Federated Optimization for Heterogeneous Networks

Anonymous Authors[1]

Abstract

Federated learning involves training machine learning models in massively distributed networks. While Federated Averaging (`FedAvg`) is the leading optimization method for training non-convex models in this setting, its behavior is not well understood in realistic federated settings when learning across statistically heterogeneous devices, i.e., where each device collects data in a non-identical fashion. In this work, we introduce a novel framework to tackle statistical heterogeneity, `FedProx`, which encompasses `FedAvg` as a special case. We prove the convergence of `FedProx` through a *device similarity* assumption, which allows us to characterize heterogeneity in the network. Finally, we perform a detailed empirical evaluation across a suite of federated datasets, validating our theoretical analysis and demonstrating improved performance of the generalized `FedProx` framework relative to `FedAvg` within heterogeneous networks. **AT:** can we add concrete numbers to support our empirical results? Or 'in terms of final accuracy, convergence speed and stability'?

1. Introduction

Large networks of remote devices, such as phones, vehicles, and wearable sensors, generate a wealth of data each day. Due to user privacy concerns and systems constraints (i.e., high communication costs, device-level computational constraints, and low availability amongst devices), federated learning has emerged as increasingly attractive paradigm to push the training of statistical models in such networks to the edge.

Optimization methods that allow for local updating and low participation have become the *de facto* solvers for federated learning (McMahan et al., 2016; Smith et al., 2017). These methods perform a variable number of local updates on a subset of devices to enable flexible and efficient communication patterns, e.g., compared to traditional distributed gradient descent or stochastic gradient descent (SGD). Of current federated optimization methods, `FedAvg` (McMahan et al., 2016) has become state-of-the-art for non-convex federated learning. `FedAvg` works simply by running some number of epochs, E, of SGD on a subset $K \ll N$ of the total devices N at each communication round, and then averaging the resulting model updates.

However, `FedAvg` was not designed to tackle the *statistical heterogeneity* inherent in federated settings, namely that each federated device collects data in a non-identical fashion, and the number of data points on each device may vary significantly. In realistic statistically heterogeneous settings, `FedAvg` has been shown to diverge empirically (McMahan et al., 2016, Sec 3), and it also lacks theoretical convergence guarantees. Indeed, recent works exploring convergence guarantees are limited to unrealistic scenarios, e.g., where (i) the data is either shared across devices or distributed in an IID (identically and independently distributed) manner, and (ii) all devices are involved in communication at each round (Stich, 2018; Wang & Joshi, 2018; Woodworth et al., 2018; Lin et al., 2018). While these assumptions simplify the analyses, they also violate key properties of realistic federated networks.

Contributions. In this work, we ask the following two questions: (1) Can we gain a principled understanding of `FedAvg` in realistic, statistically heterogeneous federated settings? (2) Can we devise an improved federated optimization algorithm, both theoretically and empirically? To this end, we propose a novel federated optimization framework, `FedProx`, which encompasses `FedAvg`. In order to characterize the convergence behavior of `FedProx` as a function of statistical heterogeneity, we introduce a novel *device similarity* assumption. Under this assumption, we provide the first convergence guarantees for `FedProx` in practical heterogeneous data settings. Furthermore, through a set of experiments on numerous real-world federated datasets, we demonstrate that `FedProx` can improve convergence significantly when data is heterogeneous across devices.

AT: can we add concrete numbers to support our empirical results?

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

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2. Related Work

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

Several recent methods have been proposed that are tailored to the specific challenges in the federated setting (Smith et al., 2017; McMahan et al., 2016; Wang et al., 2018). For example, Smith et al. (2017) proposes to learn 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., 2016).

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 federated networks. For instance, for a classification task, heterogeneity in the data may appear with each device only having data corresponding to a small subset of the total classes. Moreover, heterogeneity may also be present in terms of the number of total samples per device. Recent works have made steps towards analyzing `FedAvg` in simpler, non-federated settings. For instance, parallel SGD (Stich, 2018; Wang & Joshi, 2018), which makes local updates similar to `FedAvg`, has been studied in the IID setting. However, the main ingredient of the proof is the observation that each local SGD 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, the heterogeneity assumption was explored in (Zhao et al., 2018), it was assumed that all devices are active at each round, which again, violates the properties of federated settings.

There are also some heuristic approaches trying to tackle the statistical heterogeneity problem, mainly through data sharing strategies (Jeong et al., 2018; Zhao et al., 2018; Huang et al., 2018). However, those methods may be unrealistic in practical federated settings. In addition to imposing burdens on network bandwidth, sending local data to the server (Jeong et al., 2018) is unfriendly towards privacy protection; and sending globally-shared data to all devices (Zhao et al., 2018; Huang et al., 2018) requires effort to carefully generate and collect such a dataset.

In this work, inspired by `FedAvg`, we propose a broader framework, `FedProx`, that is capable of handling heterogeneous federated data. We analyze the convergence behavior of the framework under a novel local similarity assumption between local functions. Our similarity assumption is inspired by the Kaczmarz method for solving linear system of equations (Kaczmarz, 1993). A similar assumption has been previously used to analyze variants of SGD for strongly convex problems (see, e.g., Schmidt & Roux, 2013). However, to the best of our knowledge, this is the first convergence analysis of any methods for federated optimization with heterogeneous data.

