Convergence and Accuracy Trade-Offs in Federated Learning and Meta-Learning

Zachary Charles
Google Research

Jakub Konečný
Google Research

Abstract

We study a family of algorithms, which we refer to as *local update methods*, generalizing many federated and meta-learning algorithms. We prove that for quadratic models, local update methods are equivalent to first-order optimization on a surrogate loss we exactly characterize. Moreover, fundamental algorithmic choices (such as learning rates) explicitly govern a trade-off between the condition number of the surrogate loss and its alignment with the true loss. We derive novel convergence rates showcasing these trade-offs and highlight their importance in communication-limited settings. Using these insights, we are able to compare local update methods based on their convergence/accuracy trade-off, not just their convergence to critical points of the empirical loss. Our results shed new light on a broad range of phenomena, including the efficacy of server momentum in federated learning and the impact of proximal client updates.

1 Introduction

Federated learning (McMahan et al., 2017) is a distributed framework for learning models without directly sharing data. In this framework, clients perform *local updates* (typically using first-order optimization) on their own data. In the popular FedAvg algorithm (McMahan et al., 2017), the client models are then averaged at a central server. Since the proposal of FedAvg, many new federated optimization algorithms have been developed (Li et al., 2020a; Reddi et al., 2020; Hsu et al., 2019; Xie et al., 2019; Basu et al., 2019; Li et al., 2020b; Karimireddy et al., 2019). These methods typically employ multiple local client epochs in order to improve communication-efficiency. We defer to Kairouz et al. (2019) and Li et al. (2019) for more detailed summaries of federated learning.

Local updates have also been used extensively in meta-learning. The celebrated MAML algorithm (Finn et al., 2017) employs multiple local model updates on a set of tasks in order to learn a model that quickly adapt to new tasks. MAML has inspired a number of model-agnostic meta-learning methods that also employ first-order local updates (Balcan et al., 2019; Fallah et al., 2020a; Nichol et al., 2018; Zhou et al., 2019). There are strong connections between federated learning and meta-learning, despite differences in practical concerns. Formal connections between the two were shown by Khodak et al. (2019) and have since been explored in many other works (Jiang et al., 2019; Fallah et al., 2020b).

We refer to methods that utilize multiple local updates across clients (or in the language of meta-learning, tasks) as *local update methods* (see Section 2.1 for a formal characterization). In practice, local update methods frequently outperform "centralized" methods such as SGD (McMahan et al., 2017; Finn et al., 2017; Hard et al., 2018; Yang et al., 2018; Hard et al., 2020). However, the empirical benefits of local update methods are not fully explained by existing theoretical analyses. For example, Woodworth et al. (2020) show that FedAvg often obtains convergence rates comparable to or worse than those of mini-batch SGD.

We focus on two difficulties that arise when analyzing local update methods. First, analyses must account for *client drift* (Karimireddy et al., 2019). As clients perform local updates on heterogeneous datasets, their local models drift apart. This hinders convergence to globally optimal models, and makes theoretical analyses more challenging. Similar phenomena were examined by Li et al. (2020a); Malinovsky et al. (2020); Pathak and Wainwright (2020) and Fallah et al. (2020b), who show that various local update methods do not converge to critical points of

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

the empirical loss.

Second, local update methods are difficult to compare. Analyses of different methods may use different hyperparameters regimes, or make different assumptions. Even comparing seemingly similar methods can require significant theoretical insight (Karimireddy et al., 2019; Fallah et al., 2020b). Moreover, comparisons can be made in fundamentally different ways. One may wish to maximize the final accuracy, or minimize the number of communication rounds needed to attain a given accuracy. Thus, it is not even clear *how* local update methods should be compared.

Contributions In this work, we invert the conventional narrative that issues such as client drift harm convergence. Instead, we view such phenomena as improving convergence, but to sub-optimal points.

More generally, we show that local update methods face a fundamental trade-off between convergence and accuracy that is explicitly governed by algorithmic hyperparameters. Perceived failures of methods such as FedAvg actually correspond to operating points prioritizing convergence over accuracy. We use this trade-off to develop a novel framework for comparing local update methods. We compare methods based on their entire convergence-accuracy trade-off, not just their convergence to optimal points. In more detail:

1. We show that for quadratic models, local update methods are equivalent to optimizing a single *surrogate* loss function. The condition number of the surrogate is controlled by algorithmic choices. Popular local update methods, including FedAvg and MAML, reduce the surrogate's condition number, but increase the discrepancy between the empirical and surrogate losses. Our results also encompass *proximal* local update methods (Li et al., 2020a; Zhou et al., 2019).

