FEDERATED OPTIMIZATION IN HETEROGENEOUS NETWORKS

Anonymous Authors[1]

ABSTRACT

Federated Learning is a distributed learning paradigm with two key challenges that differentiate it from traditional distributed optimization: (1) significant variability in terms of the systems characteristics on each device in the network (systems heterogeneity), and (2) non-identically distributed data across the network (statistical heterogeneity). In this work, we introduce a framework, `FedProx`, to tackle heterogeneity in federated networks. `FedProx` can be viewed as a generalization and re-parametrization of `FedAvg`, the current state-of-the-art method for federated learning. While this re-parameterization makes only minor modifications to the method itself, these modifications have important ramifications both in theory and in practice. Theoretically, we provide convergence guarantees for our framework when learning over data from non-identical distributions (statistical heterogeneity), and while adhering to device-level systems constraints by allowing each participating device to perform a variable amount of work (systems heterogeneity). Practically, we demonstrate that `FedProx` allows for more robust convergence than `FedAvg` across a suite of realistic federated datasets. In particular, in highly heterogeneous settings, `FedProx` demonstrates significantly more stable and accurate convergence behavior relative to `FedAvg`—improving absolute test accuracy by 18.8% on average.

1 INTRODUCTION

Federated learning has emerged as an attractive paradigm for distributing training of machine learning models in networks of remote devices. While there is a wealth of work on distributed optimization in the context of machine learning, two key challenges that distinguish federated learning from traditional distributed optimization are high degrees of *systems and statistical heterogeneity*[1] (Li et al., 2019; McMahan et al., 2017).

In an attempt to handle heterogeneity and tackle high communication costs, optimization methods that allow for local updating and low participation are a popular approach for federated learning (McMahan et al., 2017; Smith et al., 2017). In particular, `FedAvg` (McMahan et al., 2017) is an iterative method that has emerged as the de facto optimization method in the federated setting. At each iteration, `FedAvg` first locally performs E epochs of stochastic gradient descent (SGD) on K devices—where E is a small constant and K is a small fraction of the total devices in the network. The devices then communicate their model updates to a central server, where they are averaged.

While `FedAvg` has demonstrated empirical success in heterogeneous settings, it does not fully address the underlying challenges associated with heterogeneity. In the context of systems heterogeneity, `FedAvg` does not allow participating devices to perform variable amounts of local work based on their underlying systems constraints; instead it is common to simply drop devices that fail to compute E epochs within a specified time window (Bonawitz et al., 2019). From a statistical perspective, `FedAvg` has been shown to diverge empirically in some settings where the data is non-identically distributed across devices (e.g., McMahan et al., 2017, Sec 3). Unfortunately, `FedAvg` is difficult to analyze theoretically in such realistic scenarios and thus lacks convergence guarantees to characterize its behavior (see Section 2 for additional details).

In this work, we propose `FedProx`, a federated optimization algorithm that addresses the challenges of heterogeneity both theoretically and empirically. A key insight we have in developing `FedProx` is that an interplay exists between systems and statistical heterogeneity in federated learning. Indeed, both dropping stragglers (as in `FedAvg`) or naively incorporating partial information from stragglers (as in `FedProx` with the proximal term set to 0) implicitly increases statistical heterogeneity and can adversely impact convergence behavior. To mitigate this issue, we propose adding a proximal term to the objective that helps to improve

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the Systems and Machine Learning (SysML) Conference. Do not distribute.

[1]Privacy is a third key challenge in the federated setting. While not the focus of this work, standard privacy-preserving approaches such as differential privacy and secure multiparty communication can naturally be combined with the methods proposed herein.

the stability of the method. This term provides a principled way for the server to account for statistical heterogeneity associated with partial information. Theoretically, these modifications allow us to provide convergence guarantees for our method and to analyze the effect of heterogeneity. Empirically, we demonstrate that the modifications improve the stability and overall accuracy of federated learning in heterogeneous networks.

The remainder of this paper is organized as follows. In Section 2, we provide background on federated learning and an overview of related work. We then present our proposed framework, `FedProx` (Section 3), and derive convergence guarantees for the framework that account for both issues of statistical and systems heterogeneity (Section 4). Finally, in Section 5, we provide a thorough empirical evaluation of `FedProx` on a suite of synthetic and real-world federated datasets. Our empirical results help to illustrate and validate our theoretical analysis, and also demonstrate the practical improvements of `FedProx` over `FedAvg` in heterogeneous networks.

2 Background and Related Work

Large-scale machine learning, particularly in data center settings, has motivated the development of numerous distributed optimization methods in the past decade (see, e.g., Boyd et al., 2010; Dean et al., 2012; Dekel et al., 2012; Li et al., 2014a; Reddi et al., 2016; Richtárik & Takáč, 2016; Shamir et al., 2014; Smith et al., 2018; Zhang et al., 2015; 2013). However, as computing substrates such as phones, sensors, and wearable devices grow both in power and in popularity, it is increasingly attractive to learn statistical models directly in networks of distributed devices locally, as opposed to moving the data to the data center. This problem, known as federated learning, requires tackling novel challenges with privacy, heterogeneous data and devices, and massively distributed computational networks.

Recent optimization methods have been proposed that are tailored to the specific challenges in the federated setting. These methods have shown significant improvements over traditional distributed approaches like ADMM (Boyd et al., 2010) or mini-batch methods (Dekel et al., 2012) by allowing both for inexact local updating in order to balance communication vs. computation in large networks, and for a small subset of devices to be active at any communication round (McMahan et al., 2017; Smith et al., 2017). For example, Smith et al. (2017) propose a communication-efficient primal-dual optimization method that learns separate but related models for each device through a multi-task learning framework. Despite the theoretical guarantees and practical efficiency of the proposed method, such an approach is not generalizable to non-convex problems, e.g., deep learning, where strong duality is no longer guaranteed. In the non-convex setting, Federated Averaging (`FedAvg`), a heuristic method based on averaging local Stochastic Gradient Descent (SGD) updates in the primal, has instead been shown to work well empirically (McMahan et al., 2017).

Unfortunately, `FedAvg` is quite challenging to analyze due to its local updating scheme, the fact that few devices are active at each round, and the issue that data is frequently distributed in a heterogeneous nature in the network. In particular, as each device generates its own local data, *statistical heterogeneity* is common with data being non-identically distributed between devices. Recent works have made steps towards analyzing `FedAvg` in simpler, non-federated settings. For instance, parallel SGD and related variants (Lin et al., 2018; Reddi et al., 2016; Shamir et al., 2014; Stich, 2019; Wang & Joshi, 2018; Woodworth et al., 2018; Zhang et al., 2015; Zhou & Cong, 2018), which make local updates similar to `FedAvg`, have been studied in the IID setting. However, the results rely on the premise that each local solver is a copy of the same stochastic process (due to the IID assumption). This line of reasoning does not apply to the heterogeneous setting.

Although some works (Hao et al., 2019; Jiang & Agrawal, 2018; Wang et al., 2019; Yu et al., 2018) have recently explored convergence guarantees in heterogeneous settings, they make the limiting assumption that all devices participate in each round of communication, which is often infeasible in realistic federated networks (McMahan et al., 2017). Further, they rely on specific solvers to be used on each device (either SGD or GD), as compared to the solver-agnostic framework proposed herein, and add additional assumptions of convexity (Wang et al., 2019) or uniformly bounded gradients (Yu et al., 2018) to their analyses. There are also heuristic approaches that aim to tackle statistical heterogeneity, either by sharing the local device data or some server-side proxy data (Huang et al., 2018; Jeong et al., 2018; Zhao et al., 2018). However, these methods may be unrealistic: in addition to imposing burdens on network bandwidth, sending local data to the server (Jeong et al., 2018) violates the key privacy assumption of federated learning, and sending globally-shared proxy data to all devices (Huang et al., 2018; Zhao et al., 2018) requires effort to carefully generate or collect such auxiliary data.

Beyond statistical heterogeneity, *systems heterogeneity* is also a critical concern in federated networks. The storage, computational, and communication capabilities of each device in federated networks may differ due to variability in hardware (CPU, memory), network connectivity (3G, 4G, 5G, wifi), and power (battery level). These system-level characteristics dramatically exacerbate challenges such as straggler mitigation and fault tolerance. One strategy used in practice is to ignore the more constrained devices failing to complete a certain amount of training (Bonawitz et al.,

2019). However (as we demonstrate in Section 5), this can have negative effects on convergence as it limits the number of effective devices contributing to training, and may induce bias in the device sampling procedure if the dropped devices have specific data characteristics.

In this work, inspired by FedAvg, we explore a broader framework, FedProx, that is capable of handling heterogeneous federated environments while maintaining similar privacy and computational benefits. We analyze the convergence behavior of the framework through a local *statistical* dissimilarity characterization between local functions, while also taking into account practical *systems* constraints. Our characterization is inspired by the randomized Kaczmarz method for solving linear system of equations (Kaczmarz, 1993; Strohmer & Vershynin, 2009), a similar assumption of which has been used to analyze variants of SGD in other settings (see, e.g., Schmidt & Roux, 2013; Vaswani et al., 2019; Yin et al., 2018). Our proposed framework allows for improved robustness and stability of convergence in heterogeneous federated networks.

Finally, in terms of related work, we note that two aspects of our proposed work—the proximal term in FedProx and the bounded dissimilarity assumption used in our analysis—have been previously studied in the optimization literature, though often with very different motivations and in non-federated settings. For completeness, we provide a further discussion in Appendix B on this background work.

3 Federated Optimization: Methods

In this section, we introduce the key ingredients behind recent methods for federated learning, including FedAvg, and then outline our proposed framework, FedProx.

Federated learning methods (e.g., McMahan et al., 2017; Smith et al., 2017) are designed to handle multiple devices collecting data and a central server coordinating the global learning objective across the network. In particular, the aim is to minimize:

$$\min_w \; f(w) = \sum_{k=1}^{N} p_k F_k(w) = \mathbb{E}_k[F_k(w)], \qquad (1)$$

where N is the number of devices, $p_k \geq 0$, and $\sum_k p_k$=1. In general, the local objectives measure the local empirical risk over possibly differing data distributions $\mathcal{D}_k$, i.e., $F_k(w) := \mathbb{E}_{x_k \sim \mathcal{D}_k}[f_k(w; x_k)]$, with n_k samples available at each device k. Hence, we can set p_k=$\frac{n_k}{n}$, where n=$\sum_k n_k$ is the total number of data points. In this work, we consider $F_k(w)$ to be possibly non-convex.

To reduce communication, a common technique in federated optimization methods is that on each device, a *local objective function* based on the device's data is used as a surrogate for the global objective function. At each outer iteration, a subset of the devices are selected and *local solvers* are used to optimize the local objective functions on each of the selected devices. The devices then communicate their local model updates to the central server, which aggregates them and updates the global model accordingly. The key to allowing flexible performance in this scenario is that each of the local objectives can be solved *inexactly*. This allows the amount of local computation vs. communication to be tuned based on the number of local iterations that are performed (with additional local iterations corresponding to more exact local solutions). We introduce this notion formally below, as it will be utilized throughout the paper.

Definition 1 (γ-inexact solution)**.** For a function $h(w; w_0) = F(w) + \frac{\mu}{2}\|w - w_0\|^2$, and $\gamma \in [0, 1]$, we say w^* is a γ-inexact solution of $\min_w h(w; w_0)$ if $\|\nabla h(w^*; w_0)\| \leq \gamma \|\nabla h(w_0; w_0)\|$, where $\nabla h(w; w_0) = \nabla F(w) + \mu(w - w_0)$. Note that a smaller γ corresponds to higher accuracy.

We use γ-inexactness in our analysis (Section 4) to measure the amount of local computation from the local solver at each round. As discussed earlier, different devices are likely to make different progress towards solving the local subproblems due to variable systems conditions, it is therefore important to allow γ to vary both by device and by iteration. This is one of the motivations for our proposed framework discussed in the next sections. For ease of notations, we first derive our main convergence results assuming a uniform γ as defined here (Section 4), and then provide results with variable γ's in Corollary 8.

3.1 Federated Averaging (FedAvg)

In Federated Averaging (FedAvg) (McMahan et al., 2017), the local surrogate of the global objective function at device k is $F_k(\cdot)$, and the local solver is stochastic gradient descent (SGD), with the same learning rate and number of local epochs used on each device. At each round, a subset $K \ll N$ of the total devices are selected and run SGD locally for E number of epochs, and then the resulting model updates are averaged. The details of FedAvg are summarized in Algorithm 1.

