note that with Assumption 3, each sub-problem  $h_{ks}$  is also  $\mu_\lambda$  strongly convex and  $L_\lambda$  smooth, where  $\mu_\lambda \geq \mu_F$ ,  $L_\lambda \geq L_F$ , and the subscript  $\lambda$  indicates they increase with  $\lambda$ .

To measure the distance of different clusters, we quantify

$$\delta \leq \|c_s^* - c_{s'}^*\| \leq \Delta, \forall s \neq s' \quad (12)$$

As we will see later, soft clustered FL performs best when  $\delta$  and  $\Delta$  are close. Intuitively, a very small  $\delta$  indicates two clusters are almost identical, and thus might be better combined into one distribution. On the other hand, a very large  $\Delta$  implies that two clusters are too divergent, making it hard for one model to acquire useful knowledge from the other.

Next, we bound  $\mathbb{E}[u_{ks}^t]$  with respect to the true  $u_{ks}$ , for which we rely on the following lemma (Ghosh et al. 2020):

**Lemma 1.** Suppose Assumptions 3 and 4 hold. Denoting by  $\mathcal{E}_t^{j,j'}$  the event that a data point  $(x_j, y_j) \sim \mathcal{P}_j$  is incorrectly classified into cluster  $j' \neq j$  at  $t$ , there exists a  $c_\epsilon$  such that

$$\mathbb{P}(\mathcal{E}_t^{j,j'}) \leq p_\epsilon \triangleq \frac{c_\epsilon}{\alpha_0^2 \delta^4} \quad (13)$$

Based on Lemma 1, we can bound  $\mathbb{E}[u_{ks}^t]$  as follows

**Theorem 1.** (Bounded estimation errors) The expectation of  $u_{ks}^t$  is bounded as

$$\mathbb{E}[u_{ks}^t] \leq (1 - p_\epsilon)u_{ks} + p'_\epsilon \quad (14)$$

Here  $p'_\epsilon = p_\epsilon + \sigma$ , and the expectation is taken over the randomness of samples.

Next, we seek to characterize each sub-problem  $h_{ks}$  at the  $\gamma_0$ -inexact solution  $w_k^{t+1}$  that approximately minimizes  $h_k$ . Intuitively,  $w_k^{t+1}$  should perform better for sub-problems with larger  $u_{ks}$ . On the other hand, if  $u_{ks} = 0$ , we generally cannot expect that  $w_k^{t+1}$  will be close to  $c_s^*$ . We summarize this intuition in Theorems 2 and 3.

**Theorem 2.** (Inexact solutions of sub-problems) If  $u_{ks} > 0$ , and Assumptions 1 to 3 hold, then

$$\|\nabla h_{ks}(w_k^t; c_s^t)\| \leq \frac{\gamma}{\sqrt{u_{ks}}} \|\nabla F_s(c_s^t)\| \quad (15)$$

where  $\gamma = \sqrt{(\gamma_0^2 + \beta)L_\lambda/\mu_\lambda}$ .

**Theorem 3.** (Divergence of local model to centers) If Assumptions 1, 3, and 4 hold, we have

$$\|w_k^{t+1} - c_s^t\| \leq r\Delta, \forall s \quad (16)$$

$$\text{where } r = \frac{\gamma S+1}{4} \sqrt{\frac{L_F}{\mu_F}} + \frac{1}{2} \sqrt{\frac{\mu_F}{L_F}} + 1.$$

Theorem 2 indicates that if the  $h_k$  is solved with high quality  $w_k^{t+1}$  (small  $\gamma_0$ ), and the sub-problems are sufficiently similar (small  $\beta$ ), then sub-problems with  $u_{ks} > 0$  can also be well solved by  $w_k^{t+1}$ . It also justifies using  $v_{sk}^t \propto u_{ks}^t$  as aggregation weights in (6). In the case  $u_{ks} = 0$ , according to Theorem 3 (which holds for any  $u_{ks}$ ), approaching  $c_s^t$  with  $w_k^{t+1}$  will introduce an error of at most  $O(\Delta)$ .

Finally, we show the convergence of  $F_s(c_s^t)$ . The following analysis does not depend on  $\tau$ ; we show how  $\tau$  affects the convergence in Appendix B.

**Theorem 4.** (Convergence of centers) Suppose Assumptions 1 to 4 hold, and define the quantities:  $n \triangleq \sum_k n_k, n_s \triangleq \sum_k u_{ks} n_k, m_s \triangleq \sum_{k, u_{ks} \neq 0} n_k, \bar{m}_s \triangleq \sum_{k, u_{ks}=0} n_k, \hat{m}_s \triangleq (1 - p_\epsilon)m_s + p'_\epsilon \sum_{k, u_{ks} \neq 0} \frac{n_k}{u_{ks}}$ . Suppose  $\lambda$  is chosen such that  $\rho \triangleq \frac{n_s - \gamma m_s}{\lambda} - \frac{(\gamma+1)L_F m_s}{\mu_\lambda \lambda} - \frac{p'_\epsilon \bar{m}_s}{2\mu_\lambda} - \frac{L_F(\gamma+1)^2 \hat{m}_s}{2\mu_\lambda^2} - \frac{4L_F(\gamma+1)^2 \bar{m}_s}{\mu_\lambda^2 \sqrt{K}} - \frac{(\gamma+1)^2 \bar{m}_s + (1-p_\epsilon)n_s + p'_\epsilon n}{\mu_\lambda \sqrt{2K}} > 0$  and denote  $R \triangleq \frac{1}{2} (\mu_\lambda + L_F) \bar{m}_s r^2 + \frac{(4L_F + \mu_\lambda) \bar{m}_s r^2}{\sqrt{K}}$ . Then we have

$$\mathbb{E}[F_s(c_s^{t+1})] - F_s(c_s^t) \leq -\frac{\rho \|\nabla F_s(c_s^t)\|^2}{(1 - p_\epsilon)n_s + p'_\epsilon n} + \frac{p'_\epsilon R \Delta^2}{(1 - p_\epsilon)n_s - p'_\epsilon(S - 2)n} \quad (17)$$

at any time  $t$ , where the expectation is taken over the selection of clients and all  $\{u_{ks}^t\}$ .

**Corollary 1.** Suppose  $F_s(c_s^0) - F_s^* = B_s$ . After  $T$  iterations,

$$\sum_{t=1}^T \frac{\rho \mathbb{E}\|\nabla F_s(c_s^t)\|^2}{(1 - p_\epsilon)n_s + p'_\epsilon n} \leq \frac{B_s}{T} + O(p_\epsilon \Delta^2) \quad (18)$$

The choices of  $\lambda$  to make  $\rho > 0$  is discussed in (Li et al. 2018). From Corollary 1, the gradient norm converges to a remaining error controlled by  $p_\epsilon$ . Intuitively, when  $c_s^t = c_s^*$ , further updating  $c_s^t$  with misclassified models will inevitably move  $c_s^t$  away from  $c_s^*$ . This bias cannot be removed unless we have a perfect guess of  $u_{ks}$ . Recall that  $p_\epsilon = O(\frac{1}{\delta^4})$ , and thus the remaining term is  $O(\frac{\Delta^2}{\delta^4})$ , which decreases as  $\Delta$  approaches  $\delta$ . Thus, **FedSoft** performs better when the divergences between clusters are more homogeneous. Note that Corollary 1 seems to imply the remaining error will explode if  $\delta \rightarrow 0$ , but Lemma 1 is only valid when  $p_\epsilon < 1$ . Thus when  $\delta$  is very small, i.e., there exist two distributions that are extremely similar, the remaining error is determined by the maximum divergence of the other distributions. Furthermore, the divergence  $\Delta$  determines the degree of non-IID of a local dataset (not among clients), which also implicitly affects the accuracy of local solutions  $\gamma_0$ . Intuitively, a larger  $\Delta$  implies it is more difficult to exactly solve a local problem involving multiple distributions, resulting in a greater  $\gamma_0$ .