2. We derive novel convergence rates that showcasing this trade-off between convergence and accuracy. Our bounds demonstrate the benefit of local update methods over methods such as mini-batch SGD in communication-limited settings.

3. We use this theory to develop a framework for comparing local update methods through a novel *Pareto frontier*, which compares convergence-accuracy trade-offs of classes of algorithms. We use this to derive novel comparisons of many popular local update methods.

4. We use this technique to shed light on a broad range of phenomena, including the benefit of server momentum, the effect of proximal local updates, and differences between the dynamics of FedAvg and MAML.

5. While our theoretical results are restricted to quadratic models, we show that such convergence-accuracy trade-offs occur empirically in non-convex settings. We also validate our theoretical observations regarding server momentum and proximal updates on a non-convex task.

We view our work as a step towards holistic understandings of local update methods. Using the aforementioned Pareto frontiers, we highlight a number of new phenomena and open problems. One particularly intriguing observation is that the convergence-accuracy trade-off for FedAvg with heavy-ball server momentum appears to be completely symmetric. For more details, see Section 5. Our proof techniques may be of independent interest. We derive a novel analog of the Bhatia-Davis inequality (Bhatia and Davis, 2000) for mean absolute deviations, and use this to understand the accuracy of local update methods.

Notation We let $\|\cdot\|$ denote the ℓ_2 norm for vectors and the spectral norm for matrices. For a symmetric positive semi-definite matrix A, we let $A^{1/2}$ denote its matrix square root. We let $\preceq$ be the *Loewner order* on positive semi-definite matrices. For a real symmetric matrix A, we let $\lambda_{\max}(A)$, $\lambda_{\min}(A)$ denote its largest and smallest eigenvalues, and let $\text{cond}(A)$ denote their ratio. In a slight abuse of notation, if f is a L-smooth, μ-strongly convex function, we say $\text{cond}(f) \leq L/\mu$.

Accuracy and Meta-Learning We study the accuracy of local update methods on the training population. However, meta-learning algorithms are designed to learn a model that adapts well to new tasks; The empirical loss is not necessarily indicative of the "post-adaptation" accuracy of such methods (Finn et al., 2017). Despite this our focus still yields novel insights into qualitative differences between the training dynamics of federated learning and meta-learning methods. Perhaps surprisingly, we show that in certain hyperparameter regions, these methods exhibit identical trade-offs between convergence and pre-adaptation accuracy (see Figures 4 and 5). While we believe our results can be adapted to post-adaptation accuracy via techniques developed by Fallah et al. (2020a), we leave the analysis to future work.

2 Problem Setup

Let $\mathcal{I}$ denote some collection of clients, and let $\mathcal{P}$ be a distribution over $\mathcal{I}$. For each $i \in \mathcal{I}$, there is an associated distribution $\mathcal{D}_i$ over the space $\mathcal{Z}$ of examples.

For any $z \in \mathcal{Z}$, we assume there is symmetric matrix $B_z \in \mathbb{R}^{d \times d}$ and vector $c_z \in \mathbb{R}^d$ such that the loss of a model $x \in \mathbb{R}^d$ at z is given by

$$f(x; z) := \tfrac{1}{2}\|B_z^{1/2}(x - c_z)\|^2. \quad (1)$$

For $i \in \mathcal{I}$, we define the client loss function f_i and the overall loss function f as follows:

$$f_i(x) := \mathop{\mathbb{E}}_{z \sim \mathcal{D}_i}[f(x; z)], \quad f(x) := \mathop{\mathbb{E}}_{i \sim \mathcal{P}}[f_i(x)]. \quad (2)$$

The joint distribution $(\mathcal{I}, \mathcal{Z})$ defines a distribution over $\mathcal{Z}$, recovering standard risk minimization, as well as distributed risk minimization in which $\mathcal{P}$ and all $\mathcal{D}_i$ are uniform over finite sets. For $i \in \mathcal{I}$, define:

$$A_i := \mathop{\mathbb{E}}_{z \sim \mathcal{D}_i}[B_z], \quad c_i := A_i^{-1} \mathop{\mathbb{E}}_{z \sim \mathcal{D}_i}[B_z c_z]. \quad (3)$$

We assume these expectations exist and are finite. One can show that up to some additive constant,

$$f_i(x) = \tfrac{1}{2}\|A_i^{1/2}(x - c_i)\|^2.$$

We make the following assumptions throughout.