McMahan et al. (2017) show empirically that it is crucial to tune the optimization hyperparameters for FedAvg properly. In particular, carefully tuning the number of local epochs is critical for FedAvg to converge. On one hand, performing more local epochs provides a better computation to communication tradeoff, and can potentially improve convergence. On the other hand, with dissimilar (heterogeneous) local objectives F_k, a larger number of local epochs may lead each device towards the optima of its local objective as opposed to the global objective, potentially hurting convergence or even resulting in divergence.

Algorithm 1 Federated Averaging (`FedAvg`)

Input: K, T, η, E, w^0, N, p_k, $k = 1, \cdots, N$
for $t = 0, \cdots, T-1$ **do**
  Server selects a subset S_t of K devices at random (each device k is chosen with probability p_k)
  Server sends w^t to all chosen devices
  Each device $k \in S_t$ updates w^t for E epochs of SGD on F_k with step-size η to obtain w_k^{t+1}
  Each device $k \in S_t$ sends w_k^{t+1} back to the server
  Server aggregates the w's as $w^{t+1} = \frac{1}{K}\sum_{k \in S_t} w_k^{t+1}$
end for

However, in practice, it is often not possible for the server to mandate a fixed number of local epochs E for all devices, as the amount of work performed on each device is often limited by underlying systems constraints. Alternatively, a more natural approach would be to adhere to the underlying systems constraints on each device, and have the server adapt to local updates corresponding to different values of E on each device. We formalize this strategy in `FedProx`, introduced below.

3.2 Proposed Framework: `FedProx`

Our proposed framework, `FedProx` (Algorithm 2), is similar to `FedAvg` in that a subset of devices are selected at each round, local updates are performed, and these updates are then averaged to form a global update. However, `FedProx` makes the following simple yet critical modifications, which result in significant empirical improvements and also allow us to provide convergence guarantees for the method.

Tolerating partial work. As previously discussed, different devices in federated networks often have different resource constraints in terms of the computing hardware, network connections, and battery levels. Therefore, it is unrealistic to force each device to perform the uniform amount of work (i.e., running the same number of local epochs, E), as in `FedAvg`. It is common that in `FedAvg`, some of the more resource-constrained devices drop from the training round after they are selected. `FedProx` generalizes `FedAvg` by allowing for variable amounts of work to be performed locally across devices based on their systems conditions, and aggregates the partial solutions sent from the stragglers. In other words, instead of assuming a uniform γ for all devices throughout the training process, `FedProx` implicitly accommodates variable γ's for different devices and at different iterations. We formally define γ_k^t-inexactness for device k at iteration t below, which is a natural extension from Definition 1.

Definition 2 (γ_k^t-inexact solution)**.** For a function $h_k(w; w_t) = F_k(w) + \frac{\mu}{2}\|w - w_t\|^2$, and $\gamma \in [0, 1]$, we say w^* is a γ_k^t-inexact solution of $\min_w h_k(w; w_t)$ if $\|\nabla h_k(w^*; w_t)\| \leq \gamma_k^t \|\nabla h_k(w_t; w_t)\|$, where $\nabla h_k(w; w_t) = \nabla F_k(w) + \mu(w - w_t)$. Note that a smaller γ_k^t corresponds to higher accuracy.

Similarly, γ_k^t measures how much local computation is performed to solve the local subproblem on device k at the tth round. The variable number of local iterations can be viewed as a proxy of γ_k^t. Utilizing the more flexible γ_k^t-inexactness, we can readily extend the convergence results under Definition 1 (Theorem 4) to consider issues related to systems heterogeneity such as stragglers (see Corollary 8).

Proximal term. As mentioned in Section 3.1, while tolerating nonuniform amounts of work to be performed across devices can help alleviate negative impacts of systems heterogeneity, too many local updates may still (potentially) cause the methods to diverge due to the underlying heterogeneous data. We propose to add a proximal term to the local subproblem to effectively limit the impact of variable local updates. In particular, instead of just minimizing the local function $F_k(\cdot)$, device k uses its local solver of choice to approximately minimize the following surrogate objective h_k:

$$\min_w h_k(w;\ w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2 . \qquad (2)$$

The proximal term is beneficial in two aspects. (1) It addresses the issue of statistical heterogeneity by restricting the local updates to be closer to the initial (global) model without any need to manually set the number of local epochs. (2) It allows for safely incorporating variable amounts of local work resulting from systems heterogeneity. We summarize the steps of `FedProx` in Algorithm 2.

Algorithm 2 `FedProx` (Proposed Framework)

Input: K, T, μ, γ, w^0, N, p_k, $k = 1, \cdots, N$
for $t = 0, \cdots, T-1$ **do**
  Server selects a subset S_t of K devices at random (each device k is chosen with probability p_k)
  Server sends w^t to all chosen devices
  Each chosen device $k \in S_t$ finds a w_k^{t+1} which is a γ_k^t-inexact minimizer of: $w_k^{t+1} \approx \arg\min_w h_k(w;\ w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2$
  Each device $k \in S_t$ sends w_k^{t+1} back to the server
  Server aggregates the w's as $w^{t+1} = \frac{1}{K}\sum_{k \in S_t} w_k^{t+1}$
end for

We note that proximal terms such as the one above are a popular tool utilized throughout the optimization literature; for completeness, we provide a more detailed discussion on this in Appendix B. An important distinction of the proposed usage is that we suggest, explore, and analyze such a term for the purpose of tackling heterogeneity in federated networks. Our analysis (Section 4) is also unique in considering solving such an objective in a distributed setting

with: (1) non-IID partitioned data, (2) the use of any local solver, (3) variable inexact updates across devices, and (4) a subset of devices being active at each round. These assumptions are critical to providing a characterization of such a framework in realistic federated scenarios.

In our experiments (Section 5), we demonstrate that tolerating partial work is beneficial in the presence of systems heterogeneity and our modified local subproblem in `FedProx` results in more robust and stable convergence compared to vanilla `FedAvg` for heterogeneous datasets. In Section 4, we also see that the usage of the proximal term makes `FedProx` more amenable to theoretical analysis (i.e., the local objective may be more well-behaved). In particular, if μ is chosen accordingly, the Hessian of h_k may be positive semi-definite. Hence, when F_k is non-convex, h_k will be convex, and when F_k is convex, it becomes μ-strongly convex.

`FedProx` bears some resemblance to `FedAvg`, which helps reason about the behavior of the widely-used `FedAvg` method, and allows for easy integration of `FedProx` into existing packages/systems, such as TensorFlow Federated and LEAF (TFF; Caldas et al., 2018). In particular, we note that `FedAvg` is a special case of `FedProx` with (1) $\mu = 0$, (2) the local solver specifically chosen to be SGD, and (3) a constant γ (corresponding to the number of local epochs) across devices and updating rounds (i.e., no notion of systems heterogeneity). `FedProx` is significantly more general in this regard, as it allows for partial work to be performed across devices and any local (possibly non-iterative) solver to be used on each device.

4 FedProx: Convergence Analysis

`FedAvg` and `FedProx` are stochastic algorithms by nature: in each round, only a fraction of the devices are sampled to perform the update, and the updates performed on each device may be inexact. It is well known that in order for stochastic methods to converge to a stationary point, a decreasing step-size is required. This is in contrast to non-stochastic methods, e.g., gradient descent, that can find a stationary point by employing a constant step-size. In order to analyze the convergence behavior of methods with constant step-size (as is usually implemented in practice), we need to quantify the degree of dissimilarity among the local objective functions. This could be achieved by assuming the data to be IID, i.e., homogeneous across devices. Unfortunately, in realistic federated networks, this assumption is impractical. Thus, we first propose a metric that specifically measures the dissimilarity among local functions (Section 4.1), and then analyze `FedProx` under this assumption while allowing for variable γ's (Section 4.2).

4.1 Local dissimilarity

Here we introduce a measure of dissimilarity between the devices in a federated network, which is sufficient to prove convergence. This can also be satisfied via a simpler and more restrictive bounded variance assumption of the gradients (Corollary 10), which we explore in our experiments in Section 5.

Definition 3 (B-local dissimilarity)**.** The local functions F_k are B-locally dissimilar at w if $\mathbb{E}_k\left[\|\nabla F_k(w)\|^2\right] \leq \|\nabla f(w)\|^2 B^2$. We further define $B(w) = \sqrt{\frac{\mathbb{E}_k[\|\nabla F_k(w)\|^2]}{\|\nabla f(w)\|^2}}$ for[2] $\|\nabla f(w)\| \neq 0$.

Here $\mathbb{E}_k[\cdot]$ denotes the expectation over devices with masses $p_k = n_k/n$ and $\sum_{k=1}^N p_k = 1$ (as in Equation 1). Definition 3 can be seen as a generalization of the IID assumption with bounded dissimilarity, while allowing for statistical heterogeneity. As a sanity check, when all the local functions are the same, we have $B(w) = 1$ for all w. However,in the federated setting, the data distributions are often heterogeneous and $B > 1$ due to sampling discrepancies even if the samples are assumed to be IID. Let us also consider the case where $F_k(\cdot)$'s are associated with empirical risk objectives. If the samples on all the devices are homogeneous, i.e., they are sampled in an IID fashion, then as $\min_k\ n_k \to \infty$, it follows that $B(w) \to 1$ for every w as all the local functions converge to the same expected risk function in the large sample limit. Thus, $B(w) \geq 1$ and the larger the value of $B(w)$, the larger is the dissimilarity among the local functions. Interestingly, similar assumptions (e.g., Schmidt & Roux, 2013; Vaswani et al., 2019; Yin et al., 2018) have been explored elsewhere but for differing purposes; we provide a discussion of these works in Appendix B.

Using Definition 3, we now state our formal dissimilarity assumption, which we use in our convergence analysis. This simply requires that the dissimilarity defined in Definition 3 is bounded. As discussed later, our convergence rate is a function of the statistical heterogeneity/device dissimilarity in the network.

Assumption 1 (Bounded dissimilarity)**.** *For some $\epsilon > 0$, there exists a B_ϵ such that for all the points $w \in \mathcal{S}_\epsilon^c = \{w \mid \|\nabla f(w)\|^2 > \epsilon\}$, $B(w) \leq B_\epsilon$.*

For most practical machine learning problems, there is no need to solve the problem to arbitrarily accurate stationary solutions, i.e., ϵ is typically not very small. Indeed, it is well-known that solving the problem beyond some threshold may even hurt generalization performance due to overfitting (Yao et al., 2007). Although in practical federated learning problems the samples are not IID, they are still sampled from

[2] As an exception we define $B(w) = 1$ when $\mathbb{E}_k\left[\|\nabla F_k(w)\|^2\right] = \|\nabla f(w)\|^2$, i.e. w is a stationary solution that all the local functions F_k agree on.

distributions that are not entirely unrelated (if this were the case, e.g., fitting a single global model w across devices would be ill-advised). Thus, it is reasonable to assume that the dissimilarity between local functions remains bounded throughout the training process. We also measure the dissimilarity metric empirically on real and synthetic datasets in Section 5.3.3 and show that this metric captures real-world statistical heterogeneity and thus implies practical performance (the smaller the dissimilarity, the better the convergence).

4.2 `FedProx` Analysis

Using the bounded dissimilarity assumption (Assumption 1), we now analyze the amount of expected decrease in the objective when one step of `FedProx` is performed. Our convergence rate (Theorem 6) can be directly derived from the results of the expected decrease per updating round. We assume the same γ_k^t for any k, t in the following analyses.

Theorem 4 (Non-convex `FedProx` convergence: B-local dissimilarity)**.** *Let Assumption 1 hold. Assume the functions F_k are non-convex, L-Lipschitz smooth, and there exists $L_- > 0$, such that $\nabla^2 F_k \succeq -L_- \mathbf{I}$, with $\bar{\mu} := \mu - L_- > 0$. Suppose that w^t is not a stationary solution and the local functions F_k are B-dissimilar, i.e. $B(w^t) \leq B$. If μ, K, and γ in Algorithm 2 are chosen such that*

$$\rho = \left(\frac{1}{\mu} - \frac{\gamma B}{\mu} - \frac{B(1+\gamma)\sqrt{2}}{\bar{\mu}\sqrt{K}} - \frac{LB(1+\gamma)}{\bar{\mu}\mu} - \frac{L(1+\gamma)^2 B^2}{2\bar{\mu}^2} - \frac{LB^2(1+\gamma)^2}{\bar{\mu}^2 K}\left(2\sqrt{2K}+2\right)\right) > 0,$$

then at iteration t of Algorithm 2, we have the following expected decrease in the global objective:

$$\mathbb{E}_{S_t}\left[f(w^{t+1})\right] \leq f(w^t) - \rho \|\nabla f(w^t)\|^2,$$

where S_t is the set of K devices chosen at iteration t.