To see the role of cluster heterogeneity, suppose  $\|c_1^* - c_2^*\|$  is closer than the distance of all other centers to  $c_1$ , then the misclassified samples for cluster 1 are more likely to be matched to cluster 2. Thus, cluster 2 gets more updates from data that it does not own, producing greater remaining training error that drags its center towards cluster 1. On the other hand, if the cluster divergence is homogeneous, then the effect of mis-classification is amortized among all clusters, resulting in a universally smaller remaining error.

Theorem 4 shows the convergence of cluster models  $\{c_s\}$  in terms of the cluster risks  $\{F_s\}$ . For the local models  $\{w_k\}$ , we focus on how clients integrate global knowledge into their local personalizations, which cannot be captured only with the original client risk functions  $\{f_k(w)\}$ . Thus, we are interested in the convergence performance of  $\{w_k\}$  with respect to the proximal objective  $\{h_k\}$ . Note that under Assumption 3, **FedSoft** is effectively a cyclic block coordinate descent algorithm on a jointly convex objective function of  $\{w_k\}$  and  $\{c_s\}$ , for which the convergence is guaranteed:

**Theorem 5.** (*Joint convergence of cluster and client models*) For fixed importance weights  $\tilde{u}$ , let  $w^*, c^* = \operatorname{argmin} \sum_{k=1}^N h_k(w_k; c, \tilde{u}_k)$ , and  $w^T, c^T$  be the outputs of **FedSoft**. Then  $w^T \rightarrow w^*, c^T \rightarrow c^*$  linearly with  $T$ .

## 5 Experiments

In this section, we verify the effectiveness of **FedSoft** with two base datasets under various mixture patterns. For all experiments, we use  $N = 100$  clients, and the number of samples in each client  $n_k$  is chosen uniformly at random from 100 to 200. For ease of demonstration, for every base dataset, we first investigate the mixture of  $S = 2$  distributions and then increase  $S$ . In the case with two distributions, suppose the cluster distributions are named  $\mathcal{D}_A$  and  $\mathcal{D}_B$ . We evaluate the following partition patterns:

- 10:90 partition: 50 clients have a mixture of 10%  $\mathcal{D}_A$  and 90%  $\mathcal{D}_B$ , and 50 have 10%  $\mathcal{D}_B$  and 90%  $\mathcal{D}_A$ .
- 30:70 partition: Same as above except the ratio is 30:70.
- Linear partition: Client  $k$  has  $(0.5 + k)\%$  data from  $\mathcal{D}_A$  and  $(99.5 - k)\%$  data from  $\mathcal{D}_B$ ,  $k = 0, \dots, 99$ .

We further introduce the random partition, where each client has a random mixture vector generated by dividing the  $[0, 1]$  range into  $S$  segments with  $S - 1$  points drawn from Uniform(0, 1). We use all four partitions for  $S = 2$ , and only use the random partition when  $S > 2$  for simplification. Each partition produces non-IID local distributions, i.e., clients have different local data distributions. Specifically, the 10:90 and 30:70 partitions yield 2 local distributions, while the linear and random partitions yield 100. Unless otherwise noted, we choose **FedSoft**'s estimation interval  $\tau = 2$ , client selection size  $K = 60$ , counter smoother  $\sigma = 1e-4$ , and all experiments are run until both cluster and client models have fully converged. All models are randomly initialized with the Xavier normal (Glorot and Bengio 2010) initializer without pre-training, so that the association among clients, centers, and cluster distributions is built automatically during the training process.

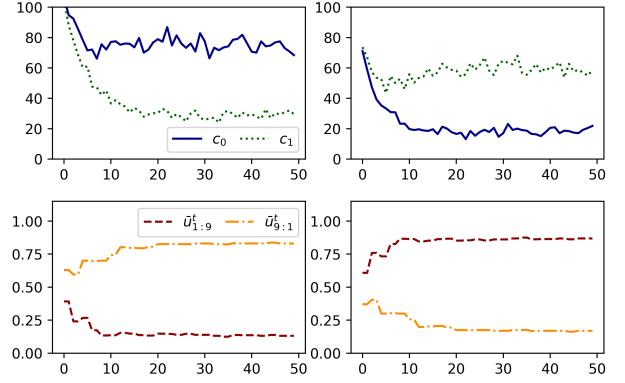


Figure 1: Evolution of the test mean squared error of centers (top) and the importance weight estimation of clients (bottom) over time for the mixture of two synthetic distributions under the 10:90 partition. The left/right columns represent the first/second distributions. Center indices are assigned randomly in the beginning. The importance weight estimations  $\bar{u}_{a:b}^t$  are averaged on clients with the mixture coefficients  $a : b$  (i.e., they have the same local distribution).

We compare **FedSoft** with two baselines: **IFCA** (Ghosh et al. 2020) and **FedEM** (Marfoq et al. 2021). Both baseline algorithms produce one center for each cluster, but they do not explicitly generate local models as in **FedSoft**. Nevertheless, they also estimate the importance weights for each client, we thus use the center corresponding to the largest importance weight as a client's local model. Since we expect cluster models will be deployed to new users, we evaluate their test accuracy/error on holdout datasets sampled from the corresponding cluster distributions. For local models, they are expected to fit the local data of participating clients, we hence evaluate their accuracy/error on local training datasets. Throughout this section, we use  $\bar{c}$  and  $\bar{w}$  to represent the average accuracy/error of the cluster and client models, not the accuracy/error of the averaged models.

We use three datasets to generate the various distributions: Synthetic, EMNIST and CIFAR-10. Due to the space limit, we put the details of experiment parameters and CIFAR-10 results in Appendix C.

- **Synthetic Data.** We generate synthetic datasets according to  $y_i = \langle x_i, \theta_s \rangle + \epsilon_i$  where  $\theta_s \sim \mathcal{N}(0, \sigma_0^2 I_{10})$ ,  $x_i \sim \mathcal{N}(0, I_{10})$ ,  $\epsilon_i \sim \mathcal{N}(0, 1)$  (Ghosh et al. 2020). Unless otherwise noted, we use  $\sigma_0 = 10$ . We use the conventional linear regression model for this dataset.
- **EMNIST Letters.** We use the handwritten images of English letters in the EMNIST dataset to create 2 distributions for the lower and uppercase letters (Guha, Talwalkar, and Smith 2019), each with 26 classes. Then we rotate these images counterclockwise by  $90^\circ$  (Lopez-Paz and Ranzato 2017), resulting in 4 total distributions. In the  $S = 2$  setting we compare the two  $0^\circ$  distributions. A rotation variant CNN model is used for this dataset.

In general, the letter distributions share more similarities with each other, while the synthetic distributions are more divergent, e.g., letters like “O” have very similar upper and

Table 1: MSE or accuracy of cluster models for the mixture of two distributions. Each row represents the distribution of a test dataset. The center with the smallest error or highest accuracy is underlined for each test distribution.

#### Synthetic data: mean squared error

	10:90		30:70		Linear		Random	
	$c_0$	$c_1$	$c_0$	$c_1$	$c_0$	$c_1$	$c_0$	$c_1$
$\theta_0$	68.4	<u>29.5</u>	44.2	49.6	<u>38.2</u>	59.1	<u>42.2</u>	60.6
$\theta_1$	21.8	58.6	41.3	<u>36.3</u>	47.1	<u>27.8</u>	42.7	<u>27.0</u>

#### EMNIST letters: accuracy (%)

	10:90		30:70		Linear		Random	
	$c_0$	$c_1$	$c_0$	$c_1$	$c_0$	$c_1$	$c_0$	$c_1$
Lower	68.9	70.3	65.9	65.8	71.8	71.7	72.0	72.5
Upper	<u>74.6</u>	73.3	70.1	<u>70.4</u>	73.9	<u>74.1</u>	<u>77.7</u>	77.2

Table 2: Comparison between FedSoft and baselines on the letters data.  $c_{lo}^*/c_{up}^*$  represents the accuracy of the center that performs best on the lower/upper distribution, and the number in the parenthesis indicates the index of that center.  $\bar{w}$  is the accuracy of local models averaged over all clients.