Assumption 1. *There are $\mu, L > 0$ such that for all i, $\mu I \preceq A_i \preceq LI$.*

Assumption 2. *There is some $C > 0$ such that for all i, $\|c_i\| \leq C$.*

Assumption 1 bounds the Lipschitz and strong convexity parameters of the f_i, and holds if the matrices B_z satisfy bounded eigenvalue conditions. Assumption 2 states the c_i are bounded. Intuitively, local update methods provide larger benefit for smaller values of C, as the clients progress towards similar optima. While our analysis can be directly generalized to the case where the c_i are contained in a ball of radius C about $p \in \mathbb{R}^d$, we assume $p = 0$ for simplicity. Assumptions 1 and 2 can be relaxed to only hold in expectation, though this complicates the analysis.

2.1 Local Update Methods

We consider a class of algorithms we refer to as *local update* methods. In these methods, at each round t the server samples a set I_t of M clients (in the language of meta-learning, tasks) from $\mathcal{P}$, and broadcasts its model x_t to all clients in I_t. Each client $i \in I_t$ optimizes its loss function f_i (starting at x_t) by applying K iterations of mini-batch SGD with batch size B and client learning rate γ. As proposed by Li et al. (2020a) and Zhou et al. (2019), clients also add ℓ_2 regularization with parameter $\alpha \geq 0$ towards the broadcast model x_t.

The client sends a linear combination of the gradients it computes to the server. The coefficients of the linear combination are given by $\Theta = (\theta_1, \theta_2, \ldots, \theta_k, \ldots)$ for $\theta_i \in \mathbb{R}_{\geq 0}$, where Θ has finite and non-zero support. For such Θ, we define

$$K(\Theta) := \max\{k \mid \theta_k > 0\}, \quad w(\Theta) = \sum_{k=1}^{K(\Theta)} \theta_k. \quad (4)$$

After receiving all client updates, the server treats their average q_t as an estimate of the gradient of the loss function f, and applies q_t to a first-order optimization algorithm ServerOpt. For example, the server could perform a gradient descent step using the "pseudo-gradient" q_t. We refer to this process (parameterized by α, γ, Θ and ServerOpt) as LocalUpdate and give pseudo-code in Algorithms 1 and 2.

Algorithm 1 LocalUpdate: ServerUpdate

ServerUpdate$(x, \text{ServerOpt}, \alpha, \gamma, \Theta)$**:**
  $x_0 = x$
  for each round $t = 0, 1, \ldots, T-1$ **do**
    sample a set I_t of size M from $\mathcal{P}$
    for each client $i \in I_t$ **in parallel do**
      $q_t^i = \text{ClientUpdate}(i, x_t, \alpha, \gamma, \Theta)$
    $q_t = (1/M) \sum_{i \in I_t} q_t^i$
    $x_{t+1} = \text{ServerOpt}(x_t, q_t)$
  return x_{T+1}

Algorithm 2 LocalUpdate: ClientUpdate

ClientUpdate$(i, x, \alpha, \gamma, \Theta)$**:**
  $x_1 = x$
  for $k = 1, 2, \ldots, K(\Theta)$ **do**
    sample a set S_k of size B from $\mathcal{D}_i$
    $g_k = (1/B) \sum_{z \in S_k} \nabla_{x_k} \left(f_i(x_k; z) + \frac{\alpha}{2} \|x_k - x\|^2 \right)$
    $x_{k+1} = x_k - \gamma g_k$
  return $\sum_{k=1}^{K(\Theta)} \theta_k g_k$

LocalUpdate recovers many well-known algorithms for various choices Θ. For convenience, define

$$\Theta_K := (\underbrace{0, \ldots, 0}_{\text{K-1 times}}, 1), \quad \Theta_{1:K} := (\underbrace{1, \ldots, 1}_{\text{K times}}). \quad (5)$$

Special cases of LocalUpdate when ServerOpt is gradient descent are given in Table 1. For details on the relation between FedAvg and LocalUpdate, see Appendix A. By changing ServerOpt, we can recover methods such as FedAvgM (Hsu et al., 2019) (server gradient descent with momentum), and FedAdam (Reddi et al., 2020) (server Adam (Kingma and Ba, 2014)).