We direct the reader to Appendix A.1 for a detailed proof. The key steps include applying our notion of γ-inexactness (Definition 1) for each subproblem and using the bounded dissimilarity assumption, while allowing for only K devices to be active at each round. This last step in particular introduces $\mathbb{E}_{S_t}$, an expectation with respect to the choice of devices, S_t, in round t. We note that in our theory, we require $\bar{\mu} > 0$, which is a sufficient but not necessary condition for `FedProx` to converge. Hence, it is possible that some other μ (not necessarily satisfying $\bar{\mu} > 0$) can also make `FedProx` converge, as explored in our experiments (Section 5).

Theorem 4 uses the dissimilarity in Definition 3 to identify sufficient decrease of the objective value at each iteration for `FedProx`. In Appendix A.2, we provide a corollary characterizing the performance with a more common (though slightly more restrictive) bounded variance assumption. This assumption is commonly employed, e.g., when analyzing methods such as SGD. We next provide sufficient (but not necessary) conditions that ensure $\rho > 0$ in Theorem 4 such that sufficient decrease is attainable after each round.

Remark 5. *For ρ in Theorem 4 to be positive, we need $\gamma B < 1$ and $\frac{B}{\sqrt{K}} < 1$. These conditions help to quantify the trade-off between dissimilarity (B) and the algorithm parameters (γ, K).*

Finally, we can use the above sufficient decrease to the characterize the rate of convergence to the set of approximate stationary solutions $\mathcal{S}_s = \{w \mid \mathbb{E}\left[\|\nabla f(w)\|^2\right] \leq \epsilon\}$ under the bounded dissimilarity assumption, Assumption 1. Note that these results hold for general non-convex $F_k(\cdot)$.

Theorem 6 (Convergence rate: `FedProx`)**.** *Given some $\epsilon > 0$, assume that for $B \geq B_\epsilon$, μ, γ, and K the assumptions of Theorem 4 hold at each iteration of `FedProx`. Moreover, $f(w^0) - f^* = \Delta$. Then, after $T = O(\frac{\Delta}{\rho\epsilon})$ iterations of `FedProx`, we have $\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\|\nabla f(w^t)\|^2\right] \leq \epsilon$.*

While the results thus far hold for non-convex $F_k(\cdot)$, we can also characterize the convergence for the special case of convex loss functions with exact minimization in terms of local objectives (Corollary 7). A proof is provided in Appendix A.3.

Corollary 7 (Convergence: Convex case)**.** *Let the assertions of Theorem 4 hold. In addition, let $F_k(\cdot)$'s be convex and $\gamma_k^t = 0$ for any k, t, i.e., all the local problems are solved exactly, if $1 \ll B \leq 0.5\sqrt{K}$, then we can choose $\mu \approx 6LB^2$ from which it follows that $\rho \approx \frac{1}{24LB^2}$.*

The previous analyses assume no systems heterogeneity and use the same γ for all devices and at all iterations. They can be easily extended to allow for γ to vary by device and by iteration (as in Definition 2), which corresponds to allowing devices to perform variable amounts of work locally determined by the local systems conditions. We provide the convergence rate accounting for variable γ's below.

Corollary 8 (Convergence: Variable γ's)**.** *Assume the functions F_k are non-convex, L-Lipschitz smooth, and there exists $L_- > 0$, such that $\nabla^2 F_k \succeq -L_- \mathbf{I}$, with $\bar{\mu} := \mu - L_- > 0$. Suppose that w^t is not a stationary solution and the local functions F_k are B-dissimilar, i.e. $B(w^t) \leq B$. If μ, K, and γ_k^t in Algorithm 2 are chosen such that*

$$\rho^t = \left(\frac{1}{\mu} - \frac{\gamma^t B}{\mu} - \frac{B(1+\gamma^t)\sqrt{2}}{\bar{\mu}\sqrt{K}} - \frac{LB(1+\gamma^t)}{\bar{\mu}\mu} - \frac{L(1+\gamma^t)^2 B^2}{2\bar{\mu}^2} - \frac{LB^2(1+\gamma^t)^2}{\bar{\mu}^2 K}\left(2\sqrt{2K}+2\right)\right) > 0,$$

then at iteration t of Algorithm 2, we have the following

expected decrease in the global objective:

$$\mathbb{E}_{S_t}\left[f(w^{t+1})\right] \leq f(w^t) - \rho^t \|\nabla f(w^t)\|^2,$$

where S_t is the set of K devices chosen at iteration t and $\gamma_t = \max_{k \in S_t} \gamma_k^t$.

The proof can be easily extended from the proof for Theorem 4 , noting the fact that $\mathbb{E}_k[(1+\gamma_k^t)\|\nabla F_k(w^t)\|] \leq (1+\max_{k\in S_t}\ \gamma_k^t)\mathbb{E}_k[\|\nabla F_k(w^t)\|]$.

Note that small ϵ in Assumption 1 translates to larger B_ϵ. Corollary 7 suggests that, in order to solve the problem with increasingly higher accuracies using `FedProx`, one needs to increase μ appropriately. We empirically verify that $\mu > 0$ leads to more stable convergence in Section 5.3. Moreover, in Corollary 7, if we plug in the upper bound for B_ϵ, under a bounded variance assumption (Corollary 10), the number of required steps to achieve accuracy ϵ is $O(\frac{L\Delta}{\epsilon} + \frac{L\Delta\sigma^2}{\epsilon^2})$. Our analysis helps to characterize the performance of `FedProx` and similar methods when local functions are dissimilar.

Remark 9 (Comparison with SGD)**.** *We note that `FedProx` achieves the same asymptotic convergence guarantee as SGD: Under the bounded variance assumption, for small ϵ, if we replace B_ϵ with its upper-bound in Corollary 10 and choose μ large enough, the iteration complexity of `FedProx` when the subproblems are solved exactly and $F_k(\cdot)$'s are convex is $O(\frac{L\Delta}{\epsilon} + \frac{L\Delta\sigma^2}{\epsilon^2})$, the same as SGD (Ghadimi & Lan, 2013).*

To help provide context for the rate in Theorem 6, we compare it with SGD in the convex case in Remark 9. In general, our analysis of `FedProx` does not provide better convergence rates than classical distributed SGD (without local updating)—even though `FedProx` possibly performs more work locally at each communication round. In fact, when data are generated in a non-identically distributed fashion, it is possible for local updating schemes such as `FedProx` to perform worse than distributed SGD. Therefore, our theoretical results do not necessarily demonstrate the superiority of `FedProx` over distributed SGD; rather, they provide sufficient (but not necessary) conditions for `FedProx` to converge. Our analysis is the first we are aware of to analyze any federated (i.e., with local-updating schemes and low device participation) optimization method in heterogeneous settings.

5 Experiments

We now present empirical results for the generalized `FedProx` framework. In Section 5.2, we demonstrate the improved performance of `FedProx` tolerating partial solutions in the face of systems heterogeneity. In Section 5.3, we show the effectiveness of `FedProx` in the presence of statistical heterogeneity (with and without systems heterogeneity). We also study the effects of statistical heterogeneity on convergence (Section 5.3.1) and show how empirical convergence is related to our theoretical bounded dissimilarity assumption (Assumption 1) (Section 5.3.3). We provide thorough details of the experimental setup in Section 5.1 and Appendix C, and provide an anonymized version of our code for easy reproducibility[3].

5.1 Experimental Details

We evaluate `FedProx` on diverse tasks, models, and real-world federated datasets. We simulate systems heterogeneity by assigning different amounts of local work to different devices. In order to better characterize statistical heterogeneity and study its effect on convergence, we also evaluate on a set of synthetic data, which allows for more precise manipulation of statistical heterogeneity.

Synthetic data. To generate synthetic data, we follow a similar setup to that in (Shamir et al., 2014), additionally imposing heterogeneity among devices. In particular, for each device k, we generate samples (X_k, Y_k) according to the model $y = \textit{argmax}(\text{softmax}(Wx+b))$, $x \in \mathbb{R}^{60}, W \in \mathbb{R}^{10\times 60}, b \in \mathbb{R}^{10}$. We model $W_k \sim \mathcal{N}(u_k, 1)$, $b_k \sim \mathcal{N}(u_k, 1)$, $u_k \sim \mathcal{N}(0, \alpha)$; $x_k \sim \mathcal{N}(v_k, \Sigma)$, where the covariance matrix Σ is diagonal with $\Sigma_{j,j} = j^{-1.2}$. Each element in the mean vector v_k is drawn from $\mathcal{N}(B_k, 1), B_k \sim N(0, \beta)$. Therefore, α controls how much local models differ from each other and β controls how much the local data at each device differs from that of other devices. We vary α, β to generate three heterogeneous distributed datasets, denoted Synthetic (α, β), as shown in Figure 2. We also generate one IID dataset by setting the same W, b on all devices and setting X_k to follow the same distribution. Our goal is to learn a global W and b. Full details are given in Appendix C.1.

Real data. We also explore four real datasets; statistics are summarized in Table 1. These datasets are curated from prior work in federated learning as well as recent federated learning benchmarks (Caldas et al., 2018; McMahan et al., 2017). We study a convex classification problem with MNIST (LeCun et al., 1998) using multinomial logistic regression. To impose statistical heterogeneity, we distribute the data among 1,000 devices such that each device has samples of only two digits and the number of samples per device follows a power law. We then study a more complex 62-class Federated Extended MNIST (Caldas et al., 2018; Cohen et al., 2017) (FEMNIST) dataset using the same model. For the non-convex setting, we consider a text sentiment analysis task on tweets from Sentiment140 (Go et al., 2009) (Sent140) with an LSTM classifier, where each twitter account corresponds to a device. We also investigate the task of next-character prediction on the dataset of *The*

[3] bit.ly/FedProx

Complete Works of William Shakespeare (McMahan et al., 2017) (Shakespeare). Each speaking role in the plays is associated with a different device. Details of datasets, models, and workloads are provided in Appendix C.1.

Table 1. Statistics of four real federated datasets.

Dataset	Devices	Samples	Samples/device	
			mean	stdev
MNIST	1,000	69,035	69	106
FEMNIST	200	18,345	92	159
Shakespeare	143	517,106	3,616	6,808
Sent140	772	40,783	53	32

Implementation. We implement `FedAvg` (Algorithm 1) and `FedProx` (Algorithm 2) in Tensorflow (Abadi et al., 2016). In order to draw a fair comparison with `FedAvg`, we employ SGD as a local solver for `FedProx`, and adopt a slightly different device sampling scheme than that in Algorithms 1 and 2: sampling devices uniformly and then averaging the updates with weights proportional to the number of local data points (as originally proposed in (McMahan et al., 2017)). While this sampling scheme is not supported by our analysis, we observe similar relative behavior of `FedProx` vs. `FedAvg` whether or not it is employed. Interestingly, we also observe that the sampling scheme proposed herein in fact results in more stable performance for both methods (see Appendix C.3.3, Figure 11). This suggests an additional benefit of the proposed framework. Full details are provided in Appendix C.2.

Hyper-parameters & evaluation metrics. We tune the learning rate on `FedAvg` and set the number of selected devices to be 10 for all experiments on all datasets. For each comparison, we fix the randomly selected devices and mini-batch orders across all runs. We report all metrics based on the global objective $f(w)$. Note that we assume that each communication round corresponds to a specific aggregation time stamp (measured in real-world global wall-clock time)—we therefore report results in terms of rounds rather than FLOPs or wall-clock time. See details of the hyper-parameters in Appendix C.2.

5.2 Effects of Allowing for Partial Work

In order to measure the effects of allowing for partial solutions to be sent to handle systems heterogeneity with `FedProx`, we simulate federated settings with varying system heterogeneity, as described below.

Systems heterogeneity simulations. We assume that there is a real-world global clock cycle to aggregate model updates, and each participating device determines the amount of local work as a function of this clock cycle and its systems constraints. This specified amount of local computation corresponds to some implicit value γ_k^t for device k at the tth iteration. In our simulations, we fix a global number of epochs E, and force some devices to perform fewer updates than E epochs given their current systems constraints. In particular, for varying heterogeneous settings, at each round, we assign x number of epochs (chosen uniformly at random between [1, E]) to 0%, 50%, and 90% of the selected devices, respectively. Settings where 0% devices perform fewer than E epochs of work correspond to the environments without systems heterogeneity, while 90% of the devices sending their partial solutions corresponds to highly heterogeneous environments. `FedAvg` will simply drop these 0%, 50%, and 90% stragglers upon reaching the global clock cycle, and `FedProx` will incorporate the partial updates from these devices.

