How Valuable Is Your Data? Optimizing Client Recruitment in Federated Learning

Yichen Ruan, Xiaoxi Zhang, and Carlee Joe-Wong

Abstract—Federated learning allows distributed clients to train a shared machine learning model while preserving user privacy. In this framework, user devices (i.e., clients) perform local iterations of the learning algorithm on their data. These updates are periodically aggregated to form a shared model. Thus, a client represents the bundle of the user data, the device, and the user's willingness to participate: since participating in federated learning requires clients to expend resources and reveal some information about their data, users may require some form of compensation to contribute to the training process. Recruiting more users generally results in higher accuracy, but slower completion time and higher cost. We propose the first work to theoretically analyze the resulting performance tradeoffs in deciding which clients to recruit for the federated learning algorithm. Our framework accounts for both accuracy (training and testing) and efficiency (completion time and cost) metrics. We provide solutions to this NP-Hard optimization problem and verify the value of client recruitment in experiments on synthetic and real-world data. The results of this work can serve as a guideline for the real-world deployment of federated learning and an initial investigation of the client recruitment problem.

Index Terms—Federated Learning, Client Recruitment, Optimization

1 Introduction

Emerging machine learning techniques have achieved great success in creating value from big data. Much of this data is generated by increasingly pervasive mobile devices. These devices, e.g., smart phones, are generally equipped with powerful sensors and considerable storage space, making them great sources of training data for machine learning applications like natural language processing [32] and spam detection [2]. Traditional learning deployments assume all data is stored in a central location (e.g., in a single datacenter) where it can be accessed as needed for training a model. User data is then fully exposed to the operator of the model training. In practice, however, the private nature and the potential volume of user data obstruct its centralized storage and processing, making it hard to make use of these isolated knowledge bases.

The popular federated learning framework allows users to contribute the power of their data to train machine learning models without sharing any raw records. The most common federated learning algorithm is *FedAvg* [20]. It involves one global coordinator, typically a powerful cloud server; and a group of user clients, typically at the network edge. These clients can communicate with the coordinator through the wireless network. During the training, each client in parallel works on a local model defined on its private dataset. These local models are aggregated periodically at the global coordinator, and sent back to synchronize with all clients. This distribution of computation loads makes federated learning a typical example of an edge computing system. The whole training process will take multiple rounds, repeating the local and global steps. Since only the models are shared with the coordinator, users' raw data is kept private.

Under certain conditions, *FedAvg* converges [17]. However, ensuring its good performance in more general scenarios requires overcoming additional challenges that do not present in centralized learning. Specifically, the challenges originate in the skewed distributions of data at different users (statistical challenges), and in the complexity of the edge computing system (system challenges). We will elaborate on these challenges below. Much recent work aims to address them by optimizing the learning algorithm given a set of participating clients (e.g. [29, 30, 35, 38]). However, these works neglect a complementary question: Before running the federated learning algorithm, *how should the operator recruit participating clients so as to optimize the performance of its federated learning algorithm?* In this work, we show that a good client recruitment is essential to overcoming federated learning's statistical and system challenges, complementing algorithmic innovations like carefully selecting or scheduling model updates from a given set of clients.

Client recruitment formalizes the relationship between the two market players in typical **commercial applications** of federated learning: the operators and the users. Operators are typically companies who hope to create or improve their AI products utilizing their users' data. For example, Google has utilized data from Android users to train a query suggestion model for its keyboard application [42]. Recently, the medical industry also proposed to predict hospitalizations for cardiac events [5] with federated learning. Other popular application fields include medicine [5], the Internet-of-Things [18], and vehicular networks [31]. Federated learning operators are responsible for setting up the coordinator and finding the participating clients. However, users are not free labors. Most federated learning algorithms require users to commit to computing local updates, which may consume

- *Y. Ruan and C. Joe-Wong are with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, 15213. E-mails: yichenr@alumni.cmu.edu, cjoewong@andrew.cmu.edu*
- *X. Zhang is with Sun Yat-Sen University, Guangdong, China. Email: zhangxx89@mail.sysu.edu.cn*

Xiaoxi Zhang is the corresponding author. *A preliminary version of this paper appeared in WiOpt 2021 [28].*

limited battery, to the training on demand [29, 30]. These contributions also expose information about private user data, since the local models are sent to the federated learning coordinator [3]. Typically, to ensure convergence, clients are required to stay connected and reserve enough battery to compute updates at all times [20]. Even federated learning algorithms that use only a subset of clients to send updates in each round often assume such clients can be freely selected, i.e., that clients have made a commitment to compute updates on demand [7, 41]. To compensate for these upfront commitments, recruited users may need incentives from the operator to participate in the training, which entails contributing data and computing power and may consume limited battery. In fact, since the local models are exposed to the coordinator through periodic parameter aggregations, federated learning does not entirely preserve user privacy [3]. To compensate for this privacy loss and grant users more control over their data, the operator may pay users for their data, as proposed in [16]. Such compensation, however, introduces a new challenge not commonly considered in federated learning: limiting the recruitment cost.

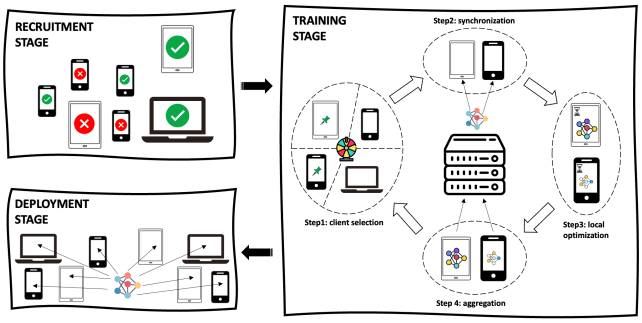


Fig. 1. Typical life cycle of federated learning comprises of client recruitment, model training and model deployment. Client recruitment is the preliminary stage and directly affects the quality of the trained model.

We define **client recruitment** as the preliminary stage of federated learning (Figure 1), in which the coordinator determines the set of candidate clients with which it will train a model. When the recruitment is finalized, we will have determined the quality and quantity of the training data, the number and types of local devices, and the associated cost of compensating users for their upfront commitments. A good client recruitment is thus fundamental to the successful execution of federated learning. Indeed, careful recruitment will reduce the number of clients required to make training commitments (by almost 5x in our experiments), improving federated learning's overall efficiency and impact on user privacy. We emphasize that client *recruitment* differs from the more commonly considered client *selection*, in which a coordinator decides which clients' updates should be included in each iteration of federated learning [23, 41]: client recruitment occurs before the training starts and determines the set of clients eligible for selection in each iteration. E.g., an operator might seek to build a model that uses smartphone weather measurements to predict temperature as a function of location and precipitation, or sensor measurements to predict traffic in a city as a function of location and time. To do so, the operator must recruit clients by respectively deciding from which smartphone users it will request model updates in the future, or which sensors to place where. We elaborate further on client recruitment's potential benefits compared to client selection in Section 2.

The study of the client recruitment problem faces three **challenges**: First, we *lack quantitative metrics for the overall performance of the learning system*. In previous works (e.g., [17, 30]), local datasets are generally taken as given, and "learning performance" is taken to be the training loss. In client recruitment, however, we have the freedom to decide the quality and quantity of the training data, which not only affects the training loss but also controls the model's generalizability to data not in the training set. If clients are not recruited carefully, the resulting training data may yield skewed models, e.g., biased training data is likely a cause of widespread racial bias in facial recognition algorithms [11]. Refining only the learning algorithms or client selection cannot fully address this challenge, e.g., if some data is simply missing from the recruited clients' datasets. Recruiting the right clients, however, must also account for user devices' divergent specifications [12], e.g., differing CPU clock speeds and operating systems. Client recruitment thus also determines the completion time and the operating cost of federated learning systems. We must quantify the dependence of these metrics on the training data, which to the best of our knowledge has not yet been done in the federated setting. Complicating this problem further, *the federated learning operator may not know the data distributions at each client a priori*: knowing such distributions may negate the privacy benefits of federated learning, and client recruitment must take place before training, which may reveal information about client distributions or specifications, begins. Quantifying the relevant metrics, even if possible in theory, is thus difficult in practice.

The second challenge comes from *the complicated tradeoffs between our identified performance metrics in client recruitment*. This challenge originates in the multiple identities of clients. When the operator recruits a client, she recruits both her local dataset and the processing capacity of the associated device to perform updates on the local data. This relationship between the statistical and system characteristics of clients further complicates federated learning's statistical and system challenges. Including the (monetary) cost of compensating clients introduces even more complexity. For example, recruiting more clients allows us to use more training data, but too many participating clients could result in longer completion time and higher operating cost. Similarly, it could be hard to determine if one should recruit a small but statistically valuable dataset (yielding a smaller training loss but potentially poor generalizability), or a big but biased one. This challenge is further complicated by clients' potentially adding noise to their model updates, e.g., to ensure differential privacy guarantees [39].

Given these complex performance tradeoffs, our third challenge is finding *the optimal client recruitment*. Client recruitment is by its nature a combinatorial optimization problem as recruiting each client is a binary choice. Since there may be thousands of candidate clients, naïve brute force is impractical. Generally, the optimal client recruitment problem is NP-Hard, making it difficult to find efficient optimization algorithms.

The **contributions of this paper** are:

- We construct a comprehensive system model to *quantify the performance measures of recruitment in federated learning*, including not only the training loss of the output model, but also the model's generalizability, the reliability and completion time of model training, and the operating expense.
- We formulate an *optimization framework* that captures the complex tradeoffs in client recruitment.
- We exploit the structure of this NP-hard optimization problem to provide a *provably optimal, tractable* solution.
- We introduce *approximation methods for our quality metrics* that can be computed in practice even when clients' data distributions are unknown.
- We demonstrate our work's *practical feasibility* by learning representative models with higher accuracy and fewer recruited clients compared to heuristic methods, on synthetic and real datasets.

Our results can help future researchers quantify the tradeoffs in this problem, and provide guidance for the big data industry on client recruitment in federated learning deployments.

Due to the roles that clients take in federated learning, we will use "users", "clients", "local datasets", and "devices" interchangeably. We review related works in Section 2 and formally introduce the federated learning algorithm in Section 3. In Section 4, we build a comprehensive model to quantify the performance of federated learning systems, and we formulate client recruitment as an optimization problem in Section 5. In Section 6 we provide solutions to this NP-Hard optimization problem, which we evaluate on sample learning problems in Section 7. We conclude in Section 8.

2 Related Works

Federated learning was first introduced with the FedAvg algorithm [20]. Many experiments since then have shown that *FedAvg* can indeed produce accurate machine learning models [15, 42]. Recent papers also demonstrated the power of federated learning for different applications [5, 18, 31]. The convergence of *FedAvg* has been proved for strongly convex and Lipschitz continuous objectives. Specifically, [38] shows that *FedAvg* converges in $O(\frac{1}{T})$ when all clients participate; [17] proves the same convergence rate in general cases, and shows that a decaying learning rate is required for the algorithm to converge to a global optimum. These works also prove that the non-independent-and-identically-distributed (non-IID) local datasets can greatly obstruct convergence, which has been experimentally verified [43].