	10:90		Linear			$\bar{w}$
	$c_{lo}^*$	$c_{up}^*$	$\bar{w}$	$c_{lo}^*$	$c_{up}^*$	
FedSoft	70.3(1)	74.6(0)	90.9	71.8(0)	74.1(1)	86.5
IFCA	58.5(0)	61.3(1)	65.2	55.4(0)	57.2(0)	62.9
FedEM	67.4(1)	69.8(1)	63.6	65.9(0)	69.0(0)	62.4

lowercase shapes and are invariant to rotations. On the other hand, data generated from  $y = x$  and  $y = -x$  can be easily distinguished. We thus expect the mixture of synthetic data to benefit more from the personalization ability of **FedSoft**.

The typical convergence process of **FedSoft** is shown in Figure 1. In this example of the synthetic data, **FedSoft** is able to automatically distinguish the two cluster distributions. After around 5 global epochs, center 1 starts to exhibit strength on the first cluster distribution, and center 0 concentrates on the other, which implies a correct association between centers and cluster distributions. Similarly, the importance weight estimations  $u_{ks}^t$ , which are initially around 50:50, soon converge to the real mixture ratio 10:90.

Table 1 lists the mean squared error (MSE) or accuracy of the output cluster models. **FedSoft** produces high quality centers under all mixture patterns. In particular, each center exhibits strength on one distribution, which indicates that **FedSoft** builds correct associations for the centers and cluster distributions. The performance gap between two distributions using the same center is larger for the synthetic data. This is because the letter distributions have smaller divergence than the synthetic distributions. Thus, letter models can more easily transfer the knowledge of one distribution to another, and a center focusing on one distribution can perform well on the other. Notably, the 30:70 mixture has the worst performance for both datasets, which is due to the degrading of local solvers when neither distribution dominates. Thus, the local problems under this partition are solved less accurately, resulting in poor local models and a large value

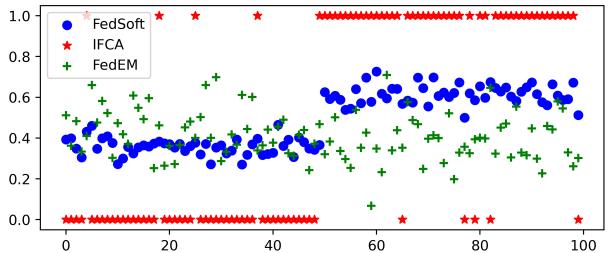


Figure 2: The clients’ estimation of importance weights on the first cluster ( $u_{k0}^T$ ) under the 10:90 partition of the EMNIST dataset. The X axis is the index of clients and each point represents a client.

of  $\gamma$  in Theorem 2, which then produces high training loss on cluster models according to Theorem 4.

Table 2 compares **FedSoft** with the baselines. Not only does **FedSoft** produce more accurate cluster and local models, but it also achieves better balance between the two trained centers. Similarly, Figure 2 shows the importance estimation of clients for the first cluster. **FedSoft** and **IFCA** are able to build the correct association (though the latter is a hard partition), while **FedEM** appears to be biased to the other center by putting less weights (< 0.5) on the first one.

Next, we evaluate the algorithm with the random partition for the mixture of more distributions. Tables 3 and 4 show the MSE or accuracy of cluster models for the mixture of 8 and 4 distributions on synthetic and letters data, where we still observe high-quality outcomes and good association between centers and cluster distributions.

Table 3: Test MSE of centers for the mixture of 8 synthetic distributions with randomly generated weights  $\theta_0 \dots \theta_7$ .

	$c_0$	$c_1$	$c_2$	$c_3$	$c_4$	$c_5$	$c_6$	$c_7$
$\theta_0$	62.2	62.7	64.0	63.2	61.4	<u>57.6</u>	63.7	61.9
$\theta_1$	65.0	67.8	69.4	65.8	64.3	67.1	<u>64.2</u>	64.5
$\theta_2$	60.1	59.9	<u>57.8</u>	58.6	62.6	63.2	60.0	59.9
$\theta_3$	96.8	95.4	96.8	98.8	<u>93.2</u>	93.8	95.3	96.8
$\theta_4$	86.1	89.8	91.8	87.3	85.4	86.6	85.6	<u>84.9</u>
$\theta_5$	161.2	160.3	<u>156.0</u>	160.0	164.7	167.3	162.0	163.6
$\theta_6$	110.2	106.7	<u>104.8</u>	109.0	111.0	107.8	111.5	110.1
$\theta_7$	34.5	<u>33.8</u>	34.8	33.9	34.2	34.1	34.4	35.0

Table 4: Test accuracy (%) of centers for the mixture of 4 distributions with original and 90°-rotated letter images.

	$c_0$		$c_1$		$c_2$		$c_3$	
	$0^\circ$	$90^\circ$	$0^\circ$	$90^\circ$	$0^\circ$	$90^\circ$	$0^\circ$	$90^\circ$
Lower	71.5	<u>67.6</u>	71.3	67.3	71.3	<u>67.6</u>	72.3	67.3
Upper	70.2	71.7	<u>70.8</u>	71.3	70.3	<u>71.9</u>	70.3	71.0

## 6 Conclusion

This paper proposes **FedSoft**, an efficient algorithm generalizing traditional clustered federated learning approaches to allow clients to sample data from a mixture of distributions. By incorporating proximal local updating, **FedSoft** enables

simultaneous training of cluster models for future users, and personalized local models for participating clients, which is achieved without increasing the workload of clients. Theoretical analysis shows the convergence of **FedSoft** for both cluster and client models, and the algorithm exhibits good performance in experiments with various mixture patterns.

## Acknowledgments

This research was partially supported by NSF CNS-1909306 and CNS-2106891.

## References

- Briggs, C.; et al. 2020. Federated learning with hierarchical clustering of local updates to improve training on non-IID data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–9. IEEE.
- Brisimi, T. S.; Chen, R.; Mela, T.; Olshevsky, A.; Paschalidis, I. C.; and Shi, W. 2018. Federated learning of predictive models from federated electronic health records. *International journal of medical informatics*, 112: 59–67.
- Dinh, C. T.; Tran, N. H.; and Nguyen, T. D. 2020. Personalized federated learning with moreau envelopes. *arXiv preprint arXiv:2006.08848*.
- Duan, M.; Liu, D.; Ji, X.; Liu, R.; Liang, L.; Chen, X.; and Tan, Y. 2020. FedGroup: Ternary Cosine Similarity-based Clustered Federated Learning Framework toward High Accuracy in Heterogeneous Data. *arXiv preprint arXiv:2010.06870*.
- Ghosh, A.; Chung, J.; Yin, D.; and Ramchandran, K. 2020. An efficient framework for clustered federated learning. *arXiv preprint arXiv:2006.04088*.
- Glorot, X.; and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 249–256. JMLR Workshop and Conference Proceedings.
- Guha, N.; Talwalkar, A.; and Smith, V. 2019. One-shot federated learning. *arXiv preprint arXiv:1902.11175*.
- Hard, A.; Rao, K.; Mathews, R.; Ramaswamy, S.; Beaufays, F.; Augenstein, S.; Eichner, H.; Kiddon, C.; and Ramage, D. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*.
- Huang, L.; Shea, A. L.; Qian, H.; Masurkar, A.; Deng, H.; and Liu, D. 2019. Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. *Journal of biomedical informatics*, 99: 103291.
- Huang, Y.; Chu, L.; Zhou, Z.; Wang, L.; Liu, J.; Pei, J.; and Zhang, Y. 2021. Personalized cross-silo federated learning on non-iid data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 7865–7873.
- Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2019. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*.
- Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2018. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*.
- Lopez-Paz, D.; and Ranzato, M. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30: 6467–6476.

Luo, Z.-Q.; and Tseng, P. 1992. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1): 7–35.

Mansour, Y.; Mohri, M.; Ro, J.; and Suresh, A. T. 2020. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*.