In Figure 1, we set E to be 20 and study the effects of aggregating partial work from the otherwise dropped devices. The synthetic dataset here is taken from Synthetic (1,1) in Figure 2. We see that on all the datasets, systems heterogeneity has negative effects on convergence, and larger heterogeneity results in worse convergence (`FedAvg`). Compared with dropping the more constrained devices (`FedAvg`), tolerating variable amounts of work (`FedProx`, $\mu = 0$) is beneficial and leads to more stable and faster convergence.

We also investigate two less heterogeneous settings. First, we limit the capability of all the devices by setting E to be 1 (i.e., all the devices can run at most one local epoch at each iteration), and impose systems heterogeneity in a similar way. We show training loss in Figure 9 and testing accuracy in Figure 10 in the appendix. We see that allowing for partial work can still improve convergence compared with `FedAvg`. Second, we explore a setting without any statistical heterogeneity using an identically distributed synthetic dataset (Synthetic IID). In this IID setting, as shown in Figure 5 in Appendix C.3.2, `FedAvg` is rather robust under device failure, and tolerating variable amounts of local work may not cause major improvement. This serves as additional motivation to rigorously study the effect of heterogeneity on new methods designed for federated learning—as simply relying on IID data (a setting unlikely to occur in practice) may not tell a complete story.

5.3 Effects of the Proximal Term

In order to better understand how the proximal term can be beneficial in heterogeneous settings, we first show that convergence can become worse as statistical heterogeneity increases.

5.3.1 Effects of Statistical Heterogeneity

In Figure 2 (the first row), we study how statistical heterogeneity affects convergence using four synthetic datasets

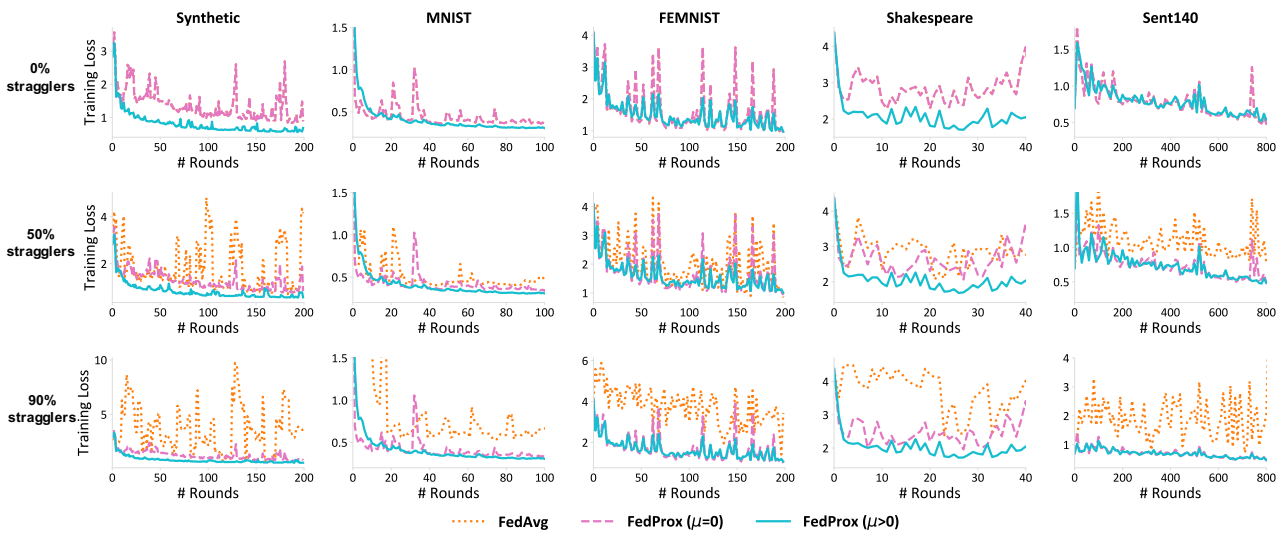

Figure 1. `FedProx` results in significant convergence improvements relative to `FedAvg` in heterogeneous networks. We simulate different levels of systems heterogeneity by forcing 0%, 50%, and 90% devices to be the stragglers (dropped by `FedAvg`). (1) Comparing `FedAvg` and `FedProx` ($\mu = 0$), we see that allowing for variable amounts of work to be performed can help convergence in the presence of systems heterogeneity. (2) Comparing `FedProx` ($\mu = 0$) with `FedProx` ($\mu > 0$), we show the benefits of our added proximal term. `FedProx` with $\mu > 0$ leads to more stable convergence and enables otherwise divergent methods to converge, both in the presence of systems heterogeneity (50% and 90% stragglers) and without systems heterogeneity (0% stragglers). Note that `FedProx` with $\mu = 0$ and without systems heterogeneity (no stragglers) corresponds to `FedAvg`.

without the presence of systems heterogeneity (fixing E to be 20). From left to right, as data become more heterogeneous, convergence becomes worse for `FedProx` with $\mu = 0$ (i.e., `FedAvg`). Though it may slow convergence for IID data, we see that setting $\mu > 0$ is particularly useful in heterogeneous settings. This indicates that the modified subproblem introduced in `FedProx` can benefit practical federated settings with varying statistical heterogeneity. For perfect IID data, some heuristics such as decreasing μ if the loss continues to decrease may help avoid the deceleration of convergence (see Figure 12 in Appendix C.3.4). In the sections to follow, we see similar results in our non-synthetic experiments.

5.3.2 *Effects of* $\mu > 0$

The key parameters of `FedProx` that affect performance are the amount of local work (as parameterized by the number of local epochs, E), and the proximal term scaled by μ. Intuitively, large E may cause local models to drift too far away from the initial starting point, thus leading to potential divergence (McMahan et al., 2017). Therefore, to handle the divergence or instability of `FedAvg` with non-IID data, it is helpful to tune E carefully. However, E is constrained by the underlying system's environments on the devices, and it is difficult to determine an appropriate uniform E for all devices. Alternatively, it is beneficial to allow for device-specific E's (variable γ's) and tune a best μ (a parameter that can be viewed as a re-parameterization of E) to prevent divergence and improve the stability of methods. A proper μ can restrict the trajectory of the iterates by constraining the iterates to be closer to that of the global model, thus incorporating variable amounts of updates and guaranteeing convergence (Theorem 6).

We show the effects of the proximal term in `FedProx` ($\mu > 0$) in Figure 1. For each experiment, we compare the results between `FedProx` with $\mu = 0$ and `FedProx` with a best μ (see the next paragraph for discussions on how to select μ). For all datasets, we observe that the appropriate μ can increase the stability for unstable methods and can force divergent methods to converge. This holds both when there is systems heterogeneity (50% and 90% stragglers) and there is no systems heterogeneity (0% stragglers). $\mu > 0$ also increases the accuracy in most cases (see Figure 6 and Figure 7 in Appendix C.3.2). In particular, `FedProx` improves absolute testing accuracy relative to `FedAvg` by 18.8% on average in highly heterogeneous environments (90% stragglers) (see Figure 7).

Choosing μ. One natural question is to determine how to set the penalty constant μ in the proximal term. A large μ may potentially slow the convergence by forcing the updates to be close to the starting point, while a small μ may not make any difference. In all experiments, we tune the best μ from the limited candidate set $\{0.001, 0.01, 0.1, 1\}$. For the five federated datasets in Figure 1, the best μ values are

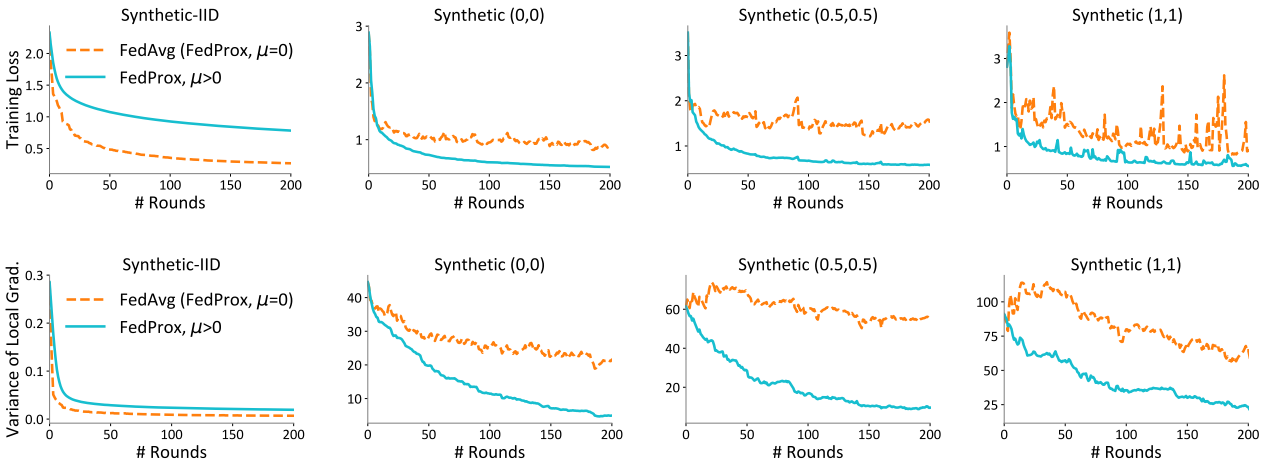

Figure 2. Effect of data heterogeneity on convergence. We remove the effects of systems heterogeneity by forcing each device to run the same amount of epochs. In this setting, `FedProx` with $\mu = 0$ reduces to `FedAvg`. (1) Top row: We show training loss (see results on testing accuracy in Appendix C.3, Figure 6) on four synthetic datasets whose statistical heterogeneity increases from left to right. Note that the method with $\mu = 0$ corresponds to `FedAvg`. Increasing heterogeneity leads to worse convergence, but setting $\mu > 0$ can help to combat this. (2) Bottom row: We show the corresponding dissimilarity measurement (variance of gradients) of the four synthetic datasets. This metric captures statistical heterogeneity and is consistent with training loss — smaller dissimilarity indicates better convergence.

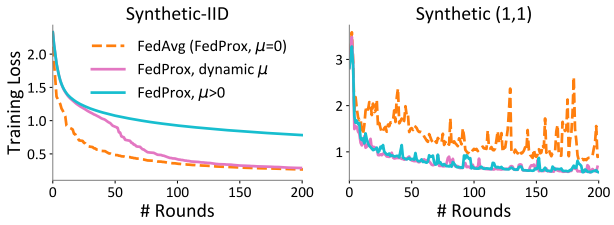

Figure 3. Effectiveness of setting μ adaptively based on the current model performance. We increase μ by 0.1 whenever the loss increases and decreases it by 0.1 whenever the loss decreases for 5 consecutive rounds. We initialize μ to 1 for Synthetic IID (in order to be adversarial to our methods), and initialize μ to 0 for Synthetic (1,1). This simple heuristic works well empirically.

1, 1, 1, 0.001, and 0.01, respectively. While automatically tuning μ is difficult to instantiate directly from the theory, in practice, we note that μ can be adaptively chosen based on the current performance of the model. For example, one simple heuristic is to increase μ when seeing the loss increasing and decreasing μ when seeing the loss decreasing. In Figure 3, we demonstrate the effectiveness of this heuristic using two synthetic datasets. Note that we start from initial μ values that are adversarial to our methods. We provide full results showing the competitive performance of this approach in Appendix C.3.4. Future work includes developing methods to automatically tune this parameter for heterogeneous datasets, based, e.g., on the theoretical groundwork provided here.

5.3.3 *Dissimilarity Measurement and Divergence*

Finally, in Figure 2 (the bottom row), we demonstrate that our B-local dissimilarity measurement in Definition 3 captures the heterogeneity of datasets and is therefore an appropriate proxy of performance. In particular, we track the variance of gradients on each device, $E_k[\|\nabla F_k(w) - \nabla f(w)\|^2]$, which is lower bounded by B_ϵ (see Bounded Variance Equivalence Corollary 10). Empirically, we observe that increasing μ leads to smaller dissimilarity among local functions F_k, and that the dissimilarity metric is consistent with the training loss. Therefore, smaller dissimilarity indicates better convergence, which can be enforced by setting μ appropriately. We also show the dissimilarity metric on real federated data in Appendix C.3.2.