Some other papers also incorporate system characteristics. E.g., [38] proposes an adaptive control algorithm that jointly considers the accuracy and the cost of training. Another topology-aware framework is proposed in [35], which allows offloading the data between clients. [6] considers federated learning over wireless networks, incorporating wireless communication and energy consumption.

Client selection, which studies the scheduling of client participation in each global round of federated learning, is closely related to client recruitment. E.g., [23] proposes an adaptive selection algorithm to maximize the number of participating clients in each round while subject to resource restrictions. Similar topics are discussed in [41], which assumes clients follow a specific scheduling policy for global aggregations. Other work considers the design of incentive mechanisms to attract federated learning users [13, 24] and the use of differential privacy to enhance the privacy of users' local data [9, 14, 34, 39]. However, generic client selection algorithms cannot guarantee the convergence of federated learning to a globally optimal solution [7]. Client selection, which is typically done at the start of each training iteration, also requires all clients to stay active and ready to be summoned anytime, even if they are not always selected, which is unrealistic in practice due to the system challenges to be discussed in Section 3.2.

Client recruitment supplements the existing works on client selection and can be understood as a pre-processing procedure. Compared to client selection, which cherry picks from a small set of clients, client recruitment is expected to deal with a much larger candidate pool. In practical settings, client recruitment is more useful since it pre-excludes disqualified clients before any training steps are taken or incentives offered, while client selection cannot remove the coordinator's business obligation to the recruited clients (e.g., monetary incentive payments), who have committed to being available for training even if they are never ultimately selected. The client recruitment method discussed in this paper is independent of the remaining training details, and can thus be coupled with any federated learning algorithm and client selection strategies.

3 Federated Learning Background

This section formally introduces federated learning based on the *FedAvg* algorithm [20] and discusses the statistical and system challenges of deploying it in practice.

3.1 Federated Optimization

Federated learning trains a single model by attempting to minimize the model's empirical risk, i.e., the training loss, over data from multiple clients. Suppose there are K clients, each of whom owns a dataset $\mathcal{D}_k = \{(u_i, v_i)\}_i$, where u_i, v_i are respectively a feature vector and the corresponding label. We index the data samples by i. Let $l(w; u, v)$ be a loss function with weight vector w and a data record (u, v), i.e., a measure of how accurately a model with weight w predicts the label v for input u. Typical loss functions are cross entropy for classification and L_2 distance for regression. The local empirical risk of client k for a given weight w is defined as the sample mean of the loss over its data $\mathcal{D}_k$:

$$\tilde{R}_k(w; \mathcal{D}_k) = \frac{1}{|\mathcal{D}_k|} \sum\nolimits_i l(w; u_i, v_i) \tag{1}$$

The ultimate goal of training is thus to find w that minimizes the empirical risk over the global dataset $\mathcal{D}_x = \cup_k \mathcal{D}_k$ [1]:

$$\min_w \tilde{R}_x(w; \mathcal{D}_x) = \sum\nolimits_{k=1}^{K} \frac{n_k}{n_x} \tilde{R}_k(w; \mathcal{D}_k) \tag{2}$$

Here, we define $n_k \triangleq |\mathcal{D}_k|$, which is the number of data samples of client k, and $n_x \triangleq \sum_k n_k$, which represents the

1. For ease of notation, we use $\cup_k$ (or $\sum_k$) to denote the union (or sum) taken over all the recuited clients, in contrast to $\sum_k^K$.

total number of samples in the global datasets over all the recruited users. The subscript x indicates that this global dataset is determined by the recruitment decision $\mathbf{x}$, which we will formally define in Section 4. To minimize $\tilde{R}_x$, the distributed stochastic gradient descent (SGD) paradigm is utilized. A central coordinator maintains a global weight vector w, and each client k maintains local weights w_k. The training repeats the following four steps for $t = 1, ..., T$.

- *Client Selection*: The coordinator selects a random subset of clients to participate in the training for this round.
- *Synchronization*: The coordinator broadcasts the global weight vector $w^{\tau t}$ to the clients through the network. Clients then update their local weights $w_k^{\tau t}$ with $w^{\tau t}$.
- *Local Optimization*: Each client k runs SGD independently and in parallel for τ steps to minimize its local risk $\tilde{R}_k$, getting local weights $w_k^{(t+1)\tau}$.
- *Aggregation*: The coordinator aggregates $\{w_k^{(t+1)\tau}\}_k$ from clients by setting the next global weight vector as their weighted average: $w^{(t+1)\tau} = \sum_k \frac{n_k}{n_x} w_k^{(t+1)\tau}$.

Since only the weight vector w is exchanged with the central coordinator, it is guaranteed that each local dataset will only be accessible by the client itself; thus no raw data records will be disclosed.

In the following sections we will assume all the algorithm parameters are given, which include the number of local/global epochs τ/T and so on. This simplification allows our results to be robust to different algorithm settings and complement optimizations of these parameters. In fact, these algorithm parameters often cannot be optimized precisely, especially as some client characteristics may not be known to the operator. Co-optimizing the parameters with client recruitment is an interesting area of future work.

3.2 Learning Challenges

The optimization paradigm introduced above resembles the distributed machine learning framework traditionally used in datacenters. However, as mentioned earlier, federated learning faces both statistical and system challenges that do not appear in the data center environment. Intuitively, this is because, first, federated learning performs multiple local iterations between aggregations; and second, mobile devices are more prone to failures than powerful servers. We will incorporate these challenges into the analysis of the client recruitment problem, and show in simulations that client recruitment allows us to overcome these challenges.

Statistical Challenge: Non-IID datasets. Unlike in the data center where data is assumed to be identically and independently (e.g., [40]) distributed among workers (e.g., by randomly shuffling data between workers), clients' data distributions in federated learning may well be non-IID. Specifically, we suppose that samples in every local dataset $\mathcal{D}_k$ are independently drawn from a distinct distribution $\mathcal{P}_k$. Depending on the application, the distributions $\{\mathcal{P}_k\}_k$ may differ significantly with each other. Both experiments and theoretical results have shown that non-IID data can greatly decelerate the training [20]. Intuitively, since clients train on their local datasets, their local models are skewed towards their local distributions. The aggregation step will help alleviate this bias but cannot completely remove it.

TABLE 1
Notations

x	Recruitment decisions, $x \in \{0,1\}^N$
K	Number of candidate clients
$\mathcal{D}_k, \tilde{\mathcal{P}}_k, \mathcal{P}_k$	Local dataset and its empirical, real dist.
$\mathcal{D}_x, \tilde{\mathcal{P}}_x$	Global dataset and its empirical distributions
$\tilde{\mathcal{P}}, \mathcal{P}$	Estimated and real population distributions
$\tilde{R}_k, \tilde{R}_x$	Local and global empirical risks
n_k, n_x	Number of data points in local/global datasets
N	Number of client groups
m_z	Number of clients in group $z, z = 1, ..., N$
λ^z, q_f^z, q_r^z	Processing, failure, recovery rate of group z
c_k	Cost to recruit client k

Recently, researchers [17, 38] have bounded the training loss with a monotonically increasing function of the average difference in loss functions evaluated with local and global optima w_k^* and w_x^*. A similar bound uses the divergence in local and global empirical distributions instead:

$$\text{training loss} \propto \sum\nolimits_k \frac{n_k}{n_x} |\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}_x| \tag{3}$$

Here $\tilde{P}_k$ and $\tilde{P}_x$ respectively denote the local and global empirical distributions. In other words, the training loss, i.e., the global empirical risk, increases for more dissimilar local and global data. For example, [43] shows that the training loss increases monotonically with the earth mover's distance between local and global distributions.

System Challenge: Stragglers and failures. User devices such as smart phones and tablets have relatively constrained computation resources (e.g. CPU, memory, storage), which furthermore must be shared among many applications. These weaknesses make stragglers (devices that take a long time to run local iterations) more likely to appear in federated learning. Furthermore, edge devices generally lack data centers' extensive backup resources [1]. Machine failures are so rare as to be generally ignored in cloud-based learning algorithms [40], but the risk of client failure in edge computing settings is considerably higher [22]. The appearance of stragglers and failures will cause the remaining devices to wait idly in the aggregation step, slowing down the training speed.

A final challenge often not considered in distributed learning is that of *user compensation*, for both computing updates and giving up potentially private information about their data through sending these updates to a central server. We discuss the resulting monetary cost in Section 4.2.

4 System Modeling

Formally speaking, given a list of candidate clients $\mathcal{U} = \{u_k\}_k$, the goal of client recruitment is to recruit the optimal subset of clients $\mathcal{U}_x \subseteq \mathcal{U}$ that will run the learning algorithm to optimize the overall performance of federated learning. In this section, we first model the relevant data distributions and then propose formal performance metrics for both the accuracy and the efficiency of the learning algorithm.

4.1 Data Distributions

We assume all data points are generated in an IID manner from a *population distribution* $\mathcal{P}$, no matter the client to which

they belong. On the other hand, each client's data individually forms a *local distribution* $\mathcal{P}_k$, which generally differs from other local distributions and from $\mathcal{P}$, also known as non-IID data distributions across clients. Figure 2 depicts an example relationship between the local and population distributions. Clients collect data only within their own domains, which are subsets of the overall data distribution, resulting in distinct perceptions of the data.

The existence of an universal population distribution $\mathcal{P}$ is the premise of federated learning. Indeed, it does not make sense to train a single model over separate data clusters if their data does not present some correlation or similarity.[2] The objective of federated learning can thus be understood as to build an universally applicable model over $\mathcal{P}$ with limited knowledge of and access to samples from the local distributions $\mathcal{P}_k$'s. Local distributions can be different or even biased from the population for various reasons, such as clients only collecting data in certain physical locations, while the population covers all such locations.

When we compare two local datasets, we assume the data is independent but not identically distributed between them. In contrast, when we discuss the union of all local datasets, we treat each sample as IID distributed in $\mathcal{P}$. For example, suppose we are training a model to predict temperature from features such as the amount of sunlight, rainfall, and humidity. In this case, $\mathcal{P}$ represents the joint distribution of world-wide temperature with these features. Since a client can only collect temperature data in a small region, its local distribution $\mathcal{P}_k$ only reflects the regional climate characteristic (e.g., high humidity may correlate with high temperatures in New York but not California). As a result, clients in different regions possess divergent local distributions, and most local datasets cannot cover the whole temperature range of $\mathcal{P}$. On the other hand, all these data points are essentially generated within the same Earth climate system. Thus when forged together, they do follow the world-wide distribution $\mathcal{P}$ in an IID manner. We can therefore use federated learning to train an universal climate prediction model based on these datasets.