3. Federated Optimization: Algorithms

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., 2016; Smith et al., 2017; Wang & Joshi, 2018; Lin et al., 2018) are designed to handle multiple devices[1] collecting data and a central server coordinating the learning objective. To reduce communication and handle systems constraints, a common theme 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. Technically speaking, we aim to minimize the following global objective function:

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

where N is the number of devices, $p_k \geq 0$, and $\sum_k p_k = 1$. In general, the local objectives $F_k(\cdot)$'s are given by local empirical risks, i.e., they are average risks over n_k samples available locally at 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.

At each outer iteration, a subset of the devices are selected and local solvers are used to optimize the local objective functions at 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*. We introduce this notion formally below, as it will be utilized throughout the paper.

[1] We use the term 'device' throughout the paper to describe entities in the network, e.g., nodes, clients, phones, sensors.

Definition 1 (γ-inexact solution)**.** For a smooth convex 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.

3.1. Federated Averaging (`FedAvg`)

In Federated Averaging (`FedAvg`) (McMahan et al., 2016), the local surrogate of the global objective function at device k is taken to be $F_k(\cdot)$ and the local solver is chosen to be stochastic gradient descent (SGD), which is homogeneous across devices in terms of the algorithm hyperparameters, i.e., the learning rate and the number of local epochs. The details of `FedAvg` are summarized in Algorithm 1.

Algorithm 1: Federated Averaging (`FedAvg`)

INPUT: K, T, η, E, w^0, N, p_k, $k = 1, \cdots, N$;
forall $t = 0, \cdots, T-1$ **do**
  Server chooses K devices at random (each device k is chosen with probability p_k);
  Server sends w^t to all chosen devices.;
  Each device k updates w^t for E epochs of SGD on F_k with step-size η to obtain w_k^t.;
  Each chosen device k sends w_k^{t+1} back to the central server.;
  Server aggregates the w's as $w^{t+1} = \frac{1}{K}\sum_k w_k^{t+1}$;

McMahan et al. show empirically that it is crucial to tune the optimization hyperparameters properly to get `FedAvg` to work in heterogeneous settings. In particular, carefully tuning the number of local epochs is critical in order to make `FedAvg` converge, as larger number of local epochs allow local models to move further away from the initial global model which leads to potential divergence. Intuitively speaking, in the face of dissimilar (heterogeneous) local objectives F_k, a larger number of local epochs may lead each local device towards the optima of its local objective as opposed to the global objective. Therefore, in a heterogeneous setting, where the local objectives may be quite different from the global, it would be beneficial to limit the amount of local updates through a more flexible tool beyond heuristically limiting the number of local updates. A natural way to enforce limited local model updates is to incorporate a constraint which penalizes large changes from the current model at the server. This observation serves as inspiration for `FedProx`, introduced below.

3.2. Proposed Framework: `FedProx`

Our proposed framework, `FedProx`, is similar to `FedAvg` in that a subset of devices are selected at each round, a number of local updates are performed, and local updates are then aggregated to form a global update. However, for each device k, instead of just minimizing the local function F_k, device k uses its local solver of choice to approximately minimize the following surrogate objective h_k:

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

Note that the proximal term in the above expression effectively limits the impact of local updates (by restricting them to be close to the initial model) without any need to manually tune the number of local epochs as in `FedAvg`. We summarize the steps of `FedProx` in Algorithm 2. In our experiments (Section 5.2), we see the modified local subproblem in `FedProx` results in markedly improved performance compared to vanilla `FedAvg` for heterogeneous datasets. As we will see in Section 4, the usage of the proximal term in `FedProx` also makes it more amenable for theoretical analysis. Note that `FedAvg` is a special case of `FedProx` with $\mu = 0$.

Algorithm 2: FedProx

INPUT: K, T, μ, γ, w^0, N, p_k, $k = 1, \cdots, N$;
forall $t = 0, \cdots, T-1$ **do**
  Server choose 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 finds a w_k^{t+1} which is a γ-inexact minimizer of: $w_k^{t+1} \approx \arg\min_w\ h_k(w;\ w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|$;
  Each chosen device k sends w_k^{t+1} back to the Server.;
  Server aggregates the w's as $w^{t+1} = \frac{1}{K}\sum_k w_k^{t+1}$;

4. Federated Optimization: Convergence Analysis

`FedAvg` and `FedProx` are stochastic algorithms by nature; in each step of these algorithms, 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 such methods with constant step-size, which is what is usually deployed in practice, we need to be able 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 propose a metric that specifically measures the dissimilarity among local functions (Section 4.1) and analyze `FedProx` under this assumption in Section 4.2.

4.1. Local dissimilarity

Here we introduce a measure of dissimilarity between the devices in a federated network, which we use throughout our analysis. We also relate this definition to a more common and simplified notion in Assumption 1 (that the gradients have bounded variance), which we explore in our experiments in Section 5.

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

Note that in this definition, a smaller value of $B(w)$ at any iterate w implies that the local functions are more locally similar. Also, $B(w) \geq 1$ by definition. In the extreme case when all the local functions are the same, we have $B(w) = 1$ for all w. 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. However, in the federated setting the data distributions are heterogeneous. And even if the samples are IID on each device, in the finite sample case, $B > 1$ due to the sampling discrepancies. Thus, it is natural to think of the case where $B > 1$ and our definition of dissimilarity as a generalization of the IID assumption when the local distributions are heterogeneous but not very dissimilar. The hope is that although the data points are not IID, the dissimilarity B would not be that large throughout the training process.