6 Conclusion

In this work, we have proposed `FedProx`, an optimization framework that tackles the systems and statistical heterogeneity inherent in federated networks. `FedProx` allows for variable amounts of work to be performed locally across devices, and relies on a proximal term to help stabilize the method. We provide the first convergence results of `FedProx` in realistic federated settings under a device dissimilarity assumption, while also accounting for practical issues such as stragglers. Our empirical evaluation across a suite of federated datasets has validated our theoretical analysis and demonstrated that the `FedProx` framework can significantly improve the convergence behavior of federated learning in realistic heterogeneous networks.

REFERENCES

Tensorflow federated: Machine learning on decentralized data. URL https://www.tensorflow.org/federated.

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M. K., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. Tensorflow: A system for large-scale machine learning. In *Operating Systems Design and Implementation*, pp. 265–283, 2016.

Allen-Zhu, Z. How to make the gradients small stochastically: Even faster convex and nonconvex sgd. In *Advances in Neural Information Processing Systems*, pp. 1157–1167, 2018.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecny, J., Mazzocchi, S., McMahan, H. B., Overveldt, T. V., Petrou, D., Ramage, D., and Roselander, J. Towards federated learning at scale: system design. In *Conference on Systems and Machine Learning*, 2019.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010.

Caldas, S., Wu, P., Li, T., Konečnỳ, J., McMahan, H. B., Smith, V., and Talwalkar, A. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

Cohen, G., Afshar, S., Tapson, J., and van Schaik, A. Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Le, Q. V., Mao, M., Ranzato, M., Senior, A., Tucker, P., Yang, K., and Ng, A. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pp. 1223–1231, 2012.

Dekel, O., Gilad-Bachrach, R., Shamir, O., and Xiao, L. Optimal Distributed Online Prediction Using Mini-Batches. *Journal of Machine Learning Research*, 13:165–202, 2012.

Ghadimi, S. and Lan, G. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Go, A., Bhayani, R., and Huang, L. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 2009.

Hao, Y., Rong, J., and Sen, Y. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. In *International Conference on Machine Learning*, pp. 7184–7193, 2019.

Huang, L., Yin, Y., Fu, Z., Zhang, S., Deng, H., and Liu, D. Loadaboost: Loss-based adaboost federated machine learning on medical data. *arXiv preprint arXiv:1811.12629*, 2018.

Jeong, E., Oh, S., Kim, H., Park, J., Bennis, M., and Kim, S.-L. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

Jiang, P. and Agrawal, G. A linear speedup analysis of distributed deep learning with sparse and quantized communication. In *Advances in Neural Information Processing Systems*, pp. 2525–2536, 2018.

Kaczmarz, S. Approximate solution of systems of linear equations. *International Journal of Control*, 57(6):1269–1271, 1993.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, M., Andersen, D. G., Smola, A. J., and Yu, K. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, pp. 19–27, 2014a.

Li, M., Zhang, T., Chen, Y., and Smola, A. J. Efficient mini-batch training for stochastic optimization. In *Conference on Knowledge Discovery and Data Mining*, pp. 661–670, 2014b.

Li, T., Sahu, A., Talwalkar, A., and Smith, V. Federated learning: Challenges, methods, and future directions. *arXiv preprint arXiv:1908.07873*, 2019.

Lin, T., Stich, S. U., and Jaggi, M. Don't use large mini-batches, use local sgd. *arXiv preprint arXiv:1808.07217*, 2018.

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

Pennington, J., Socher, R., and Manning, C. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing*, pp. 1532–1543, 2014.

Reddi, S. J., Konečnỳ, J., Richtárik, P., Póczós, B., and Smola, A. Aide: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879*, 2016.

Richtárik, P. and Takáč, M. Distributed coordinate descent method for learning with big data. *Journal of Machine Learning Research*, 17(1):1–25, 2016.

Schmidt, M. and Roux, N. L. Fast convergence of stochastic gradient descent under a strong growth condition. *arXiv preprint arXiv:1308.6370*, 2013.

Shamir, O., Srebro, N., and Zhang, T. Communication-efficient distributed optimization using an approximate newton-type method. In *International Conference on Machine Learning*, pp. 1000–1008, 2014.

Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. S. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pp. 4424–4434, 2017.

Smith, V., Forte, S., Ma, C., Takac, M., Jordan, M. I., and Jaggi, M. Cocoa: A general framework for communication-efficient distributed optimization. *Journal of Machine Learning Research*, 18(230):1–47, 2018.

Stich, S. U. Local sgd converges fast and communicates little. In *International Conference on Learning Representations*, 2019.

Strohmer, T. and Vershynin, R. A randomized kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15(2):262, 2009.

Vaswani, S., Bach, F., and Schmidt, M. Fast and faster convergence of sgd for over-parameterized models (and an accelerated perceptron). In *International Conference on Artificial Intelligence and Statistics*, pp. 1195–1204, 2019.

Wang, J. and Joshi, G. Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms. *arXiv preprint arXiv:1808.07576*, 2018.

Wang, S., Tuor, T., Salonidis, T., Leung, K. K., Makaya, C., He, T., and Chan, K. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, 37(6): 1205–1221, 2019.

Woodworth, B. E., Wang, J., Smith, A., McMahan, B., and Srebro, N. Graph oracle models, lower bounds, and gaps for parallel stochastic optimization. In *Advances in Neural Information Processing Systems*, pp. 8496–8506, 2018.

Yao, Y., Rosasco, L., and Caponnetto, A. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

Yin, D., Pananjady, A., Lam, M., Papailiopoulos, D., Ramchandran, K., and Bartlett, P. Gradient diversity: a key ingredient for scalable distributed learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 1998–2007, 2018.

Yu, H., Yang, S., and Zhu, S. Parallel restarted sgd for non-convex optimization with faster convergence and less communication. In *AAAI Conference on Artificial Intelligence*, 2018.

Zhang, S., Choromanska, A. E., and LeCun, Y. Deep learning with elastic averaging sgd. In *Advances in Neural Information Processing Systems*, pp. 685–693, 2015.

Zhang, Y., Duchi, J. C., and Wainwright, M. J. Communication-efficient algorithms for statistical optimization. *Journal of Machine Learning Research*, 14: 3321–3363, 2013.

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

Zhou, F. and Cong, G. On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization. In *International Joint Conference on Artificial Intelligence*, pp. 3219–3227, 2018.

A COMPLETE PROOFS

A.1 Proof of Theorem 4

Proof. Using our notion of γ-inexactness for each local solver (Definition 1), we can define e_k^{t+1} such that:

$$\begin{aligned} &\nabla F_k(w_k^{t+1}) + \mu(w_k^{t+1} - w^t) - e_k^{t+1} = 0, \\ &\|e_k^{t+1}\| \leq \gamma \|\nabla F_k(w^t)\|. \end{aligned} \tag{3}$$

Now let us define $\bar{w}^{t+1} = \mathbb{E}_k\left[w_k^{t+1}\right]$. Based on this definition, we know

$$\bar{w}^{t+1} - w^t = \frac{-1}{\mu}\mathbb{E}_k\left[\nabla F_k(w_k^{t+1})\right] + \frac{1}{\mu}\mathbb{E}_k\left[e_k^{t+1}\right]. \tag{4}$$

Let us define $\bar{\mu} = \mu - L_- > 0$ and $\hat{w}_k^{t+1} = \arg\min_w \; h_k(w; w^t)$. Then, due to the $\bar{\mu}$-strong convexity of h_k, we have

$$\|\hat{w}_k^{t+1} - w_k^{t+1}\| \leq \frac{\gamma}{\bar{\mu}}\|\nabla F_k(w^t)\|. \tag{5}$$

Note that once again, due to the $\bar{\mu}$-strong convexity of h_k, we know that $\|\hat{w}_k^{t+1} - w^t\| \leq \frac{1}{\bar{\mu}}\|\nabla F_k(w^t)\|$. Now we can use the triangle inequality to get

$$\|w_k^{t+1} - w^t\| \leq \frac{1+\gamma}{\bar{\mu}}\|\nabla F_k(w^t)\|. \tag{6}$$

Therefore,

$$\begin{aligned} &\|\bar{w}^{t+1} - w^t\| \leq \mathbb{E}_k\left[\|w_k^{t+1} - w^t\|\right] \leq \frac{1+\gamma}{\bar{\mu}}\mathbb{E}_k\left[\|\nabla F_k(w^t)\|\right] \\ &\leq \frac{1+\gamma}{\bar{\mu}}\sqrt{\mathbb{E}_k[\|\nabla F_k(w^t)\|^2]} \leq \frac{B(1+\gamma)}{\bar{\mu}}\|\nabla f(w^t)\|, \end{aligned} \tag{7}$$

where the last inequality is due to the bounded dissimilarity assumption.

Now let us define M_{t+1} such that $\bar{w}^{t+1} - w^t = \frac{-1}{\mu}\big(\nabla f(w^t) + M_{t+1}\big)$, i.e. $M_{t+1} = \mathbb{E}_k\left[\nabla F_k(w_k^{t+1}) - \nabla F_k(w^t) - e_k^{t+1}\right]$. We can bound $\|M_{t+1}\|$:

$$\|M_{t+1}\| \leq \mathbb{E}_k\left[L\|w_k^{t+1} - w_k^t\| + \|e_k^{t+1}\|\right] \leq \left(\frac{L(1+\gamma)}{\bar{\mu}} + \gamma\right) \times \mathbb{E}_k\left[\|\nabla F_k(w^t)\|\right] \tag{8}$$

$$\leq \left(\frac{L(1+\gamma)}{\bar{\mu}} + \gamma\right) B\|\nabla f(w^t)\|, \tag{9}$$

where the last inequality is also due to bounded dissimilarity assumption. Based on the L-Lipschitz smoothness of f and Taylor expansion, we have

$$\begin{aligned} &f(\bar{w}^{t+1}) \leq f(w^t) + \langle \nabla f(w^t), \bar{w}^{t+1} - w^t\rangle + \frac{L}{2}\|\bar{w}^{t+1} - w^t\|^2 \\ &\leq f(w^t) - \frac{1}{\mu}\|\nabla f(w^t)\|^2 - \frac{1}{\mu}\langle \nabla f(w^t), M_{t+1}\rangle + \frac{L(1+\gamma)^2 B^2}{2\bar{\mu}^2}\|\nabla f(w^t)\|^2 \\ &\leq f(w^t) - \left(\frac{1-\gamma B}{\mu} - \frac{LB(1+\gamma)}{\bar{\mu}\mu} - \frac{L(1+\gamma)^2 B^2}{2\bar{\mu}^2}\right) \times \|\nabla f(w^t)\|^2. \end{aligned} \tag{10}$$

From the above inequality it follows that if we set the penalty parameter μ large enough, we can get a decrease in the objective value of $f(\bar{w}^{t+1}) - f(w^t)$ which is proportional to $\|\nabla f(w^t)\|^2$. However, this is not the way that the algorithm works. In the algorithm, we only use K devices that are chosen randomly to approximate $\bar{w}^t$. So, in order to find the $\mathbb{E}\left[f(w^{t+1})\right]$, we use local Lipschitz continuity of the function f.

$$f(w^{t+1}) \leq f(\bar{w}^{t+1}) + L_0\|w^{t+1} - \bar{w}^{t+1}\|, \tag{11}$$

where L_0 is the local Lipschitz continuity constant of function f and we have

$$\begin{aligned} L_0 &\leq \|\nabla f(w^t)\| + L\max(\|\bar{w}^{t+1} - w^t\|, \|w^{t+1} - w^t\|) \\ &\leq \|\nabla f(w^t)\| + L(\|\bar{w}^{t+1} - w^t\| + \|w^{t+1} - w^t\|). \end{aligned} \tag{12}$$

Therefore, if we take expectation with respect to the choice of devices in round t we need to bound

$$\mathbb{E}_{S_t}\left[f(w^{t+1})\right] \leq f(\bar{w}^{t+1}) + Q_t, \tag{13}$$

where $Q_t = \mathbb{E}_{S_t}\left[L_0\|w^{t+1} - \bar{w}^{t+1}\|\right]$. Note that the expectation is taken over the random choice of devices to update.