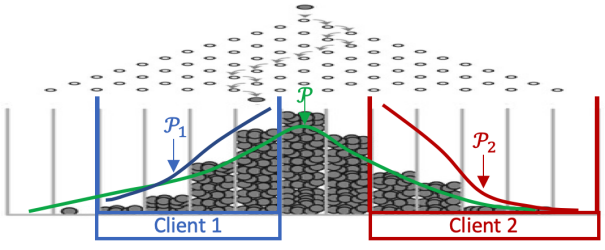


Fig. 2. The relationship between the population distribution $\mathcal{P}$ and local distributions $\mathcal{P}_k$. The numbers of balls passing through the Galton board in different slots follow the binomial distribution $\mathcal{P}$. However, two clients both only have limited views of the population. Their local datasets thus follow two distinct local distributions $\mathcal{P}_1 \neq \mathcal{P}_2 \neq \mathcal{P}$.

In the meanwhile, a local dataset $\mathcal{D}_k$ may not describe its local distribution $\mathcal{P}_k$ well if insufficient data points were collected. In fact, only in rare cases will $\mathcal{P}_k$ be known by the client, let alone by the federated learning operator. In practice, the operator estimates the divergence between $\mathcal{P}_k$ and $\mathcal{P}$ through client k's *empirical distribution* $\tilde{\mathcal{P}}_k$, which converges to the real $\mathcal{P}_k$ when $\mathcal{D}_k$ grows larger.

Similarly, we define the global dataset $\mathcal{D}_x = \cup_k \mathcal{D}_k$ as the union of all local datasets over recruited clients. Data in $\mathcal{D}_x$ forms the *global empirical distribution* $\tilde{\mathcal{P}}_x$. It is easy to check that the empirical risk of $\mathcal{D}_x$ is exactly the $\tilde{R}_x$ defined in (2) as a weighted average of $\tilde{R}_k$'s. Likewise, $\tilde{\mathcal{P}}_x$ is a weighted average of local empirical distributions: $\tilde{\mathcal{P}}_x = \sum_k \frac{n_k}{n_x} \tilde{\mathcal{P}}_k$.

Since all data in $\mathcal{D}_x$ is independently drawn from the population distribution $\mathcal{P}$, $\tilde{\mathcal{P}}_x$ can be regarded as an empirical estimation of $\mathcal{P}$ when a reasonably large number of clients are recruited. In this paper, we assume that $\tilde{\mathcal{P}}_x$ is an empirical distribution of $\mathcal{P}$ that converges to $\mathcal{P}$ as more clients (thus more datasets) get recruited. In the climate data example, if we have recruited clients from all climatic zones in the world, the union of this data $\tilde{\mathcal{P}}_x$ becomes a good representative of the whole Earth climate system $\mathcal{P}$.

To facilitate client recruitment, we assume the federated learning operator can estimate $\mathcal{P}$, e.g., with a small set of IID data points $\tilde{D}$ sampled from $\mathcal{P}$ based on the operator's prior knowledge on the population of interest (e.g., as in [36]), and we denote its empirical distribution by $\tilde{\mathcal{P}}$. Since $\tilde{D}$ is small in size, it cannot be directly used for training. Such a sample dataset is likely needed to determine whether learning a federated model on these features makes sense in the first place. It can also be considered as a basis to generate the test dataset. For example, before using climate data in user devices for model training, we can estimate the population distribution by investigating the public historic data. In principle, knowledge of $\tilde{\mathcal{P}}$ is sufficient to learn a model for our data. However, if $\tilde{D}$ is merely a basis to generate the test dataset, this knowledge may not be sufficient to learn a model. For example, in Section 5.1, we show that it is sufficient to estimate $|\tilde{P} - \tilde{P}_k|$ by knowing the distribution of data labels; such distribution is clearly insufficient to learn the entire model.

Alternatively, the operator may purchase samples directly from users who are willing to sell their data with a relatively high price. The operator can choose from which clients to purchase data by leveraging prior knowledge of context features, e.g., ensuring the collected samples cover all of the basic climate zones if the goal is to learn a climate model. Since the cost is expensive, the resulting $\tilde{D}$ is small, but can still reflect the main characteristics of the population. This educated guess of the population allows the operator to gauge the clients' quality and representativeness. After all, without a basic understanding of the target data population, it is impossible to decide what data is needed for the training.

4.2 Performance Metrics

We will consider two categories of performance measures. The first category is the accuracy of the output model, which includes not only the training loss (i.e. global empirical risk), but also the model's generalizability (i.e., the expected accuracy when applied to fresh data samples from recruited clients) and its representativenss (i.e., the expected accuracy when applied to data from the rest of the population that

2. Federated learning variants with personalized local models [33] may not require an universal population distribution, but the local models must still share some similarities. We focus on training a single global model in this work.

is not recruited). The second category measures the training efficiency, which includes the time to complete the training, and the cost incurred. Putting them together, below we list five metrics that we will consider in this paper. All of them are critical to the commercial success of federated learning, ensuring that we can train an accurate model with reasonable time and cost. We discuss how to compute these metrics in practice in Section 5.

Reduce training loss with high-quality data: A dataset $\mathcal{D}_k$ is considered of good quality if its $\tilde{\mathcal{P}}_k$ resembles the population $\mathcal{P}$. From (3), if $\tilde{\mathcal{P}}_x$ resembles $\mathcal{P}$ (see "representativeness" below), the quality of the local datasets directly determines the training loss. A dataset $\mathcal{D}_k$ with a small distribution divergence $|\tilde{\mathcal{P}}_k - \mathcal{P}|$ yields small training loss.

Reduce generalization error with more data: Given a loss function l and dataset $\mathcal{D}$, the generalization error $|\tilde{R} - R|$ is the divergence between the empirical risk $\tilde{R}(w;\mathcal{D}) = (\sum l(w;u,v))/|\mathcal{D}|$ and the real risk $R(w) = \mathbb{E}_{\mathcal{P}}[l] = \int l(w;u,v)d\mathcal{P}$. While the training loss gauges the model's performance on the training data, the generalization error reflects its accuracy when applied to *new samples drawn from the recruited distributions*. If a client has insufficient data, its local empirical distribution $\tilde{\mathcal{P}}_k$ may poorly approximate $\mathcal{P}_k$, which implies a large generalization error. Existing works generally omit the generalization error as they take the training data as given. For us, however, client recruitment determines the size of the training dataset, affecting the generalization error.

Choose for population representativeness: In order for the trained model to be applicable to *unrecruited datasets*, the recruited clients, when forged together, must be representative of the population $\mathcal{P}$. Indeed, if the clients do not cover portions of the population space, we will perform poorly in those areas. For example, including polar region data complicates the training of models that predict worldwide temperatures, but failing to do so can degrade the model's performance in this region.

Control the completion time: Federated learning is useless if the training process does not complete in reasonable time. We define the completion time as the expected time for the coordinator to finish all T rounds of aggregations, which is related to the per round runtime of the clients. Clients that come with more powerful devices allow the training to complete faster. However, recruiting too many clients increases the probability of client failures, which results in additional runtime from waiting for these failed clients.

Control the cost: Since the size of an individual local dataset is usually small, a typical execution of federated learning may need thousands of recruited clients [4]. The operator should thus make sure the resulting recruitment expense is affordable. Clients may also add noise to their contributed model updates, e.g., to satisfy differential privacy guarantees [39]. This addition may further increase the training loss [14, 39] but may lower the cost, as the contribution of updates yields less privacy leakage.[3]

3. Adding noise to the model updates may decrease the generalization error and representativeness associated with the recruited clients in practice, as the model no longer directly minimizes the training loss. For simplicity, we do not include these effects, which are hard to predict a priori, in our model.

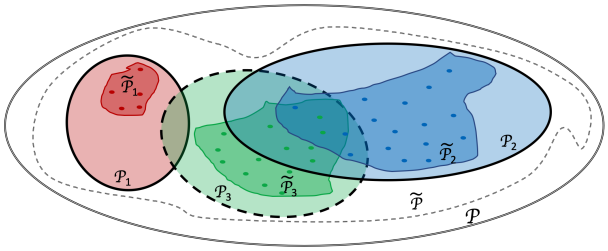


Fig. 3. The choices of three datasets. $\mathcal{D}_1, \mathcal{D}_2$ are chosen, while $\mathcal{D}_3$ is not. Each dataset $\mathcal{D}_i$ follows distribution $\mathcal{P}_i$. $\tilde{\mathcal{P}}$ is the prior knowledge on the population distribution $\mathcal{P}$. The global empirical distribution $\tilde{\mathcal{P}}_x = \tilde{\mathcal{P}}_1 \cup \tilde{\mathcal{P}}_2$ converges to the population $\mathcal{P}$ when more additional datasets get chosen.

While all these metrics are critical in practice, client recruitment is unlikely to attain their optimal points simultaneously. Generally, improving the accuracy metrics entails recruiting more clients, which increases the completion time and cost. Likewise, tradeoffs exist among accuracy metrics. Figure 3 qualitatively illustrates them with an example: First, compared to client 1's data distribution $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_3$ have better qualities because their distributions, $\mathcal{P}_2$ and $\mathcal{P}_3$, are closer to the population distribution $\mathcal{P}$. Therefore, $\tilde{\mathcal{P}}_2$ and $\tilde{\mathcal{P}}_3$ are more similar to each other. Choosing them would result in a smaller risk divergence as defined in (3), reducing the training loss. Second, the empirical distribution could be biased from its real counterpart due to insufficient samples. For example, the empirical distribution $\tilde{\mathcal{P}}_1$ is highly biased from $\mathcal{P}_1$. A federated learning model trained with only $\mathcal{D}_1$ may thus not perform well when tested on new data points sampled from $\mathcal{P}_1$. Third, the global $\mathcal{P}_x$ could be skewed away from the real population distribution $\mathcal{P}$ if we recruited only one client instead of two. Intuitively, any single local distribution fails to represent almost half of the population sample space, which may also lead the trained model to generalize badly when applied to fresh samples in the population. Thus, we may always want to recruit more clients to increase the representativeness. A tradeoff thus appears when deciding which distribution to choose. For example, we will obtain better representativeness but higher generalizability error by recruiting client 1.

5 Problem Formulation

To jointly consider the metrics in Section 4.2, we formulate client recruitment as an optimization problem: Given a set of candidate clients $\mathcal{U} = \{U_k\}_{k=1}^K$, let $x \in \{0,1\}^K$ be a binary vector denoting the recruitment decision for each client. The operator picks an optimal subset $\mathcal{U}_x = \{U_j | x_j = 1\}$ to minimize an objective function f, subject to a given maximum completion time I_t and cost I_c.