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

In most practical machine learning settings, especially in the federated setup, there is no need to solve the problem to arbitrarily accurate stationary solutions to get good generalization, i.e., ϵ is typically not very small. In fact, empirical data shows that solving the problem beyond some threshold might even hurt the generalization performance due to overfitting. Although in practical federated learning problems the samples are not IID, they are still sampled from distributions that are not drastically different. Thus, the dissimilarity between local functions would potentially stay bounded throughout most of the training process.

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

4.2. `FedProx` Analysis

With the bounded dissimilarity Assumption 1 in place, we can now analyze the amount of expected decrease in the objective when one step of `FedProx` is performed under this assumption.

Lemma 3 (`FedProx` Convergence: B-local dissimilarity)**.** *Let Assumption 1 hold. Assume the functions F_k are L-Lipschitz smooth and also there exists L_-, such that $\nabla^2 F_k \succeq -L_- \mathbf{I}$ and define $\bar{\mu} = \mu - L_- > 0$. Suppose that w^t is not a stationary solution and local functions F_k are B-dissimilar, i.e. $B(w^t) \leq B$, then if μ, K and γ in Algorithm 2 are chosen such that*

$$\rho = \left(\frac{1}{\mu} - \frac{\gamma B}{\mu} - \frac{B(1+\gamma)}{\bar{\mu}\sqrt{K}} - \frac{LB(1+\gamma)}{\bar{\mu}\mu} - \frac{L(1+\gamma)^2 B^2}{2\bar{\mu}^2} - \frac{LB^2(1+\gamma)^2}{\bar{\mu}^2 K}\left(2\sqrt{K}+1\right)\right) > 0, \quad (3)$$

then at iteration t of the `FedProx` Algorithm 2, we have the following expected decrease in the global objective $\mathbb{E}_{S_t} f(w^{t+1}) \leq f(w^t) - \rho\|\nabla f(w^t)\|^2$, where the expectation is over the set S_t of K devices chosen at iteration t.

Proof. First of all note that based on our inexactness assumption, we can define e_k^{t+1} such that

$$\nabla F_k(w_k^{t+1}) + \mu(w_k^{t+1} - w^t) - e_k^{t+1} = 0,$$
$$\|e_k^{t+1}\| \leq \gamma\|\nabla F_k(w^t)\| \quad (4)$$

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

$$\bar{w}^{t+1} - w^t = \frac{-1}{\mu}\mathbb{E}\,\nabla F_k(w_k^{t+1}) + \frac{1}{\mu}\mathbb{E}e_k^{t+1}. \quad (5)$$

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

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

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

$$\|w_k^{t+1} - w^t\| \leq \frac{1+\gamma}{\bar{\mu}}\|\nabla F_k(w^t)\|. \quad (7)$$

Therefore,

$$\|\bar{w}^{t+1} - w^t\| \leq \mathbb{E}_k\|w_k^{t+1} - w^t\| \leq \frac{1+\gamma}{\bar{\mu}}\mathbb{E}_k\|\nabla F_k(w^t)\|$$
$$\leq \frac{1+\gamma}{\bar{\mu}}\sqrt{\mathbb{E}_k\left[\|\nabla F_k(w^t)\|^2\right]} \leq \frac{B(1+\gamma)}{\bar{\mu}}\|\nabla f(w^t)\|, \quad (8)$$

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

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

$$\|E_{t+1}\| \leq \mathbb{E}_k\left[L\|w_k^{t+1} - w_k^t\| + \|e_k^{t+1}\|\right] \leq \left(\frac{L(1+\gamma)}{\bar{\mu}} + \gamma\right)$$

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

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

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

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 $\|\nabla f(w^t)\|^2$. But 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 $\mathbb{E}f(w^{t+1})$, 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

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

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

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

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.

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

From (8), we have that $\|\bar{w}^{t+1} - w^t\| \leq \frac{B(1+\gamma)}{\bar{\mu}}\|\nabla f(w^t)\|$. Moreover,

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

and

$$\mathbb{E}_{S_t}\left[\|w^{t+1} - \bar{w}^{t+1}\|^2\right] \leq \frac{1}{K}\mathbb{E}_k[\|w_k^{t+1} - \bar{w}^{t+1}\|^2]$$
$$\leq \frac{1}{K}\mathbb{E}_k[\|w_k^{t+1} - w^t\|^2], \quad (\text{as } \bar{w}^{t+1} = \mathbb{E}_k\, w_k^{t+1})$$
$$\leq \frac{1}{K}\frac{(1+\gamma)^2}{\bar{\mu}^2}\mathbb{E}_k\|\nabla F_k(w^t)\|^2 \quad (\text{from (7)})$$
$$\leq \frac{B^2}{K}\frac{(1+\gamma)^2}{\bar{\mu}^2}\|\nabla f(w^t)\|^2, \tag{16}$$

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

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

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

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

□

Lemma 3 uses the similarity assumption in Definition 2 to characterize sufficient decrease at each iteration for `FedProx`. Below, provide a corollary that describes the performance using the simpler bounded variance assumption in Assumption 1.