Marfoq, O.; Neglia, G.; Bellet, A.; Kameni, L.; and Vidal, R. 2021. Federated Multi-Task Learning under a Mixture of Distributions. *International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML 2021 (FL-ICML'21)*.

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

Qayyum, A.; Ahmad, K.; Ahsan, M. A.; Al-Fuqaha, A.; and Qadir, J. 2021. Collaborative federated learning for healthcare: Multi-modal covid-19 diagnosis at the edge. *arXiv preprint arXiv:2101.07511*.

Reynolds, D. A. 2009. Gaussian mixture models. *Encyclopedia of biometrics*, 741: 659–663.

Sattler, F.; Müller, K.-R.; and Samek, W. 2020. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*.

Sim, K. C.; Zadrazil, P.; and Beaufays, F. 2019. An investigation into on-device personalization of end-to-end automatic speech recognition models. *arXiv preprint arXiv:1909.06678*.

Smith, V.; Chiang, C.-K.; Sanjabi, M.; and Talwalkar, A. 2017. Federated multi-task learning. *arXiv preprint arXiv:1705.10467*.

Xie, M.; Long, G.; Shen, T.; Zhou, T.; Wang, X.; Jiang, J.; and Zhang, C. 2021. Multi-center federated learning. *arXiv preprint arXiv:2108.08647*.

## A Proof of Theorems

### Proof of Theorem 1

Let  $G_{ki}, i = 1, 2 \dots$  be some virtual group of client  $k$ 's data points that follow the same distribution. Thus,

*Proof.*

$$\begin{aligned} \mathbb{E}[u_{ks}^t] &\leq \frac{1}{n_k} \left( \sum_{G_{ki} \sim \mathcal{P}_s} \mathbb{P}(\operatorname{argmin}_{s'} F_{s'}(G_{ki}) = s) |G_{ki}| + \sum_{G_{ki} \not\sim \mathcal{P}_s} \mathbb{P}(\operatorname{argmin}_{s'} F_{s'}(G_{ki}) = s) |G_{ki}| \right) + \sigma \\ &\leq \frac{1}{n_k} \left( n_{ks} + \sum_{G_{ki} \not\sim \mathcal{P}_s} p_\epsilon |G_{ki}| \right) = \frac{1}{n_k} (n_{ks} + p_\epsilon (n_k - n_{ks})) + \sigma = (1 - p_\epsilon) u_{ks} + p'_\epsilon \end{aligned} \quad (19)$$

□

### Proof of Theorem 2

*Proof.* For simplification we drop the dependency of  $c_s^t$  in  $h_k$  and  $h_{ks}$ . Take  $w_{ks}^* = \operatorname{argmin}_{w_{ks}} h_{ks}(w_{ks})$ , we have

$$\begin{aligned} \gamma_0^2 \|\nabla F_s(c_s^t)\|^2 &\geq \|\nabla h_k(w_k^t)\|^2 \geq 2\mu_\lambda (h_k(w_k^t) - h_k(w_{ks}^*)) \\ &= 2\mu_\lambda \sum_{s'} u_{ks'} (h_{ks'}(w_k^t) - h_{ks'}(w_{ks}^*)) = 2\mu_\lambda u_{ks} (h_{ks}(w_k^t) - h_{ks}^*) - 2\mu_\lambda \sum_{s' \neq s} u_{ks'} (h_{ks'}(w_{ks}^*) - h_{ks'}(w_k^t)) \\ &\geq \frac{\mu_\lambda}{L_\lambda} u_{ks} \|\nabla h_{ks}(w_k^t)\|^2 - \sum_{s' \neq s} u_{ks'} \|\nabla h_{ks'}(w_{ks}^*)\|^2 \geq \frac{\mu_\lambda}{L_\lambda} u_{ks} \|\nabla h_{ks}(w_k^t)\|^2 - \beta \|\nabla F_s(c_s^t)\|^2 \end{aligned} \quad (20)$$

Thus,

$$\frac{\mu_\lambda}{L_\lambda} u_{ks} \|\nabla h_{ks}(w_k^t)\|^2 \leq (\gamma_0^2 + \beta) \|\nabla F_s(c_s^t)\|^2 \quad (21)$$

Reorganizing, we have

$$\|\nabla h_{ks}(w_k^t)\| \leq \frac{\sqrt{(\gamma_0^2 + \beta)L_\lambda/\mu_\lambda}}{\sqrt{u_{ks}}} \|\nabla F_s(c_s^t)\| = \frac{\gamma}{\sqrt{u_{ks}}} \|\nabla F_s(c_s^t)\| \quad (22)$$

□

### Proof of Theorem 3

*Proof.* Let  $s' = \operatorname{argmax}_s u_{ks}$ , we have  $u_{ks'} \geq \frac{1}{S}$ , thus

$$\begin{aligned} \|w_k^{t+1} - c_s^t\| &= \|w_k^{t+1} - c_{s'}^t + c_{s'}^t - c_s^t\| \\ &\leq \frac{1}{\mu_\lambda} \|\nabla h_{ks'}(w_k^{t+1}; c_{s'}^t) - \nabla h_{ks'}(c_{s'}^t; c_{s'}^t)\| + \|c_{s'}^t - c_s^t\| \\ &\leq \frac{1}{\mu_F} \left( \frac{\gamma}{u_{ks'}} \|\nabla F_{s'}(c_{s'}^t)\| + \|\nabla F_{s'}(c_{s'}^t)\| \right) + \frac{1}{2} \sqrt{\frac{\mu_F}{L_F}} \delta + \Delta \\ &\leq \frac{(\gamma S + 1)L_F}{\mu_F} \|c_{s'}^t - c_{s'}^*\| + \left( \frac{1}{2} \sqrt{\frac{\mu_F}{L_F}} + 1 \right) \Delta \\ &\leq \left( \frac{\gamma S + 1}{4} \sqrt{\frac{L_F}{\mu_F}} + \frac{1}{2} \sqrt{\frac{\mu_F}{L_F}} + 1 \right) \Delta \end{aligned} \quad (23)$$

□

### Proof of Theorem 4

We first introduce the following lemma:

**Lemma 2.**  $\mathbb{E} \left[ \frac{u_{ks}^t n_k}{\sum_{k'} u_{k's}^t n_{k'}} \right] \leq \mathbb{E}[u_{ks}^t n_k] \mathbb{E} \left[ \frac{1}{\sum_{k'} u_{k's}^t n_{k'}} \right]$ , where the expectation is taken over  $\{u_{ks}^t\}$ .

*Proof.* Let  $r^t = \sum_{k' \neq k} u_{k's}^t n_{k'}$ , and note that  $u_{ks}^t \perp r^t$  since each client estimates its  $u_{ks}^t$  independently. Thus,

$$\begin{aligned}
& \mathbb{E} \left[ \frac{u_{ks}^t n_k}{\sum_{k'} u_{k's}^t n_{k'}} \right] = \mathbb{E} \left[ \frac{u_{ks}^t n_k}{u_{ks}^t n_k + r^t} \right] = \mathbb{E} \left[ 1 - \frac{r^t}{u_{ks}^t n_k + r^t} \right] \\
&= \mathbb{E}_{r^t} \left[ \mathbb{E}_{\{u_{ks}^t\}|r^t} \left[ 1 - \frac{r^t}{u_{ks}^t n_k + r^t} \right] \right] \leq \mathbb{E}_{r^t} \left[ 1 - \frac{r^t}{\mathbb{E}[u_{ks}^t n_k|r^t] + r^t} \right] \\
&= \mathbb{E}[u_{ks}^t n_k] \mathbb{E}_{r^t} \left[ \frac{1}{\mathbb{E}[u_{ks}^t n_k] + r^t} \right] \leq \mathbb{E}[u_{ks}^t n_k] \mathbb{E}_{r^t} \left[ \mathbb{E}_{\{u_{ks}^t\}|r^t} \left[ \frac{1}{u_{ks}^t n_k + r^t} \right] \right] \\
&= \mathbb{E}[u_{ks}^t n_k] \mathbb{E} \left[ \frac{1}{\sum_{k'} u_{k's}^t n_{k'}} \right]
\end{aligned} \tag{24}$$