Problem 1. *Client Recruitment*

$$\min_{x\in\{0,1\}^K} \quad f(x) = \gamma_{tl} f_{tl}(x) + \gamma_{ge} f_{ge}(x) + \gamma_{rp} f_{rp}(x)$$
$$s.t. \quad g_t(x) \le I_t, g_c(x) \le I_c$$

We derive expressions for the completion time $g_t(x)$ and cost $g_c(x)$ in Section 5.2. Here $f(x)$ consists of three terms that determine the accuracy of the trained model: f_{tl}, f_{ge}, f_{rp}, which respectively upper bound the training

loss, the generalizability, and the representativeness. The objective f determines the goodness of the trained model when applied to existing or future data points generated by both recruited and unrecruited clients. The coefficients $\gamma_{tl}, \gamma_{ge}, \gamma_{rp}$ determine the relative importance of these terms. For example, if the operator runs thousands of iterations (i.e., T is large), the client recruitment may not significantly affect the training loss. The operator may instead emphasize the generalization error and the representativeness, which cannot be compensated for with more iterations.

5.1 Quantifying Accuracy Metrics

We first consider the training loss. According to (3), the training loss is determined by the divergence between local and global empirical distributions $\sum_k \frac{n_k}{n}|\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}_x|$. However, since the global distribution can only be determined after the recruitment process, it is difficult to optimize the divergence directly. Thus, we utilize the fact that $\tilde{\mathcal{P}}_x$ resembles the population $\mathcal{P}$ (Lemma 2) to define:

$$f_{tl}(x) = \sum\nolimits_k \frac{x_k n_k}{n_x}|\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}| \tag{4}$$

Sharing this metric preserves user privacy since it does not actually require the individual empirical distributions $\tilde{\mathcal{P}}_k$'s. Instead, $\tilde{\mathcal{P}}$ is broadcast to clients by the coordinator, and the returned information from the clients $|\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}|$ only encodes the distance of local distributions to the population. The model parameter transfer in federated learning reveals similar information [3]. If the user adds noise to its model updates so as to guarantee differential privacy, then the training loss will increase due to the additional noise [14]. However, as this added noise is independent across clients, we can still approximate the training loss as a sum of terms proportional to $|\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}|$, where we can absorb extra constants into the constant $\frac{n_k}{n_x}$.

Depending on the data and model being used, the divergence can in practice be calculated in many different ways, e.g., [37] empirically measures the quality using a predefined test dataset. Below we provide tractable methods and formula to approximate the training loss f_{tl} for generic machine learning tasks from limited client information.

- *Counting classes:* Consider a classification problem with L classes, and suppose the local empirical distribution $\tilde{\mathcal{P}}_k$ and empirical population distribution $\tilde{\mathcal{P}}$ have densities $\tilde{p}_k$ and $\tilde{p}$. We can then write the divergence as $|\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}| = \int |\tilde{p}_k(u,v) - \tilde{p}(u,v)| du dv = \sum_{i\in[L]} \int |\tilde{p}_k(u|v)\tilde{p}_k(v=i) - \tilde{p}(u|v)\tilde{p}(v=i)| du$. We assume $\tilde{p}_k(u|v) = \tilde{p}(u|v)$, i.e., local features have the same distribution as the population given the label v. Thus, $\int |\tilde{p}_k - \tilde{p}| \propto \sum_{i\in[L]} |\tilde{p}_k(v=i) - \tilde{p}(v=i)|$, where the population distribution $\tilde{p}$ is known a priori. If a client k has j labels, $\tilde{p}_k(v=i) = 1/j$ if label i is included in $\mathcal{D}_k$, or $\tilde{p}_k(v=i) = 0$ otherwise. Thus, the whole term $\sum |\tilde{p}_k(v=i) - \tilde{p}(v=i)|$ can be easily computed by simply counting the number of labels each client sees, and estimating $\tilde{P}$ from $\tilde{D}$ entails the same simple counting process.
- *Gaussian graphical model approximation:* For general supervised learning with continuous labels, we can formulate the features and the label as a Gaussian graphic model. A local empirical distribution is then fully specified by the mean and covariance $(\tilde{\mu}_k, \tilde{\Sigma}_k)$. The quality measure then becomes the divergence between two Gaussian distributions, which can be quantified by the Kullback–Leibler divergence: $|\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}| \propto D_{KL}\left(\mathcal{N}(\tilde{\mu}_k, \tilde{\Sigma}_k), \mathcal{N}(\tilde{\mu}, \tilde{\Sigma})\right)$. Estimating $\tilde{P}$ from $\tilde{D}$ entails computing $\tilde{\mu}, \tilde{\Sigma}$ and inferring the graph connectivity (e.g. from the covariance).

We will demonstrate the effectiveness of these two approximation methods in Section 7. Not only does our optimal recruitment strategy obtain the best performance, but we also find that recruiting more high-quality clients outperforms the learning outcome when recruiting more large-quantity clients, which shows that our quality estimations are sufficiently accurate to improve the learning process.

Next, we model the average generalizability. Since the training objective of federated learning $\tilde{R}_x$ is an average of local empirical risks as in (2), we similarly quantify the average generalization error of local datasets by $\sum_k \frac{x_k n_k}{n_x}|\tilde{R}_k - R_k|$. To formulate it, we rely on Lemma 1 as follows:

Lemma 1. *There exists a class of convex learning problems (e.g. linear regression), for which we can obtain the following generalization error bound for all clients k:*

$$|\tilde{R}_k - R_k| = O\left(n_k^{-0.5}\right) \tag{5}$$

For example, [26] proves this bound for the linear regression model. A tighter convergence bound is also possible using more sophisticated statistical tools such as Hoeffding's inequality. For simplification and to accommodate non-convex models that may have looser risk generalization bounds, this paper assumes a relatively common asymptotic bound [10, 26], but our analysis can be easily extended to using any functions of n_k as the generalization error bound as long as it can be easily computed without using private information. We thus define,

$$f_{ge}(x) = \sum\nolimits_k \frac{x_k n_k}{n_x} n_k^{-0.5} \tag{6}$$

We then model the representativeness. To make sure the chosen distributions can represent basic characteristics of the population distribution, we seek to minimize the divergence between $\tilde{\mathcal{P}}_x$ and $\mathcal{P}$. As is discussed in Section 4.1 where we assume $\tilde{\mathcal{P}}_x$ is an empirical distribution of $\mathcal{P}$, and using the central limit theorem, we have the following uniform bound:

Lemma 2. *$\tilde{\mathcal{P}}_x - \mathcal{P}$ converges in distribution to the Gaussian distribution with 0 mean at the rate of $O(n_x^{-0.5})$.*

Therefore, statistically, when n_x grows larger, $\tilde{\mathcal{P}}_x$ becomes a more accurate representative of $\mathcal{P}$. We thus define:

$$f_{rp}(x) = n_x^{-0.5} \tag{7}$$

Though we use the standard convergence rate of the central limit theorem in Lemma 2 and (7), our analysis and algorithm can still work for general $O(n_x^{-\beta})$ with $0 < \beta < 1$.[4]

4. In practice, the constant associated with this term may be large, e.g., if the data is high-dimensional. However, we can still use this expression as a guide to recruit more representative clients that benefit the overall learning process, even if the actual representativeness error is far from zero.

Combining (4), (6), and (7), and reorganizing, the objective f can be expressed as in (8). Since the associated importance coefficient γ_{rp} in (8) is a positive constant independent of x, we normalize it to 1. The s_k value here is a weighted sum of client k's quality representation $|\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}|$ and quantity representation $n_k^{-0.5}$. It thus enables us to quantify the quality-quantity tradeoff when choosing user datasets.

$$\begin{aligned} f(x) &= \gamma_{rp}\left(\frac{\sum_k x_k n_k s_k}{n_x} + n_x^{-0.5}\right) \\ s_k &= \frac{\gamma_{tl}}{\gamma_{rp}}|\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}| + \frac{\gamma_{ge}}{\gamma_{rp}} n_k^{-0.5} \end{aligned} \tag{8}$$

The required local information from clients in (8) includes only n_k and $|\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}|$. The former is trivial to compute, as it only requires the number of datapoints, and the latter can be estimated using the approximation methods discussed above. As discussed above (4), while $|\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}|$ reveals some information about client k's data, it is unlikely that this single scalar would significantly compromise data privacy, as it only encodes the distance from the local data distribution to that of the population.

5.2 Quantifying System Metrics

Now we analyze the completion time constraint. The completion time depends on the specific aggregation scheme used in federated learning. When the coordinator constructs the next global weight vector, it has to wait for the clients to upload their local weights. Some of the weights, however, may not be delivered due to straggling and failure. One possible solution is to only wait for the first few clients, and discard the rest of them. This is usually referred to as K-synchrononous SGD [8]. Unfortunately, in the federated setting, this scheme may cause the coordinator to wait indefinitely if several failures occur simultaneously. We will consider a more resilient aggregation scheme: For each round of the federated optimization iteration, the coordinator will wait up to a predetermined duration E_0. The global weight will then be calculated based on the weights received before the deadline. Thus, the runtime for one iteration equals the runtime of the slowest client if all clients complete by E_0, or E_0 if a client fails in this iteration.

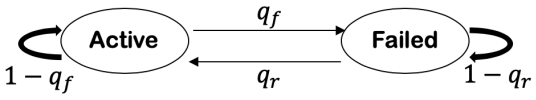


Fig. 4. A two state Markov chain representing the state transition of clients. An active client fails with probability q_f. A failed client recovers with probability q_r.

We model the client failure as a Markov chain, and we calculate the recovery and failure rates based on the mean times to failures [25]. As in Figure 4, an active client crashes with probability q_f, and a failed client recovers with probability q_r. Here $q_r, q_f > 0$. Suppose there are m recruited clients. Let A^t be the number of active clients at iteration t, which has the following properties.

Proposition 1. *A^t is an ergodic Markov chain. In the steady state with m recruited clients, the probability that there are i active clients equals*

$$\mathbb{P}(A^\infty = i) = \frac{\binom{m}{i}(q_r/q_f)^i}{(1+q_r/q_f)^m} \tag{9}$$

Proof. The transition probability is given by

$$\begin{aligned} &P_{ij} = \mathbb{P}(A^{t+1} = j | A^t = i) \\ &= \sum_k \binom{i}{k}(1-q_f)^k q_f^{i-k}\binom{m-i}{j-k} q_r^{j-k}(1-q_r)^{m-i-j+k} \end{aligned} \tag{10}$$

Here the summation is taken from $k = \max\{0, i+j-m\}$ to $k = \min\{i, j\}$. This conditional probability P_{ij} does not depend on t, so A^t is a homogeneous Markov chain. Furthermore, it is easy to check that $\max\{0, i+j-m\} \leq \min\{i, j\}$ given that $0 \leq i, j \leq m$. Therefore, A^t is positive recurrent since $P_{ij} > 0$ for all pairs of states (i, j). In addition, A^t is aperiodic since there is a self-loop for every state i: $P_{ii} > 0$. As a result, A^t is ergodic.