Table 1: Special cases of LocalUpdate when ServerOpt is gradient descent.

Algorithm	Θ	Conditions
Mini-batch SGD	Θ_1	$\lvert\mathcal{P}\rvert = 1, \alpha = 0, \gamma = 0$
Lookahead (Zhang et al., 2019)	$\Theta_{1:K}$	$\lvert\mathcal{P}\rvert = 1, \alpha = 0, \gamma > 0$
FedSGD (McMahan et al., 2017)	$\Theta_{1:K}$	$\alpha = 0, \gamma = 0$
FedAvg (McMahan et al., 2017), Reptile (Nichol et al., 2018)	$\Theta_{1:K}$	$\alpha = 0, \gamma > 0$
FedProx (Li et al., 2020a), MetaMinibatchProx (Zhou et al., 2019)	$\Theta_{1:K}$	$\alpha > 0, \gamma > 0$
FOMAML (Finn et al., 2017)	Θ_K	$\alpha = 0, \gamma > 0$
MAML (Finn et al., 2017)	Θ_{2K+1}	$\alpha = 0$, quadratics (Theorem 2)

3 Local Update Methods as First-Order Methods

LocalUpdate can vary drastically from first-order optimization methods on the empirical loss. Despite this, we will show that Algorithm 2 is equivalent in expectation to ServerOpt applied to a single *surrogate loss*. This surrogate loss is determined by the inputs α, γ and Θ to Algorithm 2. For each client $i \in \mathcal{I}$, we define its *distortion matrix* $Q_i(\alpha, \gamma, \Theta)$ as

$$Q_i(\alpha, \gamma, \Theta) := \sum_{k=1}^{K(\Theta)} \theta_k (I - \gamma(A_i + \alpha I))^{k-1}. \quad (6)$$

We define the surrogate loss function of client i as

$$\tilde{f}_i(x, \alpha, \gamma, \Theta) := \frac{1}{2} \|(Q_i(\alpha, \gamma, \Theta) A_i)^{1/2} (x - c_i)\|^2 \quad (7)$$

and the overall surrogate loss function as

$$\tilde{f}(x, \alpha, \gamma, \Theta) := \mathop{\mathbb{E}}_{i \sim \mathcal{P}} [\tilde{f}_i(x, \alpha, \gamma, \Theta)]. \quad (8)$$

When $\Theta = \Theta_1$, $Q_i(\alpha, \gamma, \Theta) = I$, in which case there is no distortion. In general, $Q_i(\alpha, \gamma, \Theta)$ can amplify the heterogeneity of the A_i. We derive the following theorem linking the surrogate losses to Algorithm 2.

Theorem 1. *For all $i \in \mathcal{P}$,*

$$\mathbb{E}[\text{ClientUpdate}(i, x, \alpha, \gamma, \Theta)] = \nabla \tilde{f}_i(x, \alpha, \gamma, \Theta).$$

For q_t as in Algorithm 1, Theorem 1 implies $\mathbb{E}[q_t] = \nabla \tilde{f}(x, \alpha, \gamma, \Theta)$. Thus, one round of LocalUpdate is equivalent in expectation to one step of ServerOpt on the surrogate loss $\tilde{f}(x, \alpha, \gamma, \Theta)$. As $\gamma \to 0$ or $K \to 1$, $\tilde{f}(x, \alpha, \gamma, \Theta) \to w(\Theta) f(x)$, so as γ gets smaller, the "pseudo-gradients" q_t more closely resemble stochastic gradients of the empirical loss function.

A version of Theorem 1 was shown for $\alpha = 0, \Theta = \Theta_2$ by Fallah et al. (2020b). We take this a step further and show that in certain settings, MAML is equivalent in expectation to ServerOpt on a surrogate loss.