□

We then formally prove Theorem 4:

*Proof.* For  $u_{ks} \neq 0$ , define

$$e_{ks}^{t+1} \triangleq \nabla h_{ks}(w_k^{t+1}; c_s^t) = \nabla F_s(w_k^{t+1}) + \lambda \frac{u_{ks}^t}{u_{ks}} (w_k^{t+1} - c_s^t) \tag{25}$$

using Theorem 2, we have

$$\|e_{ks}^{t+1}\| \leq \frac{\gamma}{\sqrt{u_{ks}}} \|\nabla F_s(c_s^t)\| \leq \frac{\gamma}{u_{ks}} \|\nabla F_s(c_s^t)\| \tag{26}$$

Define

$$\bar{c}_s^{t+1} \triangleq \sum_k v_{sk}^t w_k^{t+1} = \mathbb{E}_{v_{sk}^t} [w_k^{t+1}] \tag{27}$$

thus,

$$\bar{c}_s^{t+1} - c_s^t = -\frac{1}{\lambda} \sum_{k, u_{ks} \neq 0} \left( \frac{u_{ks} n_k}{\sum_{k'} u_{k's}^t n_{k'}} \nabla F_s(w_k^{t+1}) - \frac{u_{ks} n_k}{\sum_{k'} u_{k's}^t n_{k'}} e_{ks}^{t+1} \right) + \sum_{k, u_{ks}=0} v_{sk}^t (w_k^{t+1} - c_s^t) \tag{28}$$

Next we bound  $\|w_k^{t+1} - c_s^t\|$  for  $u_{ks} \neq 0$ ,

$$\begin{aligned}
\|w_k^{t+1} - c_s^t\| &\leq \frac{1}{\mu_\lambda} \|\nabla h_{ks}(w_k^{t+1}; c_s^t) - \nabla h_{ks}(c_s^t; c_s^t)\| \\
&\leq \left( \frac{\gamma}{\mu_\lambda \sqrt{u_{ks}}} + \frac{1}{\mu_\lambda} \right) \|\nabla F_s(c_s^t)\| \leq \frac{\gamma+1}{\mu_\lambda \sqrt{u_{ks}}} \|\nabla F_s(c_s^t)\| \leq \frac{\gamma+1}{\mu_\lambda u_{ks}} \|\nabla F_s(c_s^t)\|
\end{aligned} \tag{29}$$

Therefore,

$$\begin{aligned}
\|\bar{c}_s^{t+1} - c_s^t\|^2 &\leq \mathbb{E}_{v_{sk}^t} [\|w_k^{t+1} - c_s^t\|^2] \\
&\leq \left( \frac{\gamma+1}{\mu_\lambda} \right)^2 \sum_{k, u_{ks} \neq 0} \frac{u_{ks}^t n_k}{u_{ks} \sum_{k'} u_{k's}^t n_{k'}} \|\nabla F_s(c_s^t)\|^2 + \sum_{k, u_{ks}=0} \frac{u_{ks}^t n_k}{\sum_{k'} u_{k's}^t n_{k'}} r^2 \Delta^2
\end{aligned} \tag{30}$$

Define  $M_s^{t+1}$  such that  $\bar{c}_s^{t+1} - c_s^t = -\frac{1}{\lambda} \left( \frac{\sum_k u_{ks} n_k}{\sum_k u_{ks}^t n_k} \nabla F_s(c_s^t) + M_s^{t+1} \right)$

$$\begin{aligned}
M_s^{t+1} &= \sum_{k, u_{ks} \neq 0} \left( \frac{u_{ks} n_k}{\sum_{k'} u_{k's}^t n_{k'}} \nabla F_s(w_k^{t+1}) - \frac{u_{ks} n_k}{\sum_{k'} u_{k's}^t n_{k'}} e_{ks}^{t+1} \right) - \frac{\sum_k u_{ks} n_k}{\sum_k u_{ks}^t n_k} \nabla F_s(c_s^t) - \lambda \sum_{k, u_{ks}=0} v_{sk}^t (w_k^{t+1} - c_s^t) \\
&= \frac{1}{\sum_k u_{ks}^t n_k} \sum_{k, u_{ks} \neq 0} (u_{ks} n_k (\nabla F_s(w_k^{t+1}) - \nabla F_s(c_s^t)) - u_{ks} n_k e_{ks}^{t+1}) - \lambda \sum_{k, u_{ks}=0} v_{sk}^t (w_k^{t+1} - c_s^t)
\end{aligned} \tag{31}$$

thus,

$$\begin{aligned}
\|M_{t+1}\| &\leq \frac{1}{\sum_k u_{ks}^t n_k} \sum_{k, u_{ks} \neq 0} (u_{ks} n_k L_F \|w_k^{t+1} - c_s^t\| + u_{ks} n_k \|e_{ks}^{t+1}\|) + \lambda \sum_{k, u_{ks}=0} v_{sk}^t \|w_k^{t+1} - c_s^t\| \\
&\leq \frac{m_s}{\sum_k u_{ks}^t n_k} \left( \frac{(\gamma+1)L_F}{\mu_\lambda} + \gamma \right) \|\nabla F_s(c_s^t)\| + \lambda \sum_{k, u_{ks}=0} \frac{u_{ks}^t n_k}{\sum_{k'} u_{k's}^t n_{k'}} r \Delta
\end{aligned} \tag{32}$$

Using the smoothness of  $F_s$ , we have

$$\begin{aligned}
F_s(\bar{c}_s^{t+1}) &\leq F_s(c_s^t) + \langle \nabla F_s(c_s^t), \bar{c}_s^{t+1} - c_s^t \rangle + \frac{L_F}{2} \|\bar{c}_s^{t+1} - c_s^t\|^2 \\
&\leq F_s(c_s^t) - \frac{1}{\lambda} \sum_k \frac{u_{ks} n_k}{\sum_k u_{ks}^t n_k} \|\nabla F_s(c_s^t)\|^2 - \frac{1}{\lambda} \langle \nabla F_s(c_s^t), M_s^{t+1} \rangle + \frac{L_F}{2} \|\bar{c}_s^{t+1} - c_s^t\|^2 \\
&\leq F_s(c_s^t) - \frac{1}{\lambda} \frac{n_s}{\sum_k u_{ks}^t n_k} \|\nabla F_s(c_s^t)\|^2 + \frac{m_s}{\lambda \sum_k u_{ks}^t n_k} \left( \frac{(\gamma+1)L_F}{\mu_\lambda} + \gamma \right) \|\nabla F_s(c_s^t)\|^2 \\
&+ \left( \sum_{k, u_{ks}=0} \frac{u_{ks}^t n_k}{\sum_{k'} u_{k's}^t n_{k'}} r \Delta \right) \|\nabla F_s(c_s^t)\| + \frac{L_F}{2} \left( \frac{\gamma+1}{\mu_\lambda} \right)^2 \sum_{k, u_{ks} \neq 0} \frac{u_{ks}^t n_k}{u_{ks} \sum_{k'} u_{k's}^t n_{k'}} \|\nabla F_s(c_s^t)\|^2 \\
&+ \frac{L_F}{2} \sum_{k, u_{ks}=0} \frac{u_{ks}^t n_k}{\sum_{k'} u_{k's}^t n_{k'}} r^2 \Delta^2 \\
&= F_s(c_s^t) - \frac{1}{\sum_k u_{ks}^t n_k} \left( \frac{n_s - \gamma m_s}{\lambda} - \frac{(\gamma+1)L_F m_s}{\mu_\lambda \lambda} \right) \|\nabla F_s(c_s^t)\|^2 + \left( \sum_{k, u_{ks}=0} \frac{u_{ks}^t n_k}{\sum_{k'} u_{k's}^t n_{k'}} r \Delta \right) \|\nabla F_s(c_s^t)\| \\
&+ \frac{L_F}{2} \left( \frac{\gamma+1}{\mu_\lambda} \right)^2 \sum_{k, u_{ks} \neq 0} \frac{u_{ks}^t n_k}{u_{ks} \sum_{k'} u_{k's}^t n_{k'}} \|\nabla F_s(c_s^t)\|^2 + \frac{L_F}{2} \sum_{k, u_{ks}=0} \frac{u_{ks}^t n_k}{\sum_{k'} u_{k's}^t n_{k'}} r^2 \Delta^2
\end{aligned} \tag{33}$$