$$\begin{aligned} Q_t &\leq \mathbb{E}_{S_t}\left[\left(\|\nabla f(w^t)\| + L(\|\bar{w}^{t+1} - w^t\| + \|w^{t+1} - w^t\|)\right) \times \|w^{t+1} - \bar{w}^{t+1}\|\right] \\ &\leq \left(\|\nabla f(w^t)\| + L\|\bar{w}^{t+1} - w^t\|\right)\mathbb{E}_{S_t}\left[\|w^{t+1} - \bar{w}^{t+1}\|\right] + L\mathbb{E}_{S_t}\left[\|w^{t+1} - w^t\| \cdot \|w^{t+1} - \bar{w}^{t+1}\|\right] \\ &\leq \left(\|\nabla f(w^t)\| + 2L\|\bar{w}^{t+1} - w^t\|\right)\mathbb{E}_{S_t}\left[\|w^{t+1} - \bar{w}^{t+1}\|\right] + L\mathbb{E}_{S_t}\left[\|w^{t+1} - \bar{w}^{t+1}\|^2\right] \end{aligned} \tag{14}$$

From (7), we have that $\|\bar{w}^{t+1} - w^t\| \leq \frac{B(1+\gamma)}{\bar{\mu}}\|\nabla f(w^t)\|$. Moreover,

$$\mathbb{E}_{S_t}\left[\|w^{t+1} - \bar{w}^{t+1}\|\right] \leq \sqrt{\mathbb{E}_{S_t}\left[\|w^{t+1} - \bar{w}^{t+1}\|^2\right]} \tag{15}$$

and

$$\begin{aligned} \mathbb{E}_{S_t}\left[\|w^{t+1} - \bar{w}^{t+1}\|^2\right] &\leq \frac{1}{K}\mathbb{E}_k\left[\|w_k^{t+1} - \bar{w}^{t+1}\|^2\right] \\ &\leq \frac{2}{K}\mathbb{E}_k\left[\|w_k^{t+1} - w^t\|^2\right], \quad (\text{as } \bar{w}^{t+1} = \mathbb{E}_k\left[w_k^{t+1}\right]) \\ &\leq \frac{2}{K}\frac{(1+\gamma)^2}{\bar{\mu}^2}\mathbb{E}_k\left[\|\nabla F_k(w^t)\|^2\right] \quad (\text{from (6)}) \\ &\leq \frac{2B^2}{K}\frac{(1+\gamma)^2}{\bar{\mu}^2}\|\nabla f(w^t)\|^2, \end{aligned} \tag{16}$$

where the first inequality is a result of K devices being chosen randomly to get w^t and the last inequality is due to bounded dissimilarity assumption. If we replace these bounds in (14) we get

$$Q_t \leq \left(\frac{B(1+\gamma)\sqrt{2}}{\bar{\mu}\sqrt{K}} + \frac{LB^2(1+\gamma)^2}{\bar{\mu}^2 K}\left(2\sqrt{2K} + 2\right)\right)\|\nabla f(w^t)\|^2 \tag{17}$$

Combining (10), (13), (11) and (17) and using the notation $\alpha = \frac{1}{\mu}$ we get

$$\begin{aligned} \mathbb{E}_{S_t}\left[f(w^{t+1})\right] \leq f(w^t) - \Bigg(&\frac{1}{\mu} - \frac{\gamma B}{\mu} - \frac{B(1+\gamma)\sqrt{2}}{\bar{\mu}\sqrt{K}} - \frac{LB(1+\gamma)}{\bar{\mu}\mu} \\ &- \frac{L(1+\gamma)^2B^2}{2\bar{\mu}^2} - \frac{LB^2(1+\gamma)^2}{\bar{\mu}^2 K}\left(2\sqrt{2K} + 2\right)\Bigg)\|\nabla f(w^t)\|^2. \end{aligned}$$

□

A.2 Proof for Bounded Variance

Corollary 10 (Bounded variance equivalence)**.** *Let Assumption 1 hold. Then, in the case of bounded variance, i.e.,* $\mathbb{E}_k\left[\|\nabla F_k(w) - \nabla f(w)\|^2\right] \leq \sigma^2$*, for any* $\epsilon > 0$ *it follows that* $B_\epsilon \leq \sqrt{1 + \frac{\sigma^2}{\epsilon}}$.

Proof. We have,

$$\begin{aligned} &E_k[\|\nabla F_k(w) - \nabla f(w)\|^2] = E_k[\|\nabla F_k(w)\|^2] - \|\nabla f(w)\|^2 \leq \sigma^2 \\ &\Rightarrow E_k[\|\nabla F_k(w)\|^2] \leq \sigma^2 + \|\nabla f(w)\|^2 \\ &\Rightarrow B_\epsilon = \sqrt{\frac{E_k[\|\nabla F_k(w)\|^2]}{\|\nabla f(w)\|^2}} \leq \sqrt{1 + \frac{\sigma^2}{\epsilon}}. \end{aligned}$$

With Corollary 10 in place, we can restate the main result in Theorem 4 in terms of the bounded variance assumption.

Theorem 11 (Non-convex `FedProx` convergence: Bounded variance)**.** *Let the assertions of Theorem 4 hold. In addition, let the iterate w^t be such that $\|\nabla f(w^t)\|^2 \geq \epsilon$, and let $\mathbb{E}_k\left[\|\nabla F_k(w) - \nabla f(w)\|^2\right] \leq \sigma^2$ hold instead of the dissimilarity condition. If μ, K and γ in Algorithm 2 are chosen such that*

$$\rho = \left(\frac{1}{\mu} - \left(\frac{\gamma}{\mu} + \frac{(1+\gamma)\sqrt{2}}{\bar{\mu}\sqrt{K}} + \frac{L(1+\gamma)}{\bar{\mu}\mu}\right)\sqrt{1+\frac{\sigma^2}{\epsilon}}\right) - \left(\frac{L(1+\gamma)^2}{2\bar{\mu}^2} + \frac{L(1+\gamma)^2}{\bar{\mu}^2 K}\left(2\sqrt{2K}+2\right)\right)\left(1+\frac{\sigma^2}{\epsilon}\right)\Bigg) > 0,$$

then at iteration t of Algorithm 2, we have the following expected decrease in the global objective:

$$\mathbb{E}_{S_t}\left[f(w^{t+1})\right] \leq f(w^t) - \rho\|\nabla f(w^t)\|^2,$$

where S_t is the set of K devices chosen at iteration t.

The proof of Theorem 11 follows from the proof of Theorem 4 by noting the relationship between the bounded variance assumption and the dissimilarity assumption as portrayed by Corollary 10.

A.3 Proof of Corollary 7

In the convex case, where $L_- = 0$ and $\bar{\mu} = \mu$, if $\gamma = 0$, i.e., all subproblems are solved accurately, we can get a decrease proportional to $\|\nabla f(w^t)\|^2$ if $B < \sqrt{K}$. In such a case if we assume $1 << B \leq 0.5\sqrt{K}$, then we can write

$$\mathbb{E}_{S_t}\left[f(w^{t+1})\right] \lessapprox f(w^t) - \frac{1}{2\mu}\|\nabla f(w^t)\|^2 + \frac{3LB^2}{2\mu^2}\|\nabla f(w^t)\|^2 . \tag{18}$$

In this case, if we choose $\mu \approx 6LB^2$ we get

$$\mathbb{E}_{S_t}\left[f(w^{t+1})\right] \lessapprox f(w^t) - \frac{1}{24LB^2}\|\nabla f(w^t)\|^2 . \tag{19}$$

Note that the expectation in (19) is a conditional expectation conditioned on the previous iterate. Taking expectation of both sides, and telescoping, we have that the number of iterations to at least generate one solution with squared norm of gradient less than ϵ is $O(\frac{LB^2\Delta}{\epsilon})$.

B CONNECTIONS TO OTHER SINGLE-MACHINE AND DISTRIBUTED METHODS

Two aspects of the proposed work—the proximal term in FedProx, and the bounded dissimilarity assumption used in our analysis—have been previously studied in the optimization literature, but with very different motivations. For completeness, we provide a discussion below on our relation to these prior works.

Proximal term. The proposed modified objective in FedProx shares a connection with elastic averaging SGD (EASGD) (Zhang et al., 2015), which was proposed as a way to train deep networks in the data center setting, and uses a similar proximal term in its objective. While the intuition is similar to EASGD (this term helps to prevent large deviations on each device/machine), EASGD employs a more complex moving average to update parameters, is limited to using SGD as a local solver, and has only been analyzed for simple quadratic problems. The proximal term we introduce has also been explored in previous optimization literature with different purposes, such as (Allen-Zhu, 2018), to speed up (mini-batch) SGD training on a single machine, and in (Li et al., 2014b) for efficient SGD training both in a single machine and distributed settings. However, the analysis in (Li et al., 2014b) is limited to a single machine setting with different assumptions (e.g., IID data and solving the subproblem exactly at each round). Finally, DANE (Shamir et al., 2014) and AIDE (Reddi et al., 2016), distributed methods designed for the data center setting, propose a similar proximal term in the local objective function, but also augment this with an additional gradient correction term. Both methods assume that all devices participate at each communication round, which is impractical in federated settings. Indeed, due to the inexact estimation of full gradients (i.e., $\nabla\phi(w^{(t-1)})$ in (Shamir et al., 2014, Eq (13))) with device subsampling schemes and the staleness of the gradient correction term (Shamir et al., 2014, Eq (13)), these methods are not directly applicable to our setting. Regardless of this, we explore a variant of such an approach in federated settings and see that the gradient direction term does not help in this scenario—performing uniformly worse than the proposed FedProx framework for heterogeneous datasets, despite the extra computation required (see Figure 4).

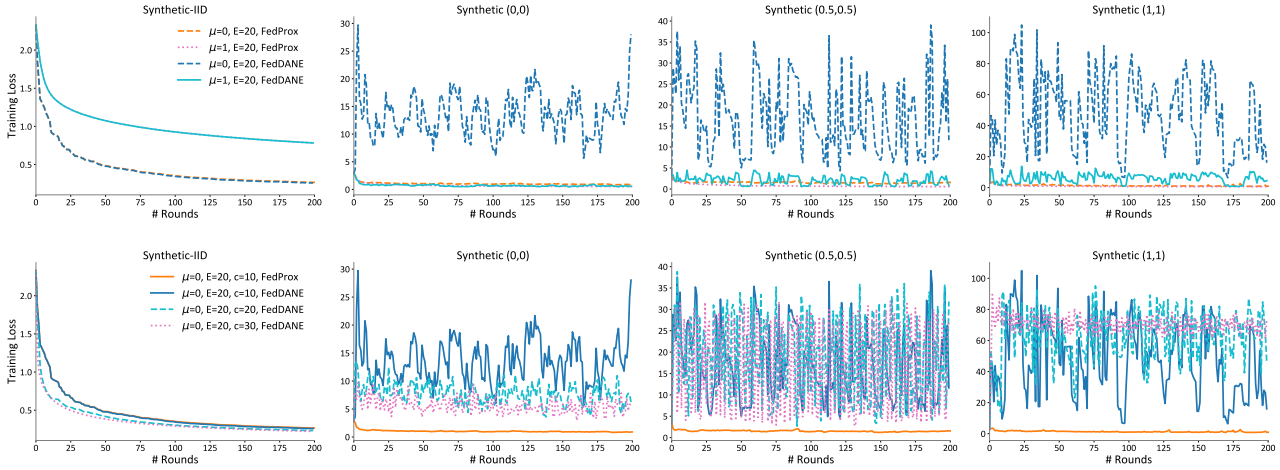

Figure 4. DANE and AIDE (Reddi et al., 2016; Shamir et al., 2014) are methods proposed in the data center setting that use a similar proximal term as FedProx as well as an additional gradient correction term. We modify DANE to apply to federated settings by allowing for local updating and low participation of devices. We show the convergence of this modified method, which we call FedDane, on synthetic datasets. In the top figures, we subsample 10 devices out of 30 on all datasets for both FedProx and FedDane. While FedDane performs similarly as FedProx on the IID data, it suffers from poor convergence on the non-IID datasets. In the bottom figures, we show the results of FedDane when we increase the number of selected devices in order to narrow the gap between our estimated full gradient and the real full gradient (in the gradient correction term). Note that communicating with all (or most of the) devices is already unrealistic in practical settings. We observe that although sampling more devices per round might help to some extent, FedDane is still unstable and tends to diverge. This serves as additional motivation for the specific subproblem we propose in FedProx.