Corollary 4 (Bounded Variance Equivalence)**.** *Let Assumption 1 hold. Then, in the case of bounded variance, i.e. $E_k[\|\nabla F_k(w) - \nabla f(w)\|^2] \leq \sigma^2$, which is essential for analyzing SGD, then for any $\epsilon > 0$, it follows that $B_\epsilon \leq \sqrt{1 + \frac{\sigma^2}{\epsilon}}$.*

With the above corollary 4 in place, we can restate the main result in Lemma 3 in terms of the bounded variance assumption.

Lemma 5 (`FedProx` Convergence: Bounded Variance)**.** *Let the assertions of lemma 3 hold. In addition, let the iterate w^t be such that $\|\nabla f(w^t)\|^2 \geq \epsilon$. Furthermore let $E_k[\|\nabla F_k(w) - \nabla f(w)\|^2] \leq \sigma^2$ hold instead of the similarity condition. Then if μ, K and γ in Algorithm 2 are*

chosen such that

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

then at iteration t of the `FedProx` *Algorithm 2, we have the following expected decrease in the global objective $\mathbb{E}_{S_t} f(w^{t+1}) \leq f(w^t) - \rho\|\nabla f(w^t)\|^2$, where the expectation is over the set S_t of K devices chosen at iteration t.*

The proof of Lemma 5 follows from the proof of Lemma 3 by noting the relationship between the bounded variance assumption and the similarity assumption as portrayed by Corollary 4.

Corollary 6 (Convergence: Convex Case)**.** *Let the assertions of Lemma 3 hold. In addition, let $F_k(\cdot)$'s be convex and $\gamma = 0$, i.e., all the local problems are solved accurately, if $1 \ll B \leq 0.5\sqrt{K}$, then we can choose $\mu \approx 6LB^2$ from which it follows that $\rho \approx \frac{1}{24LB^2}$.*

Remark 7. *In order for ρ in Lemma 3 to be positive, we need $\gamma B < 1$. Moreover, we also need $\frac{B}{\sqrt{K}} < 1$. Note that these conditions might be restrictive due to the worst case nature of our analysis. Nevertheless, they quantify the trade-off between dissimilarity bound and the method's parameter.*

We can use the above sufficient decrease to obtain a convergence to the set of approximate stationary solutions $\mathcal{S}_s = \{w \mid \mathbb{E}\|\nabla f(w)\|^2 \leq \epsilon\}$ under the bounded dissimilarity assumption, Assumption 1.

Corollary 8 (Convergence rate: `FedProx`)**.** *Given some $\epsilon > 0$, assume that for $B \geq B_\epsilon$, μ, γ and K the assumptions of Lemma 3 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 $\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(w^t)\|^2 \leq \epsilon$.*

Remark 9 (Comparison with SGD)**.** *Note that* `FedProx` *gets the same asymptotic convergence guarantee as SGD. In other words, under bounded variance assumption, for very small ϵ, if we replace B_ϵ with its upper-bound in Corollary 4 and choose μ large enough, then the iteration complexity of* `FedProx` *when the subproblems are solved exactly and $F_k(\cdot)$'s are convex would be $O(\frac{L\Delta}{\epsilon} + \frac{L\Delta\sigma^2}{\epsilon^2})$ which is the same as SGD (Ghadimi & Lan, 2013).*

Small ϵ in Assumption 1 translates to larger B_ϵ. Corollary 6 suggests that, in order to solve the problem with increasingly higher accuracies using `FedProx` one needs to increase μ appropriately. Moreover, in Corollary 6, if we plug in the upper bound for B_ϵ, under bounded variance assumption (see Corollary 4), we get the number of required steps to achieve accuracy ϵ as $O(\frac{L\Delta}{\epsilon} + \frac{L\Delta\sigma^2}{\epsilon^2})$. Our analysis captures the weaknesses of `FedProx` and similar methods when the local functions are dissimilar. As a future direction, it would be interesting to quantify lower bounds for the convergence of the methods such as `FedProx`/`FedAvg` in settings involving heterogeneous data and devices.

Remark 10 (Connection to Elastic SGD)**.** *As a special case (when using SGD locally, updating all devices at once, and aggregating updates via simple averaging), Elastic SGD (Zhang et al., 2015) can be seen as an instantiation of* `FedProx`*. Thus, the obtained results for* `FedProx` *could be specialized to obtain a more general convergence guarantee for Elastic SGD in heterogeneous settings, which is of independent interest.*

5. Experiments

We now present empirical results for the `FedProx` framework. In Section 5.2, we study the effect of statistical heterogeneity on the convergence of `FedAvg` and `FedProx`. We explore properties of the `FedProx` framework, such as the effect of μ and the local epochs E, in Section 5.3. Finally, in Section 5.4, we show how empirical convergence is related to the theoretical dissimilarity assumption presented in Section 4. 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.

5.1. Experimental Set Up

To comprehensively demonstrate the performance of `FedProx` with heterogeneous data, we evaluate on diverse tasks, models, and federated datasets. In order to characterize statistical heterogeneity and study its effect on convergence, we also evaluate our methods on a set of synthetic data, which allows for more precise manipulation.

Synthetic data. To generate synthetic data, we follow a similar setup to that described in (Shamir et al., 2014), and optionally impose heterogeneity between devices. In particular, for each device k, we generate synthetic 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 generate one IID dataset by setting the same W, b on all devices and each X_k to follow the same distribution. We vary α, β to generate three heterogeneous distributed datasets (Synthetic (α, β)), as shown in Figure 1. Our goal is to learn a global W and b. See full details in Appendix C.1.