MAML MAML with K local steps can be viewed as a modification of LocalUpdate. Algorithm 1 remains the same, and in Algorithm 2, each client executes K mini-batch SGD steps. However, the client's message to the server is different. Let $X_K^i(x)$ be the function that runs K steps of mini-batch SGD, starting from x, for fixed mini-batches $S_1, \ldots, S_K$ of size B drawn independently from $\mathcal{D}_i$. Define

$$m_K^i(x; z) = f(X_K^i(x); z), \; m_K^i(x) = \mathop{\mathbb{E}}_{z \sim \mathcal{D}_i} [m_K^i(x; z)].$$

Each client i sends a stochastic estimate of $\nabla m_K^i(x)$ to the server. The rest is identical to LocalUpdate; The server averages the client outputs and uses this as a gradient estimate for ServerOpt. While MAML is not a special case of LocalUpdate, we show that if the clients use gradient descent, MAML is equivalent in expectation to LocalUpdate with $\Theta = \Theta_{2K+1}$.

Theorem 2. *If $X_K^i(x)$ is the function that runs K steps of gradient descent on $\mathcal{D}_i$ with learning rate γ starting at x, then*

$$\nabla m_K^i(x) = \nabla_x \tilde{f}_i(x, 0, \gamma, \Theta_{2K+1}).$$

An analogous result holds if the clients perform proximal updates ($\alpha > 0$ in ClientUpdate). Thus, to understand LocalUpdate and MAML on quadratic models, it suffices to analyze the optimization dynamics of $\tilde{f}(x, \alpha, \gamma, \Theta)$. We use this viewpoint to study the convergence and accuracy of these methods.

4 Convergence and Accuracy of Local Update Methods

Comparing (2) and (8), we see that $\tilde{f}(x, \alpha, \gamma, \Theta)$ and $f(x)$ need not share critical points. Special cases of this fact were noted by Malinovsky et al. (2020), Fallah et al. (2020b), and Pathak and Wainwright (2020). We will show that this is not a failure of local update methods. Rather, by altering the loss function being optimized, LocalUpdate can greatly improve convergence, but to a less accurate point. More generally, the choice of α, γ and Θ dictates a trade-off between

convergence and accuracy. Intuitively, the larger γ and $K(\Theta)$ are, the faster LocalUpdate will converge, and the less accurate the resulting model may be.

To show this formally, we restrict to $Q_i(\alpha, \gamma, \Theta) \succ 0$, as then $\tilde{f}(x, \gamma, \Theta)$ is strongly convex with a unique minimizer. This is ensured by the following.

Lemma 1. *Suppose that $\gamma < (L + \alpha)^{-1}$. Then for all i, $Q_i(\alpha, \gamma, \Theta)$ is positive definite and $\tilde{f}(x, \alpha, \gamma, \Theta)$ is strongly convex.*

4.1 Condition Numbers

Under the conditions of Lemma 1, $\tilde{f}_i(x, \alpha, \gamma, \Theta)$ has a well-defined condition number which we bound.

Lemma 2. *Suppose $\gamma < (L + \alpha)^{-1}$. Define*

$$\kappa(\alpha, \gamma, \Theta) := \frac{\mathbb{E}_i[\lambda_{\max}(Q_i(\alpha, \gamma, \Theta)A_i]}{\mathbb{E}_i[\lambda_{\min}(Q_i(\alpha, \gamma, \Theta)A_i]}. \tag{9}$$

Then $\text{cond}(\tilde{f}) \leq \kappa(\alpha, \gamma, \Theta)$.

We wish to better understand (9) in cases of interest. We first consider $\Theta = \Theta_{1:K}$, as in FedAvg. Define

$$\phi(\lambda, \alpha, \gamma, K) := \sum_{k=1}^{K} (1 - \gamma(\lambda + \alpha))^{k-1}\lambda. \tag{10}$$

We now derive a bound on $\text{cond}(\tilde{f})$ for $\Theta = \Theta_{1:K}$.