Taking expectations over  $\{u_{ks}^t\}$ , and applying Lemma 2, we have

$$\begin{aligned}
\mathbb{E}[F_s(\bar{c}_s^{t+1})] &\leq F_s(c_s^t) - \mathbb{E} \left[ \frac{1}{\sum_k u_{ks}^t n_k} \right] \left\{ \left( \frac{n_s - \gamma m_s}{\lambda} - \frac{(\gamma+1)L_F m_s}{\mu_\lambda \lambda} \right) \|\nabla F_s(c_s^t)\|^2 \right. \\
&\quad \left. - (p'_\epsilon \bar{m}_s r \Delta) \|\nabla F_s(c_s^t)\| - \frac{L_F}{2} \left( \frac{\gamma+1}{\mu_\lambda} \right)^2 \hat{m}_s \|\nabla F_s(c_s^t)\|^2 - \frac{L_F}{2} p'_\epsilon \bar{m}_s r^2 \Delta^2 \right\} \\
&\leq F_s(c_s^t) - \mathbb{E} \left[ \frac{1}{\sum_k u_{ks}^t n_k} \right] \left( \frac{n_s - \gamma m_s}{\lambda} - \frac{(\gamma+1)L_F m_s}{\mu_\lambda \lambda} - \frac{p'_\epsilon \bar{m}_s}{2\mu_\lambda} - \frac{L_F(\gamma+1)^2 \hat{m}_s}{2\mu_\lambda^2} \right) \|\nabla F_s(c_s^t)\|^2 \\
&\quad + \mathbb{E} \left[ \frac{1}{\sum_k u_{ks}^t n_k} \right] \frac{p'_\epsilon}{2} (\mu_\lambda + L_F) \bar{m}_s r^2 \Delta^2
\end{aligned} \tag{34}$$

Define

$$\rho_0 \triangleq \frac{n_s - \gamma m_s}{\lambda} - \frac{(\gamma+1)L_F m_s}{\mu_\lambda \lambda} - \frac{p'_\epsilon \bar{m}_s}{2\mu_\lambda} - \frac{L_F(\gamma+1)^2 \hat{m}_s}{2\mu_\lambda^2} \tag{35}$$

$$R_0 \triangleq \frac{1}{2} (\mu_\lambda + L_F) \bar{m}_s r^2 \tag{36}$$

then

$$\mathbb{E}[F_s(\bar{c}_s^{t+1})] \leq F_s(c_s^t) - \mathbb{E} \left[ \frac{1}{\sum_k u_{ks}^t n_k} \right] \rho_0 \|\nabla F_s(c_s^t)\|^2 + \mathbb{E} \left[ \frac{1}{\sum_k u_{ks}^t n_k} \right] p'_\epsilon R_0 \Delta^2 \tag{37}$$

Next we incorporate the client selection.

We can write  $c_s^{t+1} = \frac{1}{K} \sum_{l=1}^K \hat{c}_{s,l}^{t+1}$ , where  $\hat{c}_{s,l}^{t+1} = \sum_{k=1}^N \mathbf{1}(k \in Sel_{s,l}^t) w_k^{t+1}$ , and

$$\mathbb{P}(k \in Sel_{s,l}^t) = \frac{u_{ks}^t n_k}{\sum_{k'} u_{k's}^t n_{k'}} = v_{sk}^t \tag{38}$$

Thus,

$$\mathbb{E}_{Sel_s^t}[\hat{c}_{s,l}^{t+1}] = \mathbb{E}_{v_{sk}^t}[w_k^{t+1}] \tag{39}$$

We then bound the difference between  $F_s(c_s^{t+1})$  and  $F_s(\bar{c}_s^{t+1})$

$$\begin{aligned}
F_s(c_s^{t+1}) &\leq F_s(\bar{c}_s^{t+1}) + \langle \nabla F_s(\bar{c}_s^{t+1}), c_s^{t+1} - \bar{c}_s^{t+1} \rangle + \frac{L_F}{2} \|c_s^{t+1} - \bar{c}_s^{t+1}\|^2 \\
&\leq F_s(\bar{c}_s^{t+1}) + \left( \|\nabla F_s(\bar{c}_s^{t+1})\| + \frac{L_F}{2} \|c_s^{t+1} - \bar{c}_s^{t+1}\| \right) \|c_s^{t+1} - \bar{c}_s^{t+1}\| \\
&\leq F_s(\bar{c}_s^{t+1}) + \left( \|\nabla F_s(\bar{c}_s^{t+1}) - \nabla F_s(c_s^t)\| + \|\nabla F_s(c_s^t)\| + \frac{L_F}{2} (\|c_s^{t+1} - c_s^t\| + \|\bar{c}_s^{t+1} - c_s^t\|) \right) \|c_s^{t+1} - \bar{c}_s^{t+1}\| \\
&\leq F_s(\bar{c}_s^{t+1}) + \left( \|\nabla F_s(c_s^t)\| + L_F \|\bar{c}_s^{t+1} - c_s^t\| + \frac{L_F}{2} (\|c_s^{t+1} - c_s^t\| + \|\bar{c}_s^{t+1} - c_s^t\|) \right) \|c_s^{t+1} - \bar{c}_s^{t+1}\| \\
&= F_s(\bar{c}_s^{t+1}) + \underbrace{\left( \|\nabla F_s(c_s^t)\| + \frac{L_F}{2} \|c_s^{t+1} - c_s^t\| + \frac{3L_F}{2} \|\bar{c}_s^{t+1} - c_s^t\| \right)}_{Q_s^t} \|c_s^{t+1} - \bar{c}_s^{t+1}\|.
\end{aligned} \tag{40}$$

Thus, we only need to bound  $\mathbb{E}[Q_s^t]$

$$\begin{aligned}
\mathbb{E}_{Sel_s^t}[Q_s^t] &= \left( \|\nabla F_s(c_s^t)\| + \frac{3L_F}{2} \|\bar{c}_s^{t+1} - c_s^t\| \right) \mathbb{E}_{Sel_s^t}[\|c_s^{t+1} - \bar{c}_s^{t+1}\|] + \frac{L_F}{2} \mathbb{E}_{Sel_s^t}[\|c_s^{t+1} - c_s^t\| \cdot \|c_s^{t+1} - \bar{c}_s^{t+1}\|] \\
&\leq (\|\nabla F_s(c_s^t)\| + 2L_F \|\bar{c}_s^{t+1} - c_s^t\|) \mathbb{E}_{Sel_s^t}[\|c_s^{t+1} - \bar{c}_s^{t+1}\|] + \frac{L_F}{2} \mathbb{E}_{Sel_s^t}[\|c_s^{t+1} - \bar{c}_s^{t+1}\|^2] \\
&\leq (\|\nabla F_s(c_s^t)\| + 2L_F \|\bar{c}_s^{t+1} - c_s^t\|) \sqrt{\mathbb{E}_{Sel_s^t}[\|c_s^{t+1} - \bar{c}_s^{t+1}\|^2]} + \frac{L_F}{2} \mathbb{E}_{Sel_s^t}[\|c_s^{t+1} - \bar{c}_s^{t+1}\|^2]
\end{aligned} \tag{41}$$