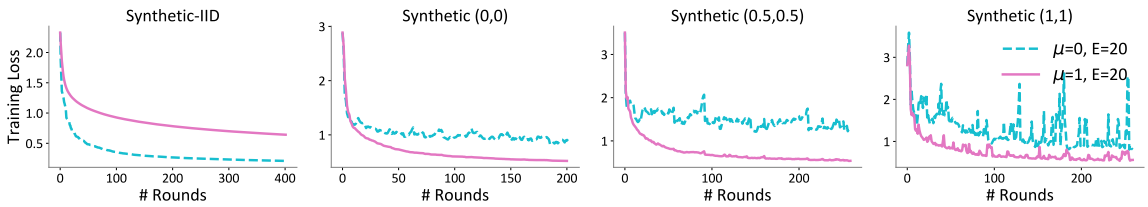

Figure 1. Effect of data heterogeneity on convergence. We show training loss (see testing accuracy and dissimilarity metric in Figure 5 in the appendix) on four synthetic datasets whose heterogeneity increases from left to right. Note that methods where $\mu = 0$ corresponds to `FedAvg`. The increasing heterogeneity leads to worse convergence and large μ is particularly useful in heterogeneous settings.

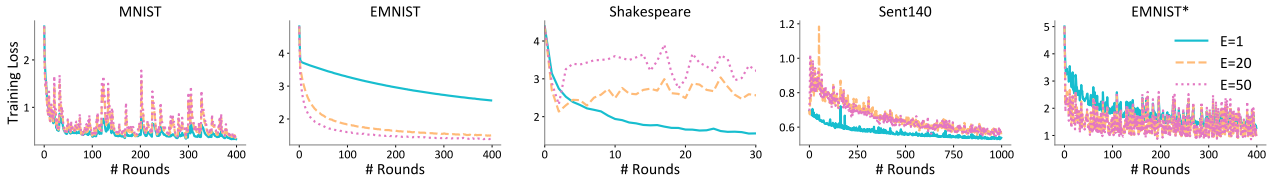

Figure 2. Effect of increasing E on real federated datasets where $\mu = 0$ (corresponds to `FedAvg`). Too many local updates would cause divergence or instability for heterogeneous datasets. Note that EMNIST* is a more skewed version of EMNIST.

Real data. We explore five non-synthetic datasets: MNIST, EMNIST, EMNIST*, Sent140, and Shakespeare, as summarized in Table 1. We begin with a simple convex setting of image classification of handwritten digits 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. We then study a harder classification problem on the 62-class Extended MNIST (Cohen et al., 2017) (EMNIST) dataset using the same model. Each device corresponds to a writer of the digits/characters. We also twist EMNIST to create a more heterogeneous dataset, EMNIST*. In non-convex settings, we consider a text sentiment analysis task on tweets from Sentiment140 (Go et al., 2009) (Sent140) with a LSTM classifier. Each twitter account corresponds to a device. Finally, we consider a dataset built from *The Complete Works of William Shakespeare* (McMahan et al., 2016), also using a LSTM for next character prediction and each speaking role in a play is a different device. Full details are provided in Appendix C.1.

Protocol. For each experiment, we tune the learning rate and ratio of active devices per round, and report results using the hyperparameters that perform best on `FedAvg`. We randomly split the data on each local device into 80% training set and 20% testing set. For each comparison, we set the random seeds to make sure that the devices selected and data read at each round are the same across all runs.

Table 1. Statistics of real federated datasets

Dataset	Devices	Samples	Samples/device	
			mean	stdev
MNIST	1,000	69,035	69	106
EMNIST	900	305,654	340	107
Shakespeare	143	517,706	3,620	4,115
Sent140	5,726	215,829	38	19
EMNIST*	200	79,059	395	873

5.2. Impacts of Data Heterogeneity

In Figure 1, we study how data heterogeneity affects convergence when the number of local epochs is large with four synthetic datasets. We fix E to be 20. From left to right, as data become more heterogeneous, convergence becomes worse for `FedProx` with both $\mu = 0$ and $\mu = 1$. However, larger μ is particularly useful in the heterogeneous set up, as evident from Figure 1. This indicates that `FedProx` can benefit practical federated settings with varying statistical heterogeneity. In the sections below, we see similar results in our non-synthetic experiments.

5.3. Properties of `FedProx` Framework

The key parameters of `FedProx` that affect its performance are the number of local epochs, E, and the proximal term, μ. Intuitively, large E may cause local models to drift too far away from the initial point, thus leading to po-

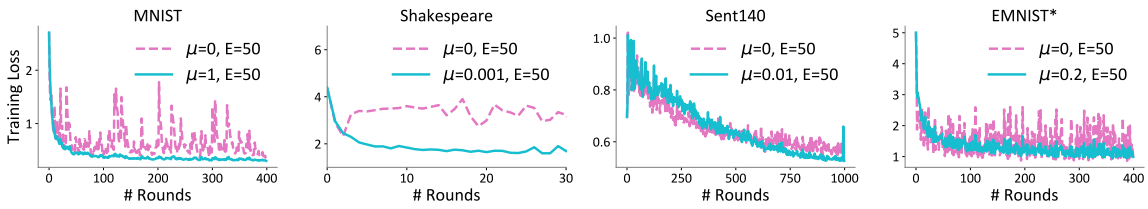

Figure 3. Effect of μ on four real datasets. The setting $\mu = 0$ corresponds to `FedAvg`. By setting μ appropriately, `FedProx` could increase the stability for unstable methods and can force divergent methods to converge.