Besides, it is easy to verify that for all (i, j), we have

$$\frac{P_{ij}}{P_{ji}} = \frac{P_{i0}P_{0j}}{P_{j0}P_{0i}} \tag{11}$$

Let $\pi = \{\mathbb{P}(A^\infty = i)\}_i$ be the steady state vector, and $\boldsymbol{P} = [P_{ij}]_{ij}$ be the transition matrix. It can be shown that with (11), the solution of the steady state equations $\pi \boldsymbol{P} = \pi$ satisfies:

$$\pi_i = \frac{P_{0i}}{P_{i0}}\pi_0 \tag{12}$$

Combining (10,12), and using the condition that $\sum_{i=0}^m \pi_i = 1$, we can get (9). □

Since ergodic Markov chains converge exponentially fast, we only consider the steady state. The probability that no clients fail is thus $\left(\frac{q_r}{q_f+q_r}\right)^m$.

In practice, clients have divergent specifications, e.g., different types of CPUs, so they may not have the same failure and recovery rates. To model this heterogeneity, we partition clients into N groups according to their device specifications, network qualities, battery levels, etc. In practice, we may use the network interface (i.e., WiFi or cellular) operating system or CPU speed to group clients, depending on which resource (e.g., computing or communication) bottlenecks the completion time of an update. For example, clients on laptops/desktops connected through WiFi that meet a minimum bandwidth requirement can be grouped together, while smartphones with the same manufacturing specification and identical cellular signal strength may form another group.

Suppose each group z has m_z clients for $z = 1, ..., N$, and all the clients inside a group z have the same failure rate q_f^z and recovery rate q_r^z. Since clients are running independently, the probability that all clients in all groups are active is then $\prod_{z=1}^N \left(\frac{q_r^z}{q_f^z+q_r^z}\right)^{m_z}$.

When a client k from group z is active, we model its per-iteration runtime (the time for the client to calculate and upload its local weight, measured from the time it receives the last global weight) as a random exponentially distributed

variable $Y_k^z \sim \exp(\lambda^z)$. The expected full iteration runtime (from the time the coordinator sends the global weight to the time it sends the next global weight) when all clients are active is: $\Gamma(m_1, ...m_N) = \mathbb{E}[\min\{\max_z \max_k Y_k^z, E_0\}]$, where E_0 is the maximum time of each iteration that the coordinator waits up to, and we should have $E_0 \geq \Gamma$.

We omit the coordinator's processing time as it is generally negligible compared to client times. The completion time is then given in (13), where m_z depends on our choice x of recruited clients.

$$\begin{aligned} &g_t(x) = g_t(m_1, ..., m_N) \\ =&T\left(\Gamma \prod_{z=1}^{N} (\frac{q_r^z}{q_f^z + q_r^z})^{m_z} + E_0(1 - \prod_{z=1}^{N} (\frac{q_r^z}{q_f^z + q_r^z})^{m_z})\right) \end{aligned} \tag{13}$$

Intuitively, the completion time increases when we recruit more clients. This is summarized in Proposition 2. In Section 6, we will use this proposition to relax the completion time constraint.

Proposition 2. *The completion time $g_t(m_1, ...m_N)$ increases when any $m_z, z = 1, ..., N$ increases.*

Proof. Let $\mathbf{1}_X$ represent an indicator function which equals 1 when X is true. We define random variable $Y = \min\{\max_z \max_k Y_k^z, E_0\}$ of which the expectation is Γ used in (13). The cumulative distribution function of Y is

$$F_Y(Y \leq y) = \mathbf{1}_{y \leq E_0} \prod_{z=1}^{N} (1 - e^{-\lambda_z y})^{m_z} + \mathbf{1}_{y > E_0} \tag{14}$$

Since $Y \geq 0$, we have

$$\begin{aligned} \Gamma(m_1, ..., m_N) = \mathbb{E}[Y] &= \int_0^{\infty} 1 - F_y(y) dy \\ &= \int_0^{E_0} 1 - \prod_{z=1}^{N} (1 - e^{-\lambda_z y})^{m_z} dy \end{aligned} \tag{15}$$

Because F_y is continuous w.r.t. $(y, m_1, ..., m_N)$, we get

$$\frac{\partial \Gamma}{\partial m_z} = -\int_0^{E_0} \log(1 - e^{-\lambda_z y}) \prod_{z=1}^{N} (1 - e^{-\lambda_z y})^{m_z} dy > 0 \tag{16}$$

Thus,

$$\begin{aligned} \frac{\partial g_t}{T \partial m_z} =& \frac{\partial \Gamma}{\partial m_z} \prod_{z=1}^{N} (\frac{q_r^z}{q_f^z + q_r^z})^{m_z} + \\ & (\Gamma - E_0) \prod_{z=1}^{N} (\frac{q_r^z}{q_f^z + q_r^z})^{m_z} \log(\frac{q_r^z}{q_f^z + q_r^z}) > 0 \end{aligned} \tag{17}$$

Therefore, g_t increases when any m_z increases. □

Finally, we consider the cost constraint. The cost of usage depends on the specific payment mechanisms actually adopted. For example, clients can auction their data to operators as in [16]. Alternatively, operators can publish their willingness to pay for each client's participation and let the clients decide whether to participate. We thus assume a generic scenario where each client k has an exogenous price c_k, so g_c becomes:

$$g_c(x) = \sum\nolimits_k^K x_k c_k \leq I_c \tag{18}$$

This price c_k may be adjusted upward or downward according to how much the user values privacy, after adding noise to the model updates so as to achieve a certain level of differential privacy guarantee. Since the noise required to achieve such a guarantee is independent of other users' participation in the training [14], we take this noise level and the associated user cost as fixed and given.

6 The Optimal Client Recruitment

From Section 5, each client $U_k \in \mathcal{U}, k = 1, .., K$ can be characterized by a tuple $(|\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}|, n_k, \mathcal{Z}(k), c_k)$, representing respectively the distribution divergence, the local dataset size, the group number, and the ask price. Clients in a group z have failure rate q_f^z, recovery rate q_r^z and processing rate λ^z. As discussed above, the client can readily compute this information and send it to the operator without significant privacy loss at the start of the recruitment.

Combining (8), (13), and (18), Problem 1 becomes:

Problem 2. *Client Recruitment*

$$\begin{aligned} \min_{x \in \{0,1\}^K} \quad & f(x) = \frac{1}{n_x} \sum_k x_k n_k s_k + n_x^{-0.5} \\ s.t. \quad & g_t(x) = (\Gamma - E_0) \prod_{z=1}^{N} \left(\frac{q_r^z}{q_f^z + q_r^z}\right)^{m_z} + E_0 \leq \frac{I_t}{T} \\ & g_c(x) = \sum\nolimits_k^K x_k c_k \leq I_c \end{aligned}$$

Proposition 3. *Problem 2 is NP-Hard.*

Proof. Let $I_t = \infty, s_k = 0$, Problem 2 is then equivalent to maximizing $n_x = \sum x_k n_k$ subject to a linear constraint g_c. This constructs a reduction from the classic Knapsack problem to Problem 2, which shows the problem is NP-Hard. □

6.1 Unconstrained Optimization

We first consider the unconstrained version of Problem 2, i.e., when all the limits I_t, I_c approach infinity. This is useful when the operator has gained complete right of usage of the clients (so that they can be used for free without time limit). This unconstrained optimization can be solved in polynomial time using the following proposition:

Proposition 4. *(Unconstrained Client Recruitment) Suppose clients are sorted by their s values, i.e. $s_1 \leq ... \leq s_K$. The solution to Problem 2 without constraints must be of the form: $x^* = (1, 1, ..., 1, 0, 0, ...0)$, i.e., if a client j is recruited, all the clients $k < j$ must also be recruited.*

Proposition 4 indicates that recruiting more clients does not always help improve the accuracy. Intuitively, when more clients are participating, the overall dataset grows larger, and the representativeness should thus improve. However, a recruited dataset $\mathcal{D}_k$ itself may be small in size, making its data biased from the local population $\mathcal{P}_k$ it is supposed to represent. This will enlarge its generalization error. Worse still, if $\mathcal{P}_k$ is also biased from the population $\mathcal{P}$, the training loss will increase as well due to the increased divergences in local distributions.

Based on the proposition, we can get a polynomial time greedy algorithm as in Algorithm 1. The correctness of Algorithm 1 is obvious. The complexity is dominated by the sorting step, which is $O(K \log K)$.

To prove Proposition 4, we first introduce the following lemma.

Lemma 3. *Consider two recruitments x^0 and x^j that contain the same set of clients, except that the latter includes client j while the former does not. If $f(x^j) \leq f(x^0)$ and we take $\{s_k\}$'s as fixed, then $f(x^j)$ decreases as we increase n_j, the number of data points from j.*

Proof. (Lemma 3) For convenience we rewrite $f(x^j) = f^j(n_j)$, and we consider general representativeness bound $n_x^{-\beta}, 0 < \beta < 1$. Note that

$$f^j(n_j) = \frac{\sum_k x_k^0 n_k s_k + n_j s_j}{\sum_k x_k^0 n_k + n_j} + \big(\sum_k x_k^0 n_k + n_j\big)^{-\beta} \quad (19)$$

$$\frac{df^j}{dn_j} = \frac{\sum_k x_k^0 n_k (s_j - s_k)}{(\sum_k x_k^0 n_k + n_j)^2} - \frac{\beta}{(\sum_k x_k^0 n_k + n_j)^{1+\beta}} \quad (20)$$

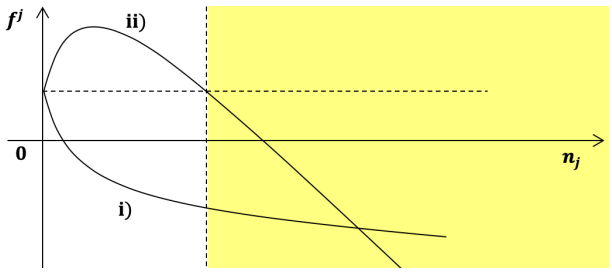


Fig. 5. Two possible cases of f^j: i) strictly decreasing, or ii) first increasing then decreasing. Since $f^j(n_j) = f(x^j) \leq f(x^0) = f^j(0)$, n_j can only be in the shaded area. Thus, further increasing n_j will only cause the objective f to decrease.

As n_j increases, (20) is either i) strictly negative, or ii) first positive then negative. Thus, (19) will either i) strictly decrease, or ii) first increase then decrease. Using the condition $f^j(n_j) = f(x^j) \leq f(x^0) = f^j(0)$, the value of n_j must fall into the decreasing interval (Figure 5). Therefore, further increasing n_j will only cause the objective f to decrease. □

Lemma 3 implies that when a client is recruited to reduce f, the accuracy improves if this client collects even more data. However, the statement may not hold for unrecruited clients. For example, if client j is not recruited due to its large s_j value, adding client j to the recruitment and increasing n_j may cause the objective to increase compared to not including client j.