Lemma 3. *If $\gamma < (L + \alpha)^{-1}$, $\tilde{f}(x, \alpha, \gamma, \Theta_{1:K})$ is $\phi(L, \alpha, \gamma, K)$-smooth, $\phi(\mu, \alpha, \gamma, K)$-strongly convex, and $\text{cond}(\tilde{f}) \leq \kappa(\alpha, \gamma, \Theta_K)$ where*

$$\kappa(\alpha, \gamma, \Theta_{1:K}) \leq \frac{\phi(L, \alpha, \gamma, K)}{\phi(\mu, \alpha, \gamma, K)}. \tag{11}$$

When $\gamma = 0, \alpha = 0$, we recover the condition number L/μ of the empirical loss f. We next consider $\Theta = \Theta_K$, as in MAML-style algorithms. Define

$$\psi(\lambda, \alpha, \gamma, K) := (1 - \gamma(\lambda + \alpha))^{K-1}\lambda. \tag{12}$$

We now derive a bound on $\text{cond}(\tilde{f})$ for $\Theta = \Theta_K$.

Lemma 4. *If $\gamma < (KL + \alpha)^{-1}$, $\tilde{f}(x, \alpha, \gamma, \Theta_K)$ is $\psi(L, \alpha, \gamma, K)$-smooth, $\psi(\mu, \alpha, \gamma, K)$-strongly convex, and $\text{cond}(\tilde{f}) \leq \kappa(\alpha, \gamma, \Theta_K)$ where*

$$\kappa(\alpha, \gamma, \Theta_K) \leq \left(\frac{1 - \gamma(L + \alpha)}{1 - \gamma(\mu + \alpha)}\right)^{K-1} \frac{L}{\mu}. \tag{13}$$

We show in Appendix B.3 that Lemmas 3 and 4 are tight. The extra condition that $\gamma < (KL + \alpha)^{-1}$ for $\Theta = \Theta_K$ is due to the fact that when $\gamma \geq (KL+\alpha)^{-1}$, $\text{cond}(\tilde{f}(x, \alpha, \gamma, \Theta_K))$ depends on intermediate eigenvalues of the A_i, and exhibits more nuanced behavior. We explore this further in Section 5.

As $K \to 1$ or $\gamma \to 0$, $\kappa(\alpha, \gamma, \Theta_K) \to L/\mu$, which bounds the condition number of the empirical loss f. If γ is not close to 0, we get an exponential reduction (in terms of K) of the condition number. While the analysis is not as clear for $\Theta_{1:K}$, one can show that $\kappa(\alpha, \gamma, \Theta_{1:K}) \leq L/\mu$, with equality if and only if $\alpha = 0$, and either $\gamma = 0$ or $K = 1$. Moreover, $\kappa(\alpha, \gamma, \Theta_{1:K})$ decreases as $K \to \infty$ or $\gamma \to (L+\alpha)^{-1}$. For both $\Theta_{1:K}$ and Θ_K, increasing α decreases κ.

Here we see the impact of local update methods on convergence: Popular methods such as FedAvg, FedProx, MAML, and Reptile reduce the condition number of the surrogate loss function they are actually optimizing. In the next section, we translate this into concrete convergence rates for LocalUpdate.

4.2 Convergence Rates

We now focus on a *deterministic* version of LocalUpdate in which all clients participate at each round and perform K steps of gradient descent. The server updates its model using $q_t = \mathbb{E}_{i\sim\mathcal{P}}[q_t^i]$, where q_t^i is the output of ClientUpdate for client i. In particular, $\mathcal{P}$ must be known to the server. In this case, Theorem 1 implies that $q_t = \nabla\tilde{f}(x_t, \alpha, \gamma, \Theta)$, so LocalUpdate is equivalent to applying ServerOpt to the true gradients of $\tilde{f}(x, \alpha, \gamma, \Theta)$.

We specialize to the setting where ServerOpt is gradient descent, with or without momentum (though our analysis can be directly extended to other optimizers). Thus, LocalUpdate is equivalent to gradient descent on $\tilde{f}(x, \alpha, \gamma, \Theta)$. Using the bound on $\text{cond}(\tilde{f})$ in Lemma 2, we can directly apply classical convergence theory gradient descent (Lessard et al. (2016, Proposition 1) give a useful summary) to derive convergence rates for LocalUpdate. Similar analyses can be done in the stochastic setting.