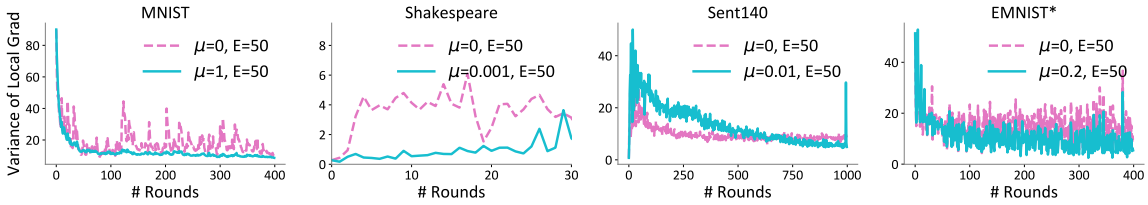

Figure 4. The dissimilarity measurement (variance of gradients) on four federated datasets. This metric captures the heterogeneity of datasets and is consistent with training loss. Smaller dissimilarity indicates better convergence.

tential divergence; while large μ can restrict the trajectory of the iterates by constraining the iterates to be closer to that of the global model, potentially slowing convergence. We study `FedProx` under different values of E and μ using the federated datasets described in Table 1.

Dependence on E. We study the effect of E in Figure 2. For each dataset, we set E to be 1, 20 and 50 while keeping $\mu = 0$ (`FedProx` reduces to `FedAvg` in this case) and show the training loss. We see that large E leads to divergence or instability on MNIST and Shakespeare datasets, which are in fact more heterogeneous. On EMNIST and Sent140, nevertheless, larger E speeds up the convergence. Based on conclusions drawn from Figure 1, we hypothesize this is due to the fact that the data distributed across devices after partitioning EMNIST and Sent140 are *homogeneous*. Note that if datasets are completely IID on each local device, then training more local epoches would naturally benefit convergence. We validate this hypothesis by observing instability on EMNIST*, which is a more skewed dataset downsampled from EMNIST. The divergence or instability indicates that `FedAvg` is problematic when communication is highly expensive (thus requiring large E) and there is significant heterogeneity across devices. Consequently, we demonstrate the impacts of μ using EMNIST* instead of EMNIST.

Dependence on μ. We consider the impacts of μ on convergence in Figure 3. For each experiment, in the case of $E = 50$, we compare the results between $\mu = 0$ and the best μ. From the four datasets except Sent140, we observe that appropriate μ can increase the stability for unstable methods and can force divergent methods to converge. It may also increase the accuracy in some cases. However, on Sent140, μ does not help speed up convergence. In practice, μ can be chosen based on specific data characteristics and communication patterns.

5.4. Dissimilarity Measurement and Divergence

In Figure 4, we demonstrate our dissimilarity measurement (see Definition 2) captures the heterogeneity of datasets and therefore it is a 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 Corollary 4). Empirically, either decreasing E (Figure 6 in the appendix) or increasing μ (Figure 4) leads to larger similarity among local functions. We also observe that the dissimilarity metric is consistent with the training loss. Therefore, smaller dissimilarity indicates better convergence, which can be enforced by setting μ appropriately.

Tian will make the colors of the lines in the figures consistent

6. Conclusion

We propose `FedProx`, a distributed optimization framework that tackles the heterogeneity inherent in federated networks. We formalize statistical heterogeneity through a novel *device similarity* assumption which allows us to characterize the convergence of `FedProx`. Our empirical evaluation across a suite of federated datasets validates our the-

oretical analysis and demonstrates that FedProx results in improved convergence with increasing data heterogeneity.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

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.

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*, 1(12), 2009.

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.

Jaggi, M., Smith, V., Takác, M., Terhorst, J., Krishnan, S., Hofmann, T., and Jordan, M. I. Communication-efficient distributed dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pp. 3068–3076, 2014.

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.

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, 2014.

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. *arXiv preprint arXiv:1602.05629*, 2016.

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.

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.

Sheller, M. J., Reina, G. A., Edwards, B., Martin, J., and Bakas, S. Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation. *arXiv preprint arXiv:1810.04304*, 2018.

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*, 18:1–47, 2018.

Stich, S. U. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

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. *arXiv preprint arXiv:1804.05271*, 2018.

Woodworth, B., Wang, J., McMahan, B., and Srebro, N. Graph oracle models, lower bounds, and gaps for parallel stochastic optimization. *arXiv preprint arXiv:1805.10222*, 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.

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.

A. Proof of Corollary 4

We have,

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

B. Proof of Corollary 6

In the convex case, where $L_- = 0$ and $\bar{\mu} = \mu$, if $\gamma = 0$, i.e., all subproblems are solved accurately, 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

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

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

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

Therefore, the number of iterations to at least generate one solution with squared norm of gradient less than ϵ is $O(\frac{LB^2\Delta}{\epsilon})$.