We then prove Proposition 4:

Proof. (Proposition 4) We prove by contradiction. Assume the clients are already sorted by the s value. Suppose the optimal recruitment $x^* = x^j$, where client j is the last recruited client, and there exists at least one unrecruited client U_i, such that $i < j, x_i^j = 0$. Denote by $f(x|U(s,n))$ the objective value for recruiting clients in x, plus an additional client U who has parameters s and n. Let x^0 be a copy of x^j, except that client j is not recruited. We thus have $f(x^*) = f(x^j) \leq f(x^0)$, and:

$$\begin{aligned} &f(x^j|U(s_i,n_i)) \leq f(x^j|U(s_j,n_i)) \\ =&f(x^0|U(s_j,n_i+n_j)) < f(x^0|U(s_j,n_j)) = f(x^j) \end{aligned} \quad (21)$$

The first inequality is due to the condition $s_i \leq s_j$. The second follows from Lemma 3 with the fact that $f(x^j) \leq f(x^0)$. Therefore, adding the unrecruited client i to the recruitment x^j results in a smaller objective value $f(x^j|U(s_i,n_i))$, contradicting that $x^* = x^j$. □

Algorithm 1 *Greedy Recruitment.* Solving Problem 2 without constraints.

```
procedure OPTIMIZE
    sort clients according to s_k (ascending)
    f* ← ∞
    for j = 1, 2, ..., K do
        x^j ← (first j elements are 1, others 0)
        f* ← min(f*, f(x^j))
    return f*
```

6.2 Constrained Optimization

As we would expect from our NP-hardness result (Proposition 3), Proposition 4 does not hold when incorporating the constraints. To solve the constrained optimization Problem 2, we first relax the completion time constraint $g_t \leq I_t$ by N linear constraints $\mathcal{G}_t(m_1,...,m_N) = \{m_z \leq M_t^z\}_{z=1}^N$ on m_z, which essentially means that we recruit at most M_t^z clients for each group z, formalized as follows.

$$\begin{aligned} M_t^z = \min\Big\{ &\sum_{k=1}^K \mathbf{1}(\mathcal{Z}(k) = z), \\ &\text{argmax}_{m_z}\{g_t(0,...,0,m_z,0,...,0) \leq I_t\}\Big\} \end{aligned} \quad (22)$$

According to Proposition 2, if $(m_1,...,m_N)$ satisfies the original completion time constraint $g_t(m_1,...m_N) \leq I_t$, it also satisfies the relaxed constraint $\mathcal{G}_t(m_1,...,m_N)$. We then construct a new optimization Problem 3. Here we define $s'_k = n_k s_k, I_s = \sum_k^K s'_k$. Problem 3 maximizes a linear objective, subject to $N+2$ linear constraints. This is a multi-dimensional Knapsack problem, and can be solved by the dynamic programming (DP) algorithm [19].

Problem 3. *Data Quantity Maximization*

$$\begin{aligned} \max_{x\in\{0,1\}^K} \quad & n_x = \sum\nolimits_k x_k n_k \\ s.t. \quad & m_z = \sum\nolimits_k \mathbf{1}(\mathcal{Z}(k) = z) x_k \leq M_t^z, z = 1,...,N \\ & g_c(x) = \sum\nolimits_k x_k c_k \leq I_c, \;\; g_s(x) = \sum\nolimits_k x_k s'_k \leq I_s \end{aligned}$$

As in conventional DP procedures, we construct a $N+3$ dimensional table $\phi(k, m_1, ..., m_N, c, s)$ to keep track of the algorithm states, where we have $c \triangleq \sum_k x_k c_k$ and $s \triangleq \sum_k x_k s'_k$. $\phi(k, m_1, ..., m_N, c, s)$ represents the maximum value of n_x we can get, under the conditions that: 1) we only pick from the first k clients (the order of clients does not matter); 2) we recruit at most m_z clients for each group z; 3) the cost is less than or equal to c; and 4) the sum $\sum s'_k \leq s$. Conditions 2) to 4) correspond to the three constraints in Problem 3. The DP algorithm gradually increments the recruitment boundary k. For each k, the following recursive relation guarantees the consistency of ϕ:

$$\begin{aligned} \phi(k, m_1, ..., m_N, c, s) = \max\{&\phi(k-1, ..., m_{\mathcal{Z}(k)} - 1, \\ &..., c - c_k, s - s'_k) + n_k, \phi(k-1, m_1, ..., m_N, c, s)\} \end{aligned} \quad (23)$$

Complete pseudo code is presented in Algorithm 2. In practice, the values of c, s may be float numbers, but we can easily discretize and normalize them to integers. The correctness of the algorithm is obvious by induction. Since c and s are only used in sorting operations or in optimization constraints, discretization will only affect the optimality of our solution if we miss their optimal values in the DP table. With sufficiently fine quantization granularity, this is unlikely to happen; moreover, since the DP problem is formulated based on estimates of client statistics, we expect that natural errors in these estimates will dominate any quantization error. Algorithm 2's complexity is bounded by the size of the DP table, which is $O(KI_cI_s\prod_{z=1}^{N} M_t^z)$.

We show in Section 7 that Algorithm 2 completes in reasonable wall clock time. As opposed to client selection, which is conducted in every iteration, the client recruitment algorithm only needs to be run once before the training and can be completed by powerful servers. On the other hand, client recruitment helps control the completion time for each federated learning round by choosing better devices, which has greater impact on the overall time consumption since it scales with T, the number of learning rounds.

Algorithm 2 *Dynamic Programming*. Solving Problem 3.

procedure OPTIMIZE
  $\phi \leftarrow N+3$ dimensional empty table
  // Initialization
  for $m_1 = 0$ to M_t^1 **do**
    .....
    for $m_N = 0$ to M_t^N **do**
      for $c = 0$ to I_c **do**
        for $s = 0$ to I_s **do**
          $\phi(0, m_1, ..., m_N, c, s) \leftarrow 0$
  // Fill out the DP table
  for $k = 1$ to K **do**
    for $m_1 = 1$ to M_t^1 **do**
      .....
      for $m_N = 1$ to M_t^N **do**
        for $c = 0$ to I_c **do**
          for $s = 0$ to I_s **do**
            Update ϕ by (23)
  return $\phi(K, M_t^1, ..., M_t^N, I_c, I_s)$

Algorithm 3 *DP and Revisit*. Solving Problem 2.

procedure OPTIMIZE
  $\phi \leftarrow$ (solve Problem 3 with DP), $f^* \leftarrow \infty$
  if $\phi(K, M_1^t, ..., M_N^t, I_c, I_s) \leq 0$ **then**
    // Infeasible
    return ∞
  for $s = 0$ to I_s **do**
    for $m_1 = 0$ to M_t^1 **do**
      .....
      for $m_N = 0$ to M_t^N **do**
        if $g_t(m_1, ..., m_N) \leq I_t$ **then**
          $f^* \leftarrow \min(f^*,$ Equation (24))
  return f^*

Now we go back to the original Problem 2. We can observe that when the value of s is fixed, minimizing the objective f is equivalent to maximizing the number of samples n_x. Since the ϕ table records a one-to-one mapping of s to the maximum n_x, we can utilize ϕ to reconstruct the original objective f.

Formally, given $(m_1, ..., m_N, s)$, we define

$$f' = \frac{s}{\phi(K, m_1, ..., m_N, I_c, s)} + (\phi(K, m_1, ..., m_N, I_c, s))^{-0.5} \tag{24}$$

Intuitively, for a solver x^* of Problem 2, if its corresponding s^* and m_z^* are recorded during the DP iteration, then since ϕ represents the summation of recruited data points n_x, plugging s^* and m_z^*'s into f' to replace s and m_z's should produce a value that is close enough to the optimal objective value $f(x^*)$. We show below that the two values are in fact identical, which leads us to solve the constrained client recruitment Problem 2 with Algorithm 3.

Proposition 5. *Algorithm 3 solves Problem 2.*

Proof. Consider the general representativeness bound $n_x^{-\beta}$. Let x^* be a solver of Problem 2, with $n_x^* = \sum_k x_k^* n_k, s^* = \sum_k x_k^* n_k s_k, m_z^* = \sum_k \mathbf{1}(\mathcal{Z}(k) = z) x_k^*$, then

$$\phi(K, m_1^*, ..., m_N^*, I_c, s^*) = n_x^* \tag{25}$$

Otherwise, if the left hand side is smaller, the DP algorithm yields a smaller objective n_x^0 for some recruitment decision x^0. Both x^0 and x^* satisfy the four conditions in the definition of ϕ at $(K, m_1^*, ..., m_N^*, I_c, s^*)$. But replacing x^0 with x^* yields a greater objective $n_x^* > n_x^0$. This contradicts the correctness of DP. In addition, if the left hand side is greater, the DP algorithm finds a recruitment x^0 that has $s^0 = \sum x_k^0 n_k s_k \leq s^*$, $n_x^0 = \sum x_k^0 n_k > n^*$, and satisfies all the constraints in Problem 2. Thus, by recruitment x^0, we have $f(x^0) = \frac{s^0}{n_x^0} + (n_x^0)^{-\beta} < \frac{s^*}{n_x^*} + (n_x^*)^{-\beta} = f(x^*)$. This shows x^0 is a better recruitment decision than x^*, which contradicts the assumption that x^* is an optimal. Thus, since Algorithm 3 iterates through all the feasible elements, we must at some point visit $(K, m_1^*, ..., m_N^*, I_c, s^*)$. □

The complexity of Algorithm 3 is dominated by the DP step and thus is also $O(KI_cI_s\prod_{z=1}^{N} M_t^z)$.

7 Performance Evaluation

We finally evaluate the performance of our client recruitment strategy with two classification problems and a regression problem. We set the aggregation deadline $E_0 = 30$. Unless otherwise noted, we assume clients have the default specification: Group I = $(q_f^1 = 0.001, q_r^1 = 0.6, \lambda^1 = 0.1)$. If a client fails upon the aggregation, we replace its $w_k^{t\tau}$ with the previous global weight $w^{t\tau}$. Throughout this section, we uniformly at random set the cost of recruiting each client c_k in the range of 1 to 9.

We consider three baseline recruitment strategies:

- **All participation:** recruiting all clients. Comparisons with this baseline show the value of intelligent client recruitment.
- **Greedy recruiting by quantity:** greedily choosing clients with the most data samples until any constraint is active.
- **Greedy recruiting by quality:** greedily choosing clients with best quality (i.e., highest estimated $|\tilde{P}_k - P|$) until any constraint is active.

Comparisons to these baselines show the value of considering both quality and quantity in the client recruitment.

In the case of unconstrained optimization, we force the greedy baselines to choose the same number of clients as the optimal recruitment. By comparing to the latter two baselines, we present the value of considering both quantity and quality of the data. Since we take the training parameters as fixed as discussed in Section 3, we will not fine-tune them in the simulation. We pick these values such that all model training in all experiments are fully converged. We then only need to determine the relative weights of the training accuracy (γ_{tl}) and generalizability (γ_{ge}). In practice, they can be tuned by optimizing the unconstrained recruitment through grid search.