Note that  $\mathbb{E}_{v_{sk}^t}[w_k^{t+1}] = \bar{c}_s^{t+1}$ , we have

$$\begin{aligned}
\mathbb{E}_{Sel_s^t}[\|c_s^{t+1} - \bar{c}_s^{t+1}\|^2] &= \frac{1}{K^2} \mathbb{E}_{Sel_{s,l}^t} \left[ \left\| \sum_{l=1}^K (\hat{c}_{s,l}^{t+1} - \bar{c}_s^{t+1}) \right\|^2 \right] \leq \frac{2}{K^2} \sum_{l=1}^K \mathbb{E}_{Sel_{s,l}^t} \left[ \left\| \hat{c}_{s,l}^{t+1} - \bar{c}_s^{t+1} \right\|^2 \right] \\
&= \frac{2}{K} \mathbb{E}_{v_{sk}^t} [\|w_k^{t+1} - \bar{c}_s^{t+1}\|^2] = \frac{2}{K} \mathbb{E}_{v_{sk}^t} [\|w_k^{t+1} - c_s^t\|^2 - 2\langle w_k^{t+1} - c_s^t, \bar{c}_s^{t+1} - c_s^t \rangle + \|\bar{c}_s^{t+1} - c_s^t\|^2] \\
&= \frac{2}{K} \mathbb{E}_{v_{sk}^t} [\|w_k^{t+1} - c_s^t\|^2 - \|\bar{c}_s^{t+1} - c_s^t\|^2] \leq \frac{2}{K} \mathbb{E}_{v_{sk}^t} [\|w_k^{t+1} - c_s^t\|^2]
\end{aligned} \tag{42}$$

Combining with (30), we can obtain

$$\begin{aligned}
\mathbb{E}_{Sel_s^t}[Q_s^t] &\leq \sqrt{\frac{2}{K}} \sqrt{\mathbb{E}_{v_{sk}^t} [\|w_k^{t+1} - c_s^t\|^2]} \|\nabla F_s(c_s^t)\| + \left( 2L_F \sqrt{\frac{2}{K}} + \frac{L_F}{K} \right) \mathbb{E}_{v_{sk}^t} [\|w_k^{t+1} - c_s^t\|^2] \\
&\leq \frac{1}{2} \sqrt{\frac{2}{K}} \left( \mu_\lambda \mathbb{E}_{v_{sk}^t} [\|w_k^{t+1} - c_s^t\|^2] + \frac{1}{\mu_\lambda} \|\nabla F_s(c_s^t)\|^2 \right) + \left( 2L_F \sqrt{\frac{2}{K}} + \frac{L_F}{K} \right) \mathbb{E}_{v_{sk}^t} [\|w_k^{t+1} - c_s^t\|^2]
\end{aligned} \tag{43}$$

where

$$\mathbb{E} \left[ \mathbb{E}_{v_{sk}^t} [\|w_k^{t+1} - c_s^t\|^2] \right] \leq \mathbb{E} \left[ \frac{1}{\sum_k u_{ks}^t n_k} \right] \left\{ \left( \frac{\gamma+1}{\mu_\lambda} \right)^2 \hat{m}_s \|\nabla F_s(c_s^t)\|^2 + p'_\epsilon \bar{m}_s r^2 \Delta^2 \right\} \tag{44}$$

hence,

$$\begin{aligned}
\mathbb{E}[Q_s^t] &\leq \frac{1}{\mu_\lambda} \sqrt{\frac{1}{2K}} \left( (\gamma+1)^2 \hat{m}_s \mathbb{E} \left[ \frac{1}{\sum_k u_{ks}^t n_k} \right] + 1 \right) \|\nabla F_s(c_s^t)\|^2 \\
&\quad + \underbrace{\left( 2L_F \sqrt{\frac{2}{K}} + \frac{L_F}{K} + \frac{\mu_\lambda}{2} \sqrt{\frac{2}{K}} \right) \mathbb{E} \left[ \frac{1}{\sum_k u_{ks}^t n_k} \right] p'_\epsilon \bar{m}_s r^2 \Delta^2}_{\leq \frac{4L_F + \mu_\lambda}{\sqrt{K}}} \\
&\quad + \underbrace{\left( 2L_F \sqrt{\frac{2}{K}} + \frac{L_F}{K} \right) \mathbb{E} \left[ \frac{1}{\sum_k u_{ks}^t n_k} \right] \left( \frac{\gamma+1}{\mu_\lambda} \right)^2 \hat{m}_s \|\nabla F_s(c_s^t)\|^2}_{\leq \frac{4L_F}{\sqrt{K}}}
\end{aligned} \tag{45}$$

Combining (37), (40), and (45) we have

$$\begin{aligned} \mathbb{E}[F_s(c_s^{t+1})] &\leq \mathbb{E}[F_s(\bar{c}_s^{t+1})] + \mathbb{E}[Q_s^t] \\ &\leq F_s(c_s^t) - \mathbb{E}\left[\frac{1}{\sum_k u_{ks}^t n_k}\right] \left(\rho_0 - \frac{4L_F(\gamma+1)^2 \hat{m}_s}{\mu_\lambda^2 \sqrt{K}} - \frac{(\gamma+1)^2 \hat{m}_s}{\mu_\lambda \sqrt{2K}}\right) \|\nabla F_s(c_s^t)\|^2 + \frac{1}{\mu_\lambda \sqrt{2K}} \|\nabla F_s(c_s^t)\|^2 \\ &\quad + \mathbb{E}\left[\frac{1}{\sum_k (1 - \sum_{s' \neq s} u_{ks}^t) n_k}\right] p'_\epsilon \left(R_0 + \frac{(4L_F + \mu_\lambda) \bar{m}_s r^2}{\sqrt{K}}\right) \Delta^2 \end{aligned} \quad (46)$$

Let  $\lambda$  be chosen such that  $\rho_0 - \frac{4L_F(\gamma+1)^2 \hat{m}_s}{\mu_\lambda^2 \sqrt{K}} - \frac{(\gamma+1)^2 \hat{m}_s}{\mu_\lambda \sqrt{2K}} > 0$ , thus

$$\mathbb{E}[F_s(c_s^{t+1})] \leq F_s(c_s^t) - \frac{\left(\rho_0 - \frac{4L_F(\gamma+1)^2 \hat{m}_s}{\mu_\lambda^2 \sqrt{K}} - \frac{(\gamma+1)^2 \hat{m}_s + (1-p_\epsilon)n_s + p'_\epsilon n}{\mu_\lambda \sqrt{2K}}\right)}{(1-p_\epsilon)n_s + p'_\epsilon n} \|\nabla F_s(c_s^t)\|^2 + \frac{p'_\epsilon \left(R_0 + \frac{(4L_F + \mu_\lambda) \bar{m}_s r^2}{\sqrt{K}}\right) \Delta^2}{(1-p_\epsilon)n_s - p'_\epsilon(S-2)n} \quad (47)$$

□

## Proof of Theorem 5

Note that under Assumption 3, the sum of proximal objectives  $h(w_1 \cdots w_N, c_1 \cdots c_S) = \sum_{k=1}^N h_k(w_k; c, \tilde{u}_k)$  is jointly convex on  $(w_1 \cdots w_N, c_1 \cdots c_S)$  for fixed  $\tilde{u}_k$ , and the training process of **FedSoft** can be regarded as a cyclic block coordinate descent algorithm that sequentially updates  $w_1 \cdots w_N, c_1 \cdots c_S$  while other blocks are fixed. This type of algorithm is known to converge at least linearly to a stationary point (Luo and Tseng 1992).

To see why the averaging of centers correspond to the minimization of them, simply set the gradients to zero

$$\nabla_{c_s} h = \sum_{k=1}^N \tilde{u}_{ks} (c_s - w_k) = 0 \quad (48)$$

This implies the optimal  $c_s^*$  equals

$$c_s^* = \sum_k \frac{\tilde{u}_{ks} w_k}{\sum_k \tilde{u}_{ks}} \quad (49)$$

which is exactly the updating rule of **FedSoft**.

## B The impact of $\tau$ on the convergence

Note that  $\tau$  only affects the accuracy of the importance weight estimations  $u_{ks}^t$ , which determines the estimation error  $p_\epsilon$ .

To incorporate  $\tau$  into the analysis, we first generalize Assumption 4 as follows (Ghosh et al. 2020)

$$\|c_s^t - c_s^*\| \leq (0.5 - \alpha_t) \sqrt{\mu_F/L_F} \delta, \forall s \quad (50)$$

where  $0 < \alpha_t \leq 0.5$  for all  $t$ .