Theorem 3. *Suppose $\gamma < (L+\alpha)^{-1}$ and ServerOpt is gradient descent with Nesterov, heavy-ball, or no momentum. Then for some hyperparameter setting of ServerOpt, and ρ as in Table 2, the iterates $\{x_t\}_{t\geq 1}$ of LocalUpdate satisfy*

$$\|x_T - x^*(\alpha, \gamma, \Theta)\| \leq \rho^T \|x_0 - x^*(\alpha, \gamma, \Theta)\|. \tag{14}$$

Thus, (properly tuned) server momentum improves the convergence of LocalUpdate, giving theoretical grounding[1] to the improved convergence of FedAvgM shown by Hsu et al. (2019) and Reddi et al. (2020). Since ServerOpt does not change the surrogate loss,

[1] Yuan and Ma (2020) first showed that momentum can accelerate FedAvg, though they use a different momentum scheme with extra per-round communication.

Table 2: Convergence rates of LocalUpdate when ServerOpt is gradient descent (with or without momentum), and $\kappa = \kappa(\alpha, \gamma, \Theta)$ is as in (9).

Momentum	Rate
None	$\rho = \frac{\kappa-1}{\kappa+1}$
Nesterov	$\rho = 1 - \frac{2}{\sqrt{3\kappa+1}}$
Heavy-ball	$\rho = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$

this improvement in convergence does not degrade the accuracy of the learned model.

Given the bounds on $\text{cond}(\tilde{f})$ in (11) and (13), we obtain explicit convergence rates for FedAvg- and MAML-style algorithms as well. In particular, one can show that increasing γ or K decreases κ (and therefore ρ). We show in the next section that this comes at the expense of increasing the empirical loss.

4.3 Distance Between Global Minimizers

We now turn our attention towards the discrepancy between the surrogate loss $\tilde{f}$ and the empirical loss f. We assume $\gamma < (L+\alpha)^{-1}$. By Lemma 1, $\tilde{f}$ and f are strongly convex with global minimizers we denote by

$$x^*(\alpha, \gamma, \Theta) := \operatorname*{argmin}_x \tilde{f}(x, \alpha, \gamma, \Theta),$$

$$x^* := \operatorname*{argmin}_x f(x).$$

We are interested in $\|x^*(\alpha, \gamma, \Theta) - x^*\|$. While we focus on the setting where $\mathcal{P}$ is a discrete distribution over some finite $\mathcal{I}$, our analysis can be generalized to arbitrary probability spaces $(\mathcal{I}, \mathcal{F}, \mathcal{P})$. We derive the following bound.

Lemma 5. *Let* $b = \max_{i\in\mathcal{I}} \lambda_{\max}(Q_i(\alpha, \gamma, \Theta))$ *and* $a = \min_{i\in\mathcal{I}} \lambda_{\min}(Q_i(\alpha, \gamma, \Theta))$. *Then*

$$\|x^*(\alpha, \gamma, \Theta) - x^*\| \leq 8C \frac{\sqrt{b} - \sqrt{a}}{\sqrt{b} + \sqrt{a}}. \tag{15}$$

When $d = 1$, we can reduce the constant factor to $2C$, which we show is tight (see Appendix C.2). While we conjecture that this bound holds with a constant of $2C$ for all d, we leave this to future work.

Our proof technique for Lemma 5 may be of independent interest. We derive this result by first proving an analog of the Bhatia-Davis inequality (Bhatia and Davis, 2000) for mean absolute deviations of bounded random variables (Theorem 5 in Appendix C).

Let $\kappa_0 := L/\mu$. Specializing to $\Theta = \Theta_{1:K}$ or Θ_K, we derive a link between $\kappa(\alpha, \gamma, \Theta)$ in (11) and (13) and the distance between optimizers.

Lemma 6. *Suppose that either (I)* $\gamma < (L+\alpha)^{-1}$ *and* $\Theta = \Theta_{1:K}$ *or (II)* $\gamma < (KL+\alpha)^{-1}$ *and* $\Theta = \Theta_K$. *Then for all* $i \in \mathcal{P}$, $\text{cond}(Q_i(\alpha, \gamma, \Theta)) \leq \kappa_0 \kappa(\alpha, \gamma, \Theta)^{-1}$.

Combining this with Lemma 5, we get:

Theorem 4. *Under the same settings as Lemma 6,*

$$\|x^*(\alpha, \gamma, \Theta) - x^*\| \leq 8C \frac{\sqrt{\kappa_0} - \sqrt{\kappa(\alpha, \gamma, \Theta)}}{\sqrt{\kappa_0} + \sqrt{\kappa(\alpha, \gamma, \Theta)}}. \tag{16}$$

Applying Theorem 3