7.1 Image Classifications

We use both the MNIST dataset with 10 labels corresponding to digits 0 to 10, and the more complicated EMNIST dataset with 26 labels corresponding to lowercase English letter images a to z. We use the same CNN models as [27]. All clients are equipped with the Adam optimizer and use the same set of training parameters. The number of global epochs $T = 50$. For MNIST, we use a batch size of 5 for local iterations. The initial learning rate is set to 2e-5, and the local epochs $\tau = 10$. For EMNIST, the batch size equals 5 and the initial learning rate is 1e-5, and $\tau = 30$. In addition, we perform random client selection in the MNIST experiments with a selection ratio of 90%, i.e., 90% of the recruited clients are randomly awakened in every communication round.

Dataset and clients. To construct the non-IID distributions of local datasets, we assign each client a set of labels (digits and letters). All labels appear with the same probability. Clients then randomly sample training images corresponding to the assigned labels. For MNIST, we limit each client to sample 50 to 100 images. For $j \in \{2, 4, 6, 8\}$, we assign j label(s) to 30 new clients, resulting in total 120 candidate clients. To solve the more challenging EMNIST problem, we allow each client to sample 50 to 300 images. To reflect the sparsity of high-quality clients in the real world. We assign 1 label to 30 clients, 10 labels to 20 clients, 15 labels to 10 clients, 20 labels to 10 clients, resulting in 70 clients. The 26 letters are drawn at random for each assignment. The default test sets are used for both datasets.

Approximation of divergence. We use the "counting classes" method described in Section 5.1 to approximate the probability divergence. For MNIST, we have $L = 10$ labels, thus $\int |\tilde{p}_k - \tilde{p}| = \sum_{i=0}^{9} |\tilde{p}_k(v = i) - \tilde{p}(v = i)|$. For the population distribution, all labels appear with the same probability, so $\tilde{p}(v = i) \equiv 0.1$. If a client k has j labels, $\tilde{p}_k(v = i) = 1/j$ if label i was assigned, or $\tilde{p}_k(v = i) = 0$ otherwise. Thus, $\sum |\tilde{p}_k(v = i) - \tilde{p}(v = i)| = j(\frac{1}{j} - \frac{1}{10}) + (10 - j)\frac{1}{10} = 2 - \frac{j}{5}$, which clients can easily compute knowing only the number of labels that they see. Similarly, we have $\sum |\tilde{p}_k(v = i) - \tilde{p}(v = i)| = 2 - \frac{j}{13}$ for EMNIST. By tuning the unconstrained recruitment, we choose $\gamma_{tl} = 0.015, \gamma_{ge} = 1$ for all experiments.

Unconstrained recruitment. The left plot of Figure 6 shows the convergence progress of the four recruitment strategies for MNIST. There are 44 out of the 120 clients recruited by the optimal strategy in this setting. The optimal

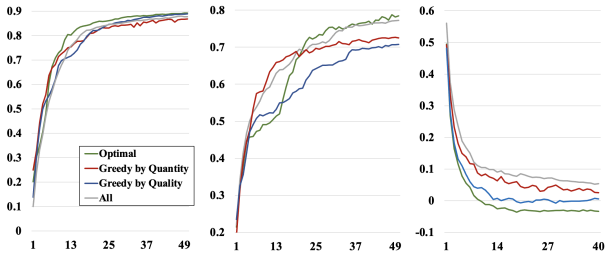


Fig. 6. Convergence curves for the unconstrained recruitments. Left and middle plots are respectively MNIST and EMNIST problems. The X axis is global epochs, and the Y axis is test accuracy. The optimal strategy's model yields higher test accuracy than other baselines after the models are sufficiently trained. The right plot is for the regression problem. The X axis is global epochs, and the Y axis is the normalized MSE on the test dataset. The untrained model has MSE=1, and the closed-form solution has MSE=0. The optimal recruitment can obtain lower MSE than the closed-form solution.

recruitment converges the fastest and obtains the highest test accuracy on the fully trained models. Notably, the optimal strategy converges faster and yields a fairly higher final test accuracy. We can observe similar performance gaps in the middle plot for EMNIST data, where 21 clients are recruited by the optimal strategy. Note that the "Greedy by Quality" baseline outperforms the "Greedy by Quantity" baseline for MNIST data, yet it yields lower performance for EMNIST data. This is because different training problems may favor divergent combinations of data quality and quantity, and our optimal recruitment can automatically balance these factors and produce the best solution.

Figure 7 shows the distribution of recruited clients w.r.t. the number of labels assigned to them for MNIST. Compared to the greedy-by-quantity strategy, the optimal recruitment chooses fewer low-quality datasets, but more high-quality ones. Also, most clients recruited by the optimal strategy have moderate data sizes, but the greedy-by-quality recruitment includes lots of tiny datasets. We observe this trend for both MNIST and EMNIST data.

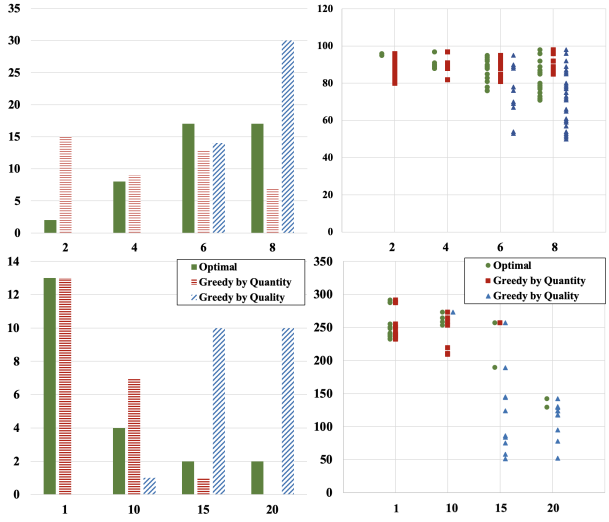


Fig. 7. The distribution of recruited clients w.r.t. the number of assigned labels (X axes). Left plot: counts of recruited clients. Right plot: sizes of local datasets for recruited clients, where each point represents a client. Top: MNIST. Bottom: EMNIST.

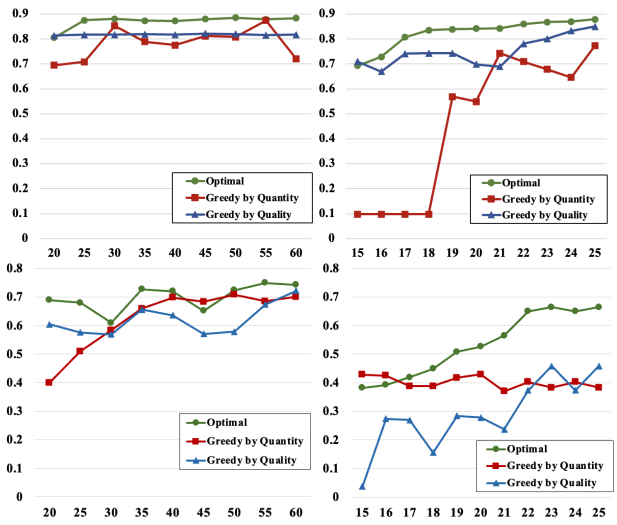


Fig. 8. Left: Test accuracy when varying the budget I_c from 20 to 60 (i.e. 4x to 12x of the expected cost per client). Right: Test accuracy when varying the per round completion time I_t/T from 15 to 25 (i.e. 1/2 to 5/6 of E_0). The optimal strategy consistently has the highest accuracy. Top: MNIST. Bottom: EMNIST.

Constrained recruitment. The left plot of Figure 8 shows the change of test accuracy when we increase the budget I_c from 20 to 60 and take I_t to be infinity. The optimal strategy obtains the highest accuracy for all the budgets I_c. In the right plot of Figure 8 we drop the I_c constraints and vary the completion time constraint I_t from $15T$ to $25T$. Apart from Group I, we also create a relatively lower-end specification Group II = ($q_f^2 = 0.01, q_r^2 = 0.5, \lambda^2 = 0.05$). We randomly pick one third of the clients and assign them to Group II. When I_t/T is down to around half of $E_0 = 30$, only 1 or 2 clients are recruited, so the models do not appear to be trained at all. The optimal strategy exhibits the best performance when I_t is reasonably large, improving the accuracy by at most 10% to 20% for both MNIST and EMNIST. The time constraint appears to have a greater impact on the convergence in Figure 8 because in our simulation, uncompleted clients do not submit their models for aggregation, which breaks the original training logic and incurs bias in the aggregation result [29]. The cost constraint on the other hand affects only the recruitment results (i.e. what clients are get involved), with no effect on the training loop.

7.2 Climate Data Regression

We now evaluate client recruitment with a 5-dimensional linear regression model, simulating a climate prediction task. All clients use the Adam optimizer with the initial learning rate set to 1e-3, decaying by 0.8 every 200 steps. We set the batch size as 20, $\tau = 10$, and $T = 40$.

Dataset and clients. We use the U.S. Historical Climatology Network (HCN) dataset [21], which contains climate records for climate stations in the 48 contiguous United States. The local datasets of these stations are by their nature non-IID, allowing us to evaluate how well our recruitment algorithm performs on realistic data distributions. For simplicity, we only use the data on the first day of December from 1960 to 2019, and we randomly pick 1-3 stations from each state, resulting in 117 stations. Each record contains 5 features: station latitude, station longitude, lowest temperature of the day, highest temperature of the day, and precipitation of the day. Our goal is to predict the snowfall of the day. To reflect the uneven sizes of local datasets, we randomly drop some data so that each client has 30 to 69 samples. We test the learned models on a holdout dataset, which is generated by randomly picking 2 unused stations from each state.

Approximation of divergence. We use the second approximation method described in Section 5.1 by assuming the 5 features and the snowfall form a fully connected Gaussian graphic model $\mathcal{N}(\mu, \Sigma)$. Thus, each local distribution can be parameterized by the sample mean and the sample covariance $\mathcal{N}(\tilde{\mu}_k, \tilde{\Sigma}_k)$. Similarly, we approximate the population distribution $\mathcal{N}(\tilde{\mu}, \tilde{\Sigma})$ utilizing the unused (neither training nor testing) data. Thus, we only need to compute the divergence between the local Gaussian $\mathcal{N}(\tilde{\mu}_k, \tilde{\Sigma}_k)$ and population Gaussian $\mathcal{N}(\tilde{\mu}, \tilde{\Sigma})$. We normalize the divergences to the range of 0 and 10, and we choose the coefficients $\gamma_{tl} = 0.01$, $\gamma_{ge} = 1$.

Unconstrained recruitment. The right plot in Figure 6 shows the mean-squared error (MSE) on the holdout dataset, which includes 1-2 stations from each state, for different strategies. 37 clients are chosen by the optimal recruitment, allowing us to drop most clients as in Section 5.1. Since linear regression is a convex problem, we can easily calculate the closed-form optimal model that minimizes the training loss over the full dataset. For ease of comparison, we normalize the MSE values so that the untrained model has MSE equal to 1, and the closed-form solution has MSE equal to 0. As in Figure 6, our optimal recruitment yields a lower MSE even than the closed-form solution, which illustrates the value of incorporating generalizability and representativeness metrics. Compared to other strategies, the optimal recruitment can decrease the MSE up to 10%.