Bounded dissimilarity assumption. The bounded dissimilarity assumption we discuss in Assumption 1 has appeared in different forms, for example in (Schmidt & Roux, 2013; Vaswani et al., 2019; Yin et al., 2018). In (Yin et al., 2018), the bounded similarity assumption is used in the context of asserting gradient diversity and quantifying the benefit in terms of scaling of the mean square error for mini-batch SGD for IID data. In (Schmidt & Roux, 2013; Vaswani et al., 2019), the authors use a similar assumption, called *strong growth condition*, which is a stronger version of Assumption 1 with $\epsilon = 0$.

They prove that some interesting practical problems satisfy such a condition. They also use this assumption to prove optimal and better convergence rates for SGD with constant step-sizes. Note that this is different from our approach as the algorithm that we are analyzing is not SGD, and our analysis is different in spite of the similarity in the assumptions.

C Simulation Details and Additional Experiments

C.1 Datasets and Models

Here we provide full details on the datasets and models used in our experiments. We curate a diverse set of non-synthetic datasets, including those used in prior work on federated learning (McMahan et al., 2017), and some proposed in LEAF, a benchmark for federated settings (Caldas et al., 2018). We also create synthetic data to directly test the effect of heterogeneity on convergence, as in Section 5.1.

- **Synthetic:** We set (α, β)=(0,0), (0.5,0.5) and (1,1) respectively to generate three non-identical distributed datasets (Figure 2). In the IID data (Figure 5), we set the same $W, b \sim \mathcal{N}(0, 1)$ on all devices and X_k to follow the same distribution $\mathcal{N}(v, \Sigma)$ where each element in the mean vector v is zero and Σ is diagonal with $\Sigma_{j,j} = j^{-1.2}$. For all synthetic datasets, there are 30 devices in total and the number of samples on each device follows a power law.

- **MNIST:** We study image classification of handwritten digits 0-9 in MNIST (LeCun et al., 1998) using multinomial logistic regression. To simulate a heterogeneous setting, we distribute the data among 1000 devices such that each device has samples of only 2 digits and the number of samples per device follows a power law. The input of the model is a flattened 784-dimensional (28×28) image, and the output is a class label between 0 and 9.

- **FEMNIST:** We study an image classification problem on the 62-class EMNIST dataset (Cohen et al., 2017) using multinomial logistic regression. To generate heterogeneous data partitions, we subsample 10 lower case characters ('a'-'j') from EMNIST and distribute only 5 classes to each device. We call this *federated* version of EMNIST *FEMNIST*. There are 200 devices in total. The input of the model is a flattened 784-dimensional (28×28) image, and the output is a class label between 0 and 9.

- **Shakespeare:** This is a dataset built from *The Complete Works of William Shakespeare* (McMahan et al., 2017). Each speaking role in a play represents a different device. We use a two-layer LSTM classifier containing 100 hidden units with an 8D embedding layer. The task is next-character prediction, and there are 80 classes of characters in total. The model takes as input a sequence of 80 characters, embeds each of the characters into a learned 8-dimensional space and outputs one character per training sample after 2 LSTM layers and a densely-connected layer.

- **Sent140:** In non-convex settings, we consider a text sentiment analysis task on tweets from Sentiment140 (Go et al., 2009) (Sent140) with a two layer LSTM binary classifier containing 256 hidden units with pretrained 300D GloVe embedding (Pennington et al., 2014). Each twitter account corresponds to a device. The model takes as input a sequence of 25 characters, embeds each of the characters into a 300-dimensional space by looking up Glove and outputs one character per training sample after 2 LSTM layers and a densely-connected layer.

C.2 Implementation Details

(**Implementation**) In order to draw a fair comparison with `FedAvg`, we use SGD as a local solver for `FedProx`, and adopt a slightly different device sampling scheme than that in Algorithms 1 and 2: sampling devices uniformly and averaging updates with weights proportional to the number of local data points (as originally proposed in (McMahan et al., 2017)). While this sampling scheme is not supported by our analysis, we observe similar relative behavior of `FedProx` vs. `FedAvg` whether or not it is employed (Figure 11). Interestingly, we also observe that the sampling scheme proposed herein results in more stable performance for both methods. This suggests an added benefit of the proposed framework.

(**Machines**) We simulate the federated learning setup (1 server and N devices) on a commodity machine with 2 Intel® Xeon® E5-2650 v4 CPUs and 8 NVidia® 1080Ti GPUs.

(**Hyperparameters**) We randomly split the data on each local device into an 80% training set and a 20% testing set. We fix the number of selected devices per round to be 10 for all experiments on all datasets. We also do a grid search on the learning rate based on `FedAvg`. We do not decay the learning rate through all rounds. For all synthetic data experiments, the learning rate is 0.01. For MNIST, FEMNIST, Shakespeare, and Sent140, we use the learning rates of 0.03, 0.003, 0.8, and 0.3. We use a batch size of 10 for all experiments.

(**Libraries**) All code is implemented in Tensorflow (Abadi et al., 2016) Version 1.10.1. They are available at bit.ly/FedProx.

C.3 Additional experiments and full results

C.3.1 Effects of Systems Heterogeneity on IID data

We show the effects of allowing for partial work on a perfect IID synthetic data (Synthetic IID).

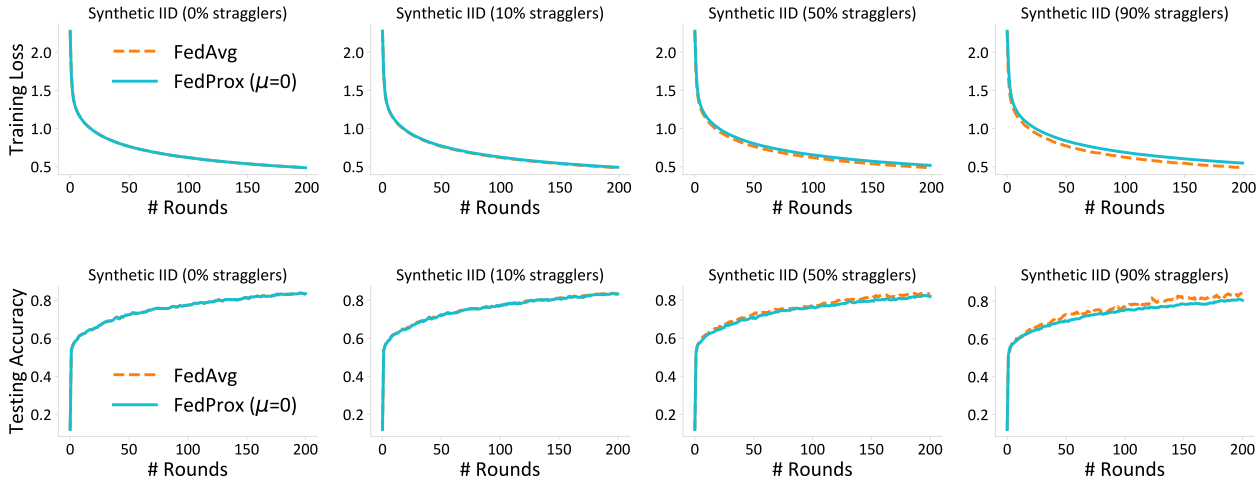

Figure 5. `FedAvg` is robust to device failure with IID data. In this case, whether incorporating partial solutions from the stragglers would not have much effect on convergence.

C.3.2 Complete Results

In Figure 6, we present testing accuracy on four synthetic datasets associated with the experiments shown in Figure 2.

In Figure 7, we show the testing accuracy associated with the experiments described in Figure 1. In Figure 8, we report the dissimilarity measurement on five datasets (including four real datasets) described in Figure 1. Again, the dissimilarity characterization is consistent with the real performance (the loss).

In Figure 9 and Figure 10, we show the effects of allowing for partial solutions (both loss and testing accuracy) under systems heterogeneity when $E = 1$ (i.e., the statistical heterogeneity is less likely to affect convergence negatively).

C.3.3 Comparing Two Device Sampling Schemes

We show the training loss, testing accuracy, and dissimilarity measurement of `FedProx` on a set of synthetic data using two different device sampling schemes in Figure 11. Since our goal is to compare these two sampling schemes, we let each device perform the uniform amount of work ($E = 20$) for both methods.

C.3.4 Adaptively setting μ

One of the key parameters of `FedProx` is μ. We provide the complete results of a simple heuristic of adaptively setting μ on four synthetic datasets in Figure 12. For the IID dataset (Synthetic-IID), μ starts from 1, and for the other non-IID datasets, μ starts from 0. Such initialization is adversarial to our methods. We decrease μ by 0.1 when the loss continues to decrease for 5 rounds and increase μ by 0.1 when we see the loss increase. This heuristic allows for competitive performance. It could also alleviate the potential issue that $\mu > 0$ might slow down convergence on IID data, which rarely occurs in real federated settings.

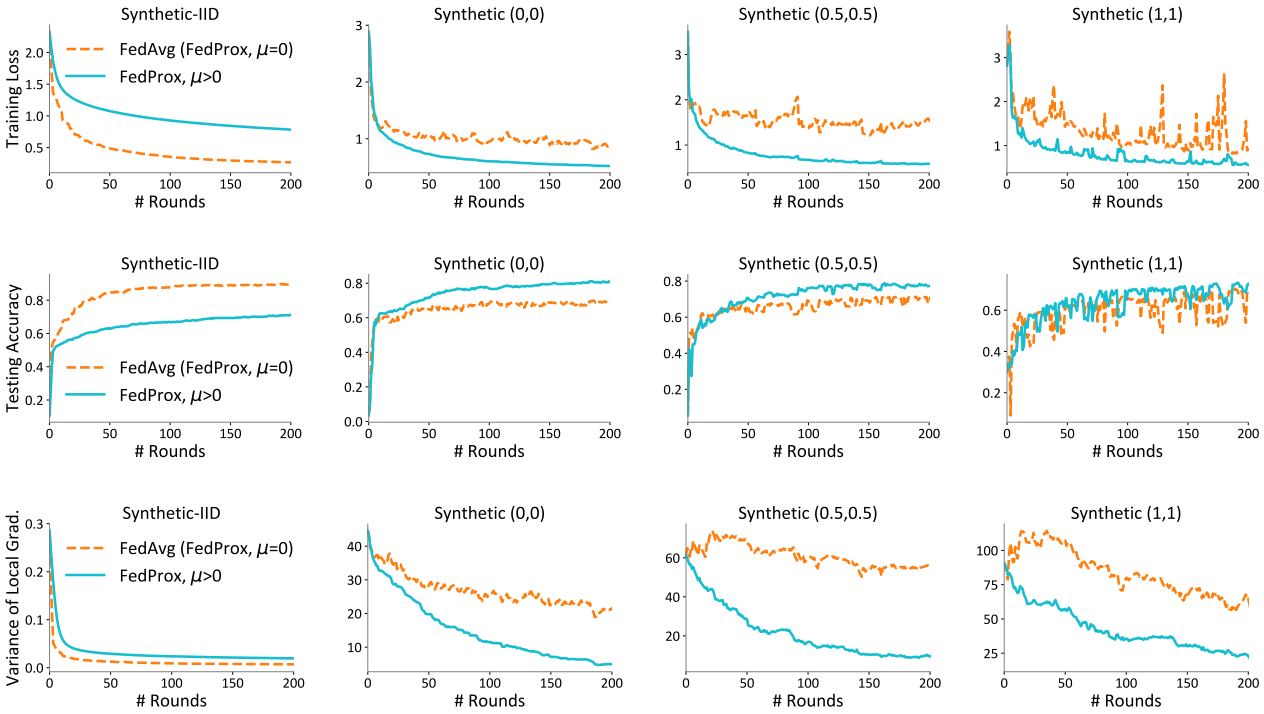

Figure 6. Full metrics (training loss, testing accuracy, and dissimilarity measurement) for experiments described in Figure 2.