C. Experimental Details

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., 2016), 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 1). In the IID data, we set the same $W, b \sim \mathcal{N}(0, 1)$ on all devices and $X_k \sim \mathcal{N}(v, \Sigma)$ where each element in v is drawn from $\mathcal{N}(0, 1)$ 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.
- **EMNIST:** We study an image classification problem on the 62-class EMNIST dataset (Cohen et al., 2017) using multinomial logistic regression. Each device corresponds to a writer of the digits/characters in EMNIST. The input of the model is a flattened 784-dimensional (28×28) image, and the output is a class label between 0 and 61.
- **Shakespeare:** It is a dataset built from *The Complete Works of William Shakespeare* (McMahan et al., 2016). Each speaking role in a play represents a different device. We use a two layer LSTM classifier containing 100 hidden units with a 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 character 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 character 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.

- **EMNIST*:** We generate EMNIST* by subsampling 26 lower case characters from EMNIST and distributing only 20 classes to each device. There are 200 devices in total. The model is the same with the one used on EMNIST.

C.2. Implementation Details

(**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.

(**Hyper-parameters**) For each dataset, we tune the ratio of active clients per round from $\{0.01, 0.05, 0.1\}$ on `FedAvg`. For synthetic datasets, there are about 10% active devices at each round. For MNIST, EMNIST, Shakespeare, Sent140 and EMNIST*, the active devices are 1%, 5%, 10%, 1% and 5% respectively. 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, EMNIST, Shakespeare, Sent140 and EMNIST*, we use the learning rates of 0.03, 0.003, 0.8, 0.3 and 0.003. We use a batch size of 10 for all experiments.

(**Libraries**) All codes are implemented in Tensorflow (Abadi et al., 2015) Version 1.10.1. See our anonymzied code submission for full details.

C.3. Full Experiments

We present testing accuracy, training loss and dissimilarity measurements of all the experiments.

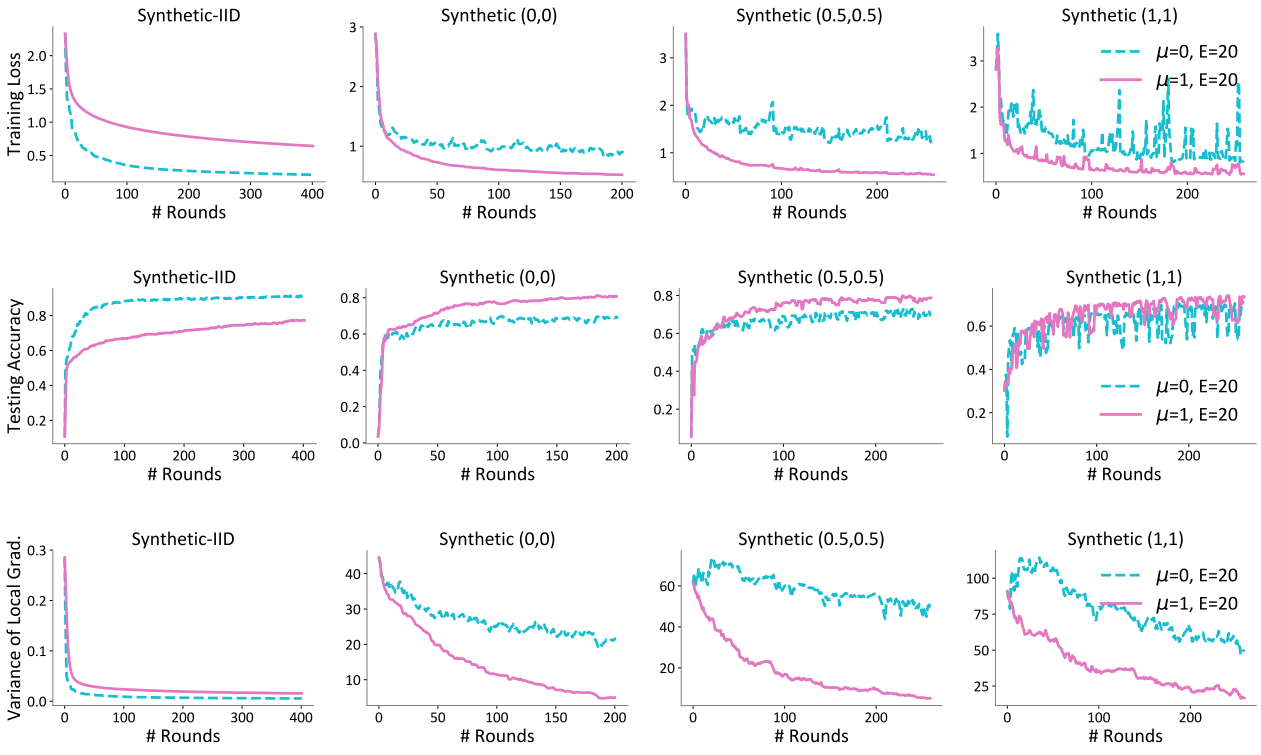

Figure 5. Training loss, testing accuracy and dissimilarity measurement for experiments in Figure 5