Similar to Figure 7, Figure 9 shows the distribution of recruited clients. Here we divide the clients based on their local-population distribution divergences into 10 bins. We observe a similar trend as in Figure 7: the optimal method chooses mostly high-quality clients, but chooses clients with larger datasets than the greedy-by-quality method. The greedy-by-quantity method chooses only clients with large datasets, without regard to the distribution divergence.

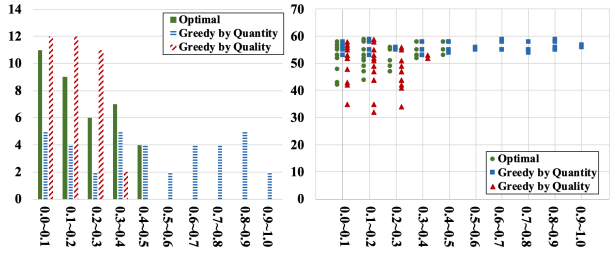


Fig. 9. The distribution (left: client count; right: dataset size per client) of recruited clients w.r.t. the distribution divergence. X axes are quantile ranges. Left bins correspond to small divergence (i.e. good quality).

Constrained recruitment. Figure 10 shows the change of MSE when varying the cost and time limits, on the same

setup as in Section 7.1. The optimal recruitment obtains the lowest MSE and much smaller variance in most cases.

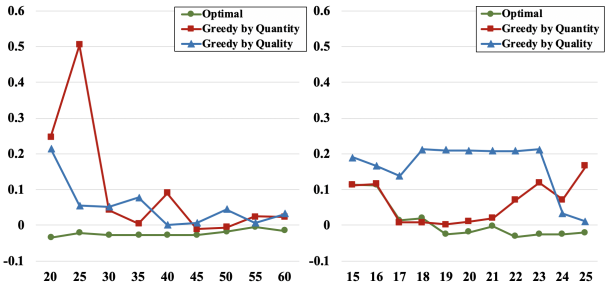


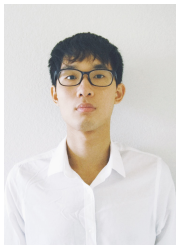
Fig. 10. Left: Test MSE when varying the budget I_c from 20 to 60. Right: Test MSE when varying I_t/T from 15 to 25. The optimal strategy consistently has the lowest error.

8 Conclusion

This paper studies the client recruitment problem in federated learning. We first introduce and quantify five performance metrics that cover both the model's accuracy (training loss, generalization error, representativeness) and the training efficiency (completion time, cost). We then formulate the client recruitment as an NP-Hard optimization problem and provide an optimal solution algorithm. Finally, we verify our theoretical results with experiments using both synthetic and real-world data. Our results show, somewhat counter-intuitively, that recruiting more clients does not always improve the model, and intelligent client recruitment can greatly improve the accuracy of the trained model in constrained execution environments. Future work may integrate this work with client selection methods to recruit a new set of clients over time, or co-optimize the recruitment with algorithm parameters. Inspired by learning-based client selection methods, we can also investigate bandit or reinforcement learning based methods for learning which clients to recruit over time, e.g., if clients must be re-recruited every so often during the training and we improve our estimates of their data quality over time.

Acknowledgement

This work was supported by NSFC grant (No. 62102460), Guangzhou Science and Technology Plan Project (No. 202201011392), Guangdong Basic and Applied Basic Research Foundation (No. 2023A1515012982), Young Outstanding Award under the Zhujiang Talent Plan of Guangdong Province, and the National Science Foundation under awards CNS-1751075, CNS-2106891, and CNS-2312761.



Yichen Ruan is currently a research scientist at Meta Platforms, Inc. He earned his B.S. in Civil Engineering from Tsinghua University in 2016, his M.S. in Civil Systems from UC Berkeley in 2017, and his Ph.D. in Electrical and Computer Engineering from Carnegie Mellon University in 2022. His research focuses on federated learning and resource allocation in network systems.

Xiaoxi Zhang received the B.E. degree in electronics and information engineering from the Huazhong University of Science and Technology in 2013 and the Ph.D. degree in computer science from The University of Hong Kong in 2017. She is currently an Associate Professor with the School of Computer Science and Engineering, Sun Yat-sen University. Before joining SYSU, she was a Post-Doctoral Researcher with the Department of Electrical and Computer Engineering, Carnegie Mellon University. She is broadly interested in networked systems, including cloud and edge computing networks, distributed systems, and machine learning systems.

Carlee Joe-Wong (S'11, M'16, SM'22) is the Robert E. Doherty Associate Professor of Electrical and Computer Engineering at Carnegie Mellon University. She received her A.B. degree (magna cum laude) in Mathematics, and M.A. and Ph.D. degrees in Applied and Computational Mathematics, from Princeton University in 2011, 2013, and 2016, respectively. Her research interests lie in optimizing various types of networked systems, including applications of machine learning and pricing to cloud computing, mobile/wireless networks, and ridesharing networks. From 2013 to 2014, she was the Director of Advanced Research at DataMi, a startup she co-founded from her research on mobile data pricing. She received the NSF CAREER award in 2018 and the ARO Young Investigator award in 2019.

REFERENCES

[1] Operational support systems. https://aws.amazon.com/compliance/data-center/controls/, 2019. Accessed: 2020-08-13.

[2] Dea Delvia Arifin, Moch Arif Bijaksana, et al. Enhancing spam detection on mobile phone short message service (sms) performance using fp-growth and naive bayes classifier. In *2016 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*, pages 80–84. IEEE, 2016.

[3] Eugene Bagdasaryan, Andreas Veit, et al. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2018.

[4] Keith Bonawitz, Hubert Eichner, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[5] Theodora S Brisimi, Ruidi Chen, et al. Federated learning of predictive models from federated electronic health records. *International journal of medical informatics*, 112:59–67, 2018.

[6] Mingzhe Chen, Zhaohui Yang, et al. A joint learning and communications framework for federated learning over wireless networks. *arXiv preprint arXiv:1909.07972*, 2019.

[7] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *arXiv preprint arXiv:2010.01243*, 2020.

[8] S. Dutta, G. Joshi, et al. Slow and stale gradients can win the race: Error-runtime trade-offs in distributed sgd. *arXiv preprint arXiv:1803.01113*, 2018.

[9] Ahmed El Ouadrhiri and Ahmed Abdelhadi. Differential privacy for deep and federated learning: A survey. *IEEE Access*, 10:22359–22380, 2022.

[10] I Fazekas and AG Kukush. Asymptotic properties of an estimator in nonlinear functional errors-in-variables models with dependent error terms. *Computers & Mathematics with Applications*, 34(10):23–39, 1997.

[11] Sarah Henderson. Nist study evaluates effects of race, age, sex on face recognition software, 2019.

[12] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[13] Jiawen Kang, Zehui Xiong, Dusit Niyato, Han Yu, Ying-Chang Liang, and Dong In Kim. Incentive design for efficient federated learning in mobile networks: A contract theory approach. *arXiv preprint arXiv:1905.07479*, 2019.

[14] Muah Kim, Onur Günlü, and Rafael F Schaefer. Federated learning with local differential privacy: Trade-offs between privacy, utility, and communication. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2650–2654. IEEE, 2021.

[15] Jakub Konečnỳ, H Brendan McMahan, et al. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[16] C. Li, D. Li, et al. A theory of pricing private data. *Communications of the ACM*, 60(12):79–86, 2017.

[17] Xiang Li, Kaixuan Huang, et al. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[18] Y. Lu et al. Blockchain and federated learning for privacy-preserved data sharing in industrial iot. *IEEE Transactions on Industrial Informatics*, 2019.

[19] Silvano Martello. Knapsack problems: algorithms and computer implementations. *Wiley-Interscience series in discrete mathematics and optimiza tion*, 1990.

[20] H Brendan McMahan, Eider Moore, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[21] Matthew J Menne, Imke Durre, et al. An overview of the global historical climatology network-daily database. *Journal of Atmospheric and Oceanic Technology*, 29(7):897–910, 2012.

[22] Luis Muñoz-González, Kenneth T Co, and Emil C Lupu. Byzantine-robust federated machine learning through adaptive model averaging. *arXiv preprint arXiv:1909.05125*, 2019.

[23] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC*, pages 1–7. IEEE, 2019.

[24] Shashi Raj Pandey, Nguyen H Tran, et al. A crowdsourcing framework for on-device federated learning. *arXiv preprint arXiv:1911.01046*, 2019.

[25] Parisa Rahimzadeh, Youngbin Im, et al. Echo: Efficiently overbooking applications to create a highly available cloud. In *ICDCS*, pages 1–11. IEEE, 2019.

[26] BLS Prakasa Rao. The rate of convergence of the least squares estimator in a non-linear regression model with dependent errors. *Journal of multivariate analysis*, 14(3):315–322, 1984.

[27] Yichen Ruan and Carlee Joe-Wong. Fedsoft: Soft clustered federated learning with proximal local updating. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 8124–8131, 2022.

[28] Yichen Ruan, Xiaoxi Zhang, and Carlee Joe-Wong. How valuable is your data? optimizing client recruitment in federated learning. In *2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*, pages 1–8. IEEE, 2021.

[29] Yichen Ruan, Xiaoxi Zhang, Shu-Che Liang, and Carlee Joe-Wong. Towards flexible device participation in federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3403–3411. PMLR, 2021.

[30] Anit Kumar Sahu, Tian Li, et al. On the convergence of federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

[31] S. Samarakoon et al. Distributed federated learning for ultra-reliable low-latency vehicular communications. *IEEE Transactions on Communications*, 2019.

[32] Bahar Sateli, Gina Cook, et al. Smarter mobile apps through integrated natural language processing services. In *International Conference on Mobile Web and Information Systems*, pages 187–202. Springer, 2013.

[33] Virginia Smith, Chao-Kai Chiang, et al. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434, 2017.

[34] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. Ldp-fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, pages 61–66, 2020.

[35] Yuwei Tu et al. Network-aware optimization of distributed learning for fog computing. *INFOCOM*, 2019.

[36] T. Tuor et al. Data selection for federated learning with relevant and irrelevant data at clients. *arXiv preprint arXiv:2001.08300*, 2020.

[37] Tiffany Tuor, Shiqiang Wang, Bong Jun Ko, Changchang Liu, and Kin K Leung. Overcoming noisy and irrelevant data in federated learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 5020–5027. IEEE, 2021.

[38] Shiqiang Wang, Tiffany Tuor, et al. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, 37(6):1205–1221, 2019.

[39] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[40] G. Joshi X. Zhang, J. Wang and C. Joe-Wong. Machine learning on volatile instances. In *INFOCOM*. IEEE, 2019.

[41] Howard H Yang, Zuozhu Liu, et al. Scheduling policies for federated learning in wireless networks. *IEEE Transactions on Communications*, 2019.

[42] Timothy Yang, Galen Andrew, et al. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

[43] Yue Zhao, Meng Li, et al. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.