If the algorithm works correctly, the distance between  $c_s^t$  and  $c_s^*$  should decrease over time, thus  $\alpha_t$  will gradually increase from  $\alpha_0$  to 0.5. Then we can change the definition of  $p_\epsilon$  in Lemma 1:

$$\mathbb{P}(\mathcal{E}_t^{j,j'}) \leq p_\epsilon^{t,\tau} \triangleq \frac{c_\epsilon}{\alpha_{\tau \lfloor t/\tau \rfloor}^2 \delta^4} \quad (51)$$

Here we replace  $\alpha_0$  with  $\alpha_{\tau \lfloor t/\tau \rfloor}$ , which takes the same value within each estimation interval. Since we expect  $\alpha_t$  to be increasing, we have  $\alpha_{\tau \lfloor t/\tau \rfloor} \leq \alpha_t$ . Thus, the estimation error  $p_\epsilon^{t,\tau}$  increases when we choose a larger interval  $\tau$ . Plugging this new definition of  $p_\epsilon^{t,\tau}$  to Corollary 1 we have

$$\sum_{t=1}^T \frac{\rho^{t,\tau} \mathbb{E} \|\nabla F_s(c_s^t)\|^2}{(1 - p_\epsilon^{t,\tau}) n_s + (p_\epsilon^{t,\tau})' n} \leq \frac{B_s}{T} + O(p_\epsilon^{0,\tau} \Delta^2) \quad (52)$$

where  $\rho^{t,\tau}$  is defined by replacing all  $p_\epsilon$  with  $p_\epsilon^{t,\tau}$ . This gives us the same asymptotic convergence rate with time as in Corollary 1, except for small differences in constant terms.

## C Experiment Details

### Experiment parameters

- **Synthetic data.** We use a conventional linear regression model without the intercept term. All clients use Adam as the local solver. The number of local epochs equals 10, batch size equals 10, and the initial learning rate equals 5e-3. The same solver is used for both **FedSoft** and the baselines. The regularization weight  $\lambda = 1.0$  for **FedSoft**. The training lasts for 50 global epochs.
- **Letters data.** We use a CNN model comprising two convolutional layers with kernel size equal to 5 and padding equal to 2, each followed by the max-pooling with kernel size equal to 2, then connected to a fully-connected layer with 512 hidden neurons followed by ReLU. All clients use Adam as the local solver, with the number of local epochs equal to 5, batch size equal to 5, and initial learning rate equal to 1e-5. The same solver is used for both **FedSoft** and the baselines. The regularization weight  $\lambda = 0.1$  for **FedSoft**. The training lasts for 200 global epochs.

### Impact of regularization weight $\lambda$

In previous experiments on the letters dataset, we choose  $\lambda = 0.1$ , which is selected through grid search. We show in Table 6 the accuracy of cluster and client models for different choices of  $\lambda$  on the letters dataset, while all other parameters are kept unchanged. As we can see, when  $\lambda = 0$ , no global knowledge is passed to clients, thus the local training is done separately without any cooperation, resulting in poorly trained models. On the other hand, when  $\lambda$  is increased to 1, the local updating is dominated by fitting the local model to the average of global models, and the local knowledge is less emphasized, which also reduces the algorithm performance.

Table 5: Accuracy of cluster and client models for different choices of  $\lambda$  for the linear partition of lower and uppercase letters. All the other parameters are kept unchanged.

	$\lambda = 0$			$\lambda = 0.1$			$\lambda = 1$		
	$c_0$	$c_1$	$\bar{w}$	$c_0$	$c_1$	$\bar{w}$	$c_0$	$c_1$	$\bar{w}$
Lower	48.4	<u>50.1</u>	-	<u>71.8</u>	71.7	-	55.1	<u>55.2</u>	-
Upper	<u>47.7</u>	47.5	-	73.9	<u>74.1</u>	-	<u>58.7</u>	58.6	-
Local	-	-	72.7	-	-	86.5	-	-	63.6

### Impact of divergence among distributions $\Delta$

Finally, we show how the divergence among different distributions  $\Delta$  affects the performance of **FedSoft**. For this experiment we use the synthetic dataset, and we control the divergence by choosing different values of  $\sigma_0$  (i.e.  $\Delta$  increases with  $\sigma_0$ ). As we can see, the MSE significantly increases as the divergence between distributions gets larger, which validates Theorem 4.

Table 6: MSE of cluster and client models for different choices of  $\sigma_0$  for the mixture of two synthetic distributions under the random partition. All the other parameters are kept unchanged.

	$\sigma_0 = 1$			$\sigma_0 = 10$			$\sigma_0 = 50$			$\sigma_0 = 100$		
	$c_0$	$c_1$	$\bar{w}$	$c_0$	$c_1$	$\bar{w}$	$c_0$	$c_1$	$\bar{w}$	$c_0$	$c_1$	$\bar{w}$
$\theta_0$	5.9	<u>4.2</u>	-	<u>42.2</u>	60.6	-	225.3	<u>89.9</u>	-	782.4	<u>454.4</u>	-
$\theta_1$	<u>3.2</u>	4.6	-	42.7	<u>27.0</u>	-	<u>117.6</u>	89.9	-	<u>432.8</u>	812.0	-
Local	-	-	0.6	-	-	4.96	-	-	18.8	-	-	78.3

### CIFAR-10 Results

In this section we evaluate the performance of **FedSoft** for the CIFAR-10 dataset. We consider two data distributions: the original CIFAR-10 images and their rotation by 90° counterclockwise. All images are preprocessed with standard data augmentation tools (Ghosh et al. 2020). We use a CNN model with six convolutional layers, whose channel sizes are sequentially 32, 64, 128, 128, 256, 256. Each convolutional layer follows the ReLU activation and every two convolutional layer follows a max pool layer. The fully connected layer has dimension 1024×512.

20 clients are used for this experiment. We choose client selection size  $K = 15$ , importance estimation interval  $\tau = 2$ , regularization weight  $\lambda = 0.01$ . The local solvers are Adam with initial learning rate equals 5e-4, number of local epochs equals 10, batch size equals 64. The training lasts for 200 global epochs.

Table 7: Accuracy of cluster models for the mixture of two CIFAR-10 distributions. Each row represents the distribution of a test dataset. The center with the highest accuracy is underlined for each test distribution.

	10:90		30:70		Linear		Random	
	$c_0$	$c_1$	$c_0$	$c_1$	$c_0$	$c_1$	$c_0$	$c_1$
$0^\circ$	74.8	<u>77.6</u>	<u>78.9</u>	<u>78.9</u>	77.2	<u>77.4</u>	<u>76.1</u>	76.0
$90^\circ$	<u>77.4</u>	75.3	<u>79.0</u>	78.8	<u>78.5</u>	78.0	75.3	<u>75.8</u>

Table 8: Comparison between FedSoft and baselines on the CIFAR-10 data.  $c_0^*/c_{90}^*$  represents the accuracy of the center that performs best on the  $0^\circ/90^\circ$  distribution, and the number in the parenthesis indicates the index of that center.  $\bar{w}$  is the accuracy of local models averaged over all clients.

	30:70			Linear		
	$c_0^*$	$c_{90}^*$	$\bar{w}$	$c_0^*$	$c_{90}^*$	$\bar{w}$
FedSoft	78.9(1)	79.0(0)	98.8	77.4(1)	78.5(0)	98.6
IFCA	75.6(0)	76.1(0)	99.0	75.1(0)	75.3(0)	98.8
FedEM	76.0(1)	74.6(1)	96.6	76.3(0)	75.8(0)	82.0

Table 7 shows the accuracy of cluster models for all the four data partition patterns presented in Section 5. As we can see, **FedSoft** yields high quality cluster models with a clear separation of different distributions. Table 8 compares **FedSoft** with **IFCA** and **FedEM**. **FedSoft** has the best performance for cluster models, and produces fairly accurate local models.