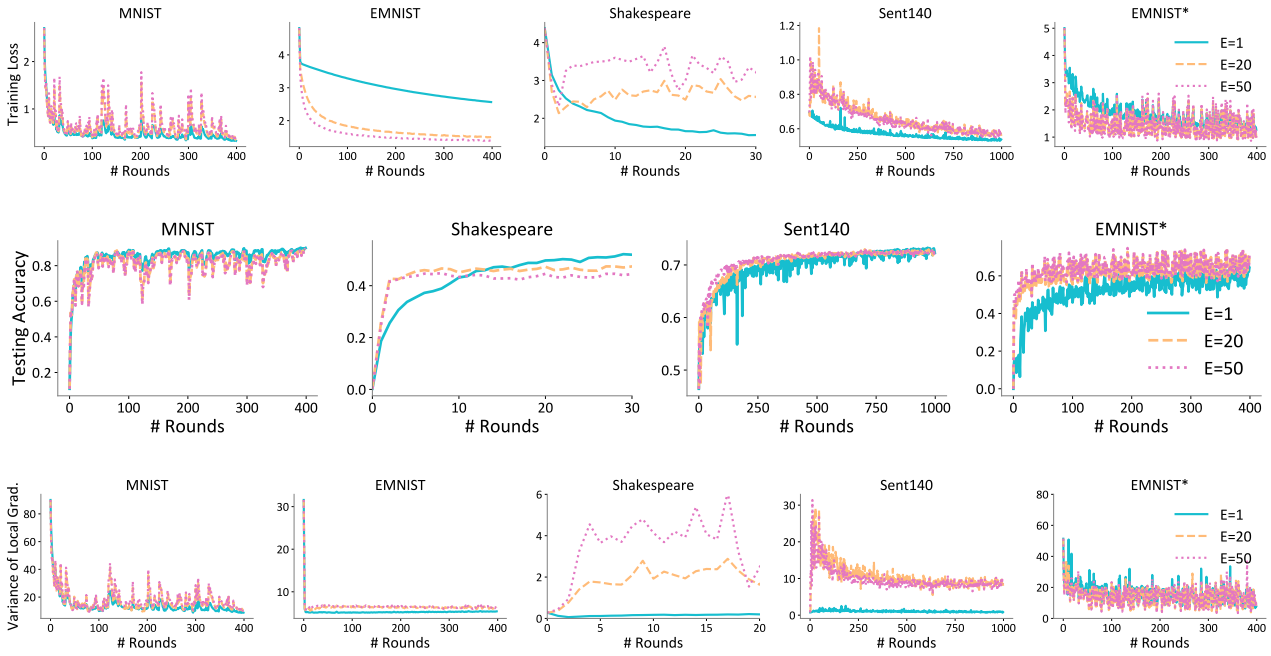

Figure 6. Training loss, testing accuracy and dissimilarity measurement for experiments in Figure 2

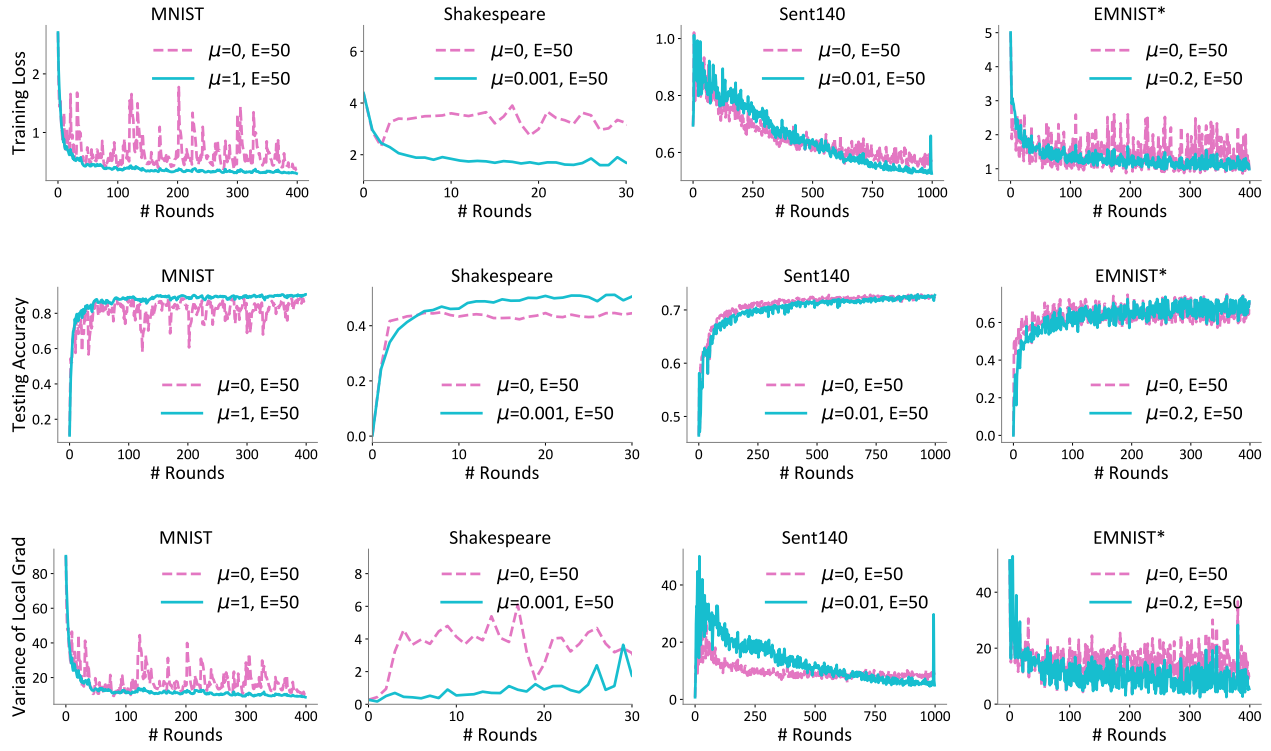

Figure 7. Training loss, testing accuracy and dissimilarity measurement for experiments in Figure 3