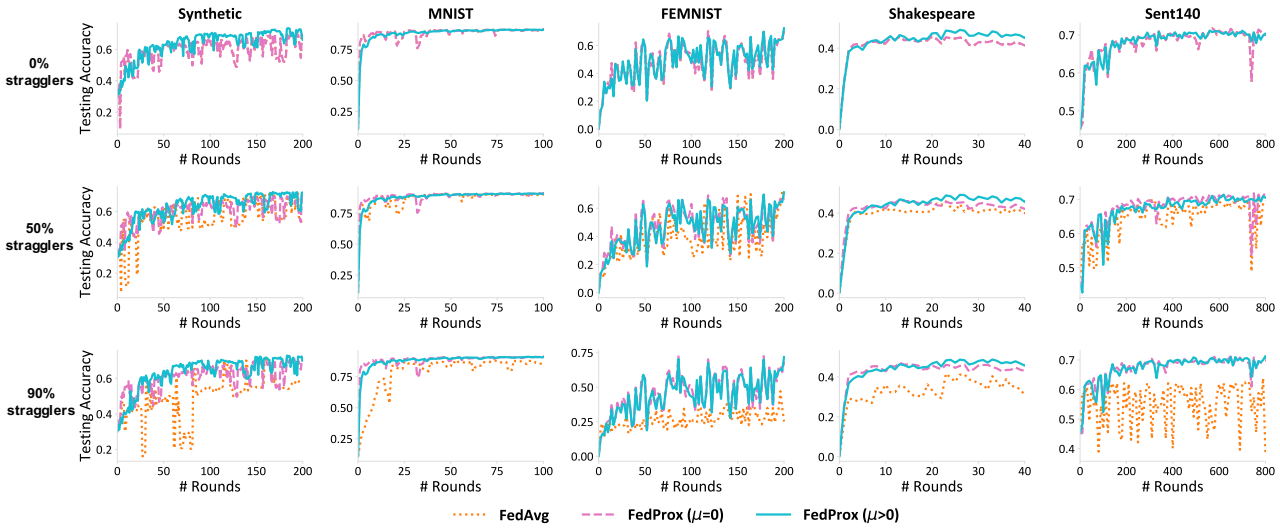

Figure 7. The testing accuracy of the experiments in Figure 1. `FedProx` achieves on average 18.8% improvement in terms of testing accuracy in highly heterogeneous settings (90% stragglers).

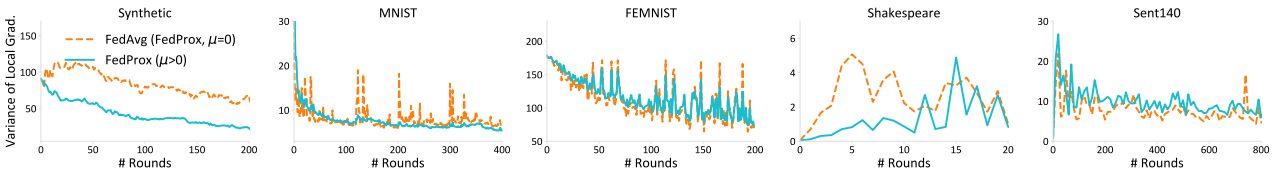

Figure 8. The dissimilarity metric on five datasets in Figure 1. We remove systems heterogeneity by only considering the case when no participating devices drop out of the network. Our dissimilarity assumption captures the data heterogeneity and is consistent with practical performance (see training loss in Figure 1).

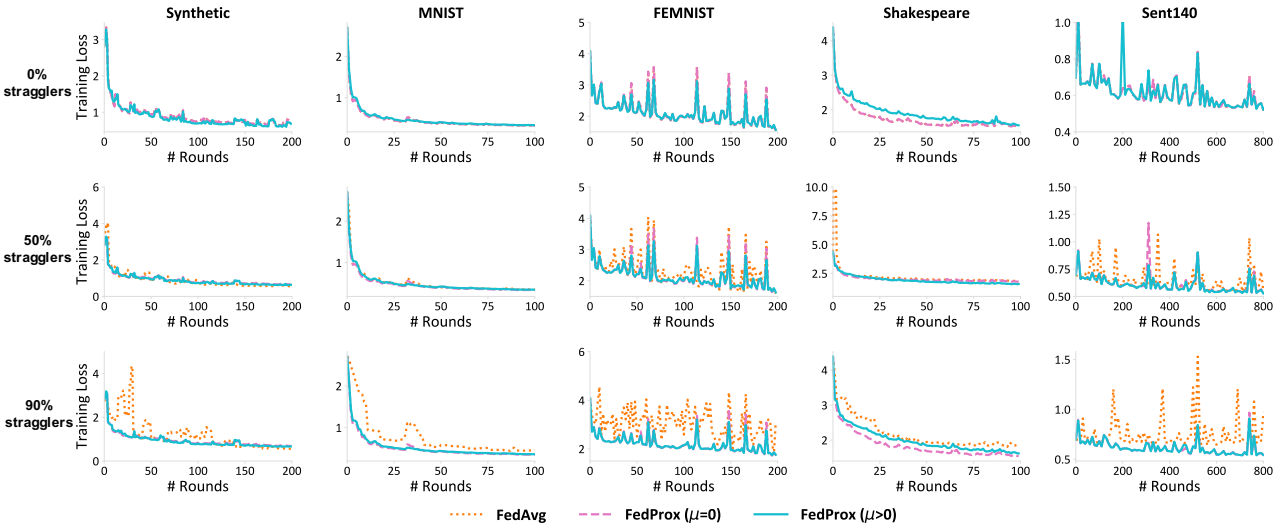

Figure 9. The loss of `FedAvg` and `FedProx` under various systems heterogeneity settings when each device can run at most 1 epoch at each iteration ($E = 1$). Since local updates will not deviate too much from the global model compared with the deviation under large E's, it is less likely that the statistical heterogeneity will affect convergence negatively. Tolerating for partial solutions to be sent to the central server (`FedProx`, $\mu = 0$) still performs better than dropping the stragglers (`FedAvg`).

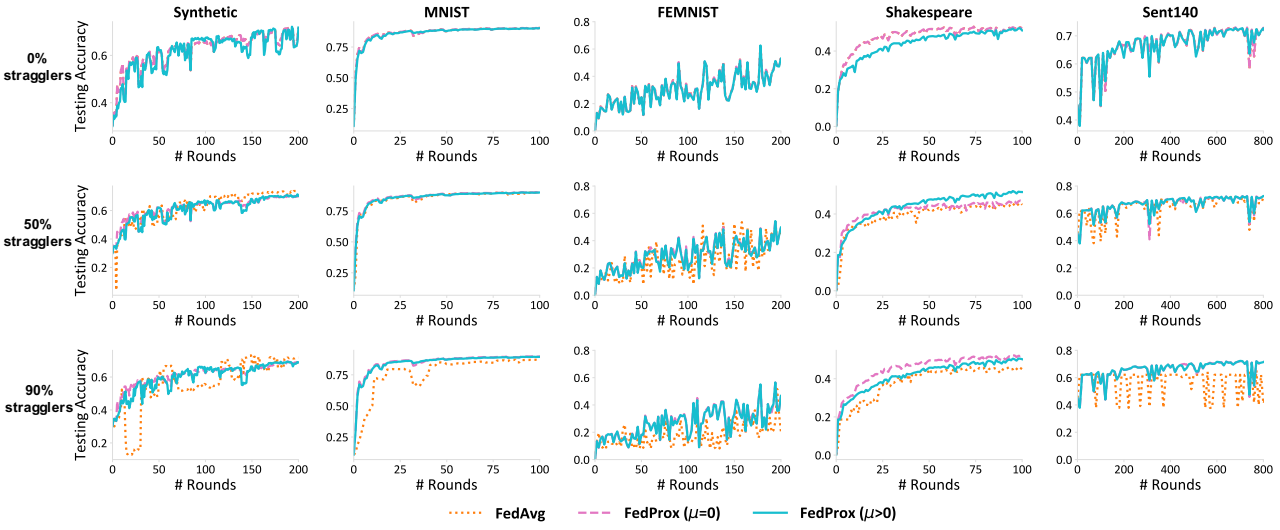

Figure 10. The testing accuracy of the experiments shown in Figure 9.

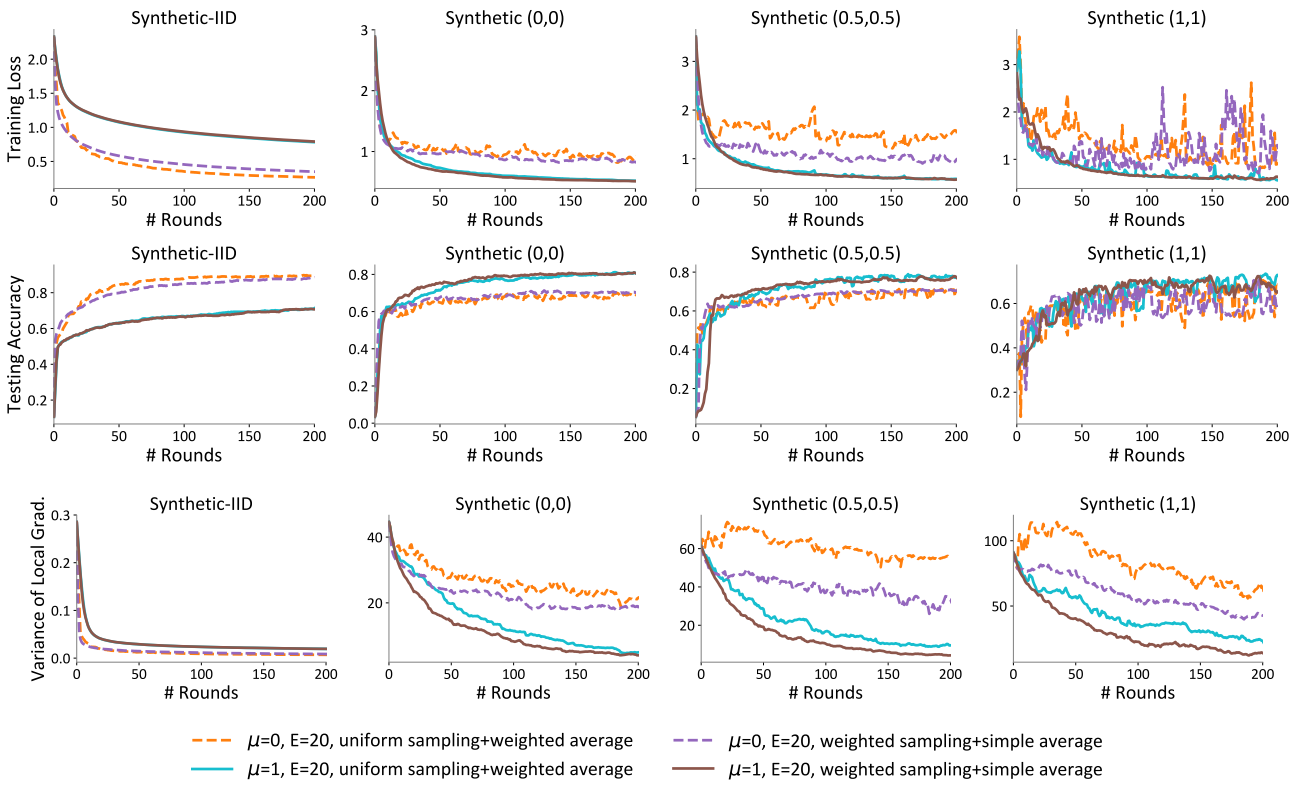

Figure 11. Differences between two sampling schemes in terms of training loss, testing accuracy, and dissimilarity measurement. Sampling devices with a probability proportional to the number of local data points and then simply averaging local models performs slightly better than uniformly sampling devices and averaging the local models with weights proportional to the number of local data points. Under either sampling scheme, the settings with $\mu = 1$ demonstrate more stable performance than settings with $\mu = 0$.

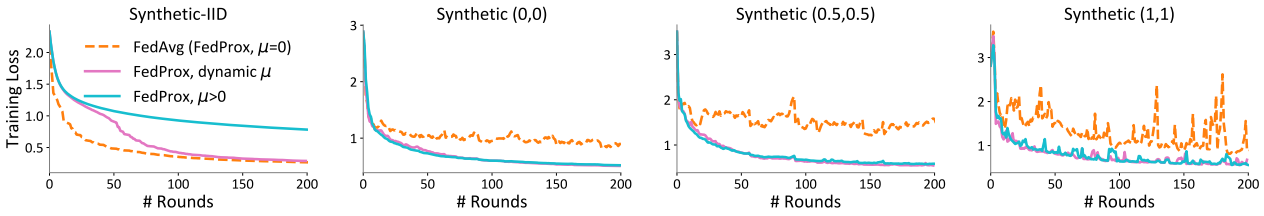

Figure 12. Full results of choosing μ adaptively on all the synthetic datasets. We increase μ by 0.1 whenever the loss increases and decreases it by 0.1 whenever the loss decreases for 5 consecutive rounds. We initialize μ to 1 for the IID data (Synthetic-IID) (in order to be adversarial to our methods), and initialize it to 0 for the other three non-IID datasets. We observe that this simple heuristic works well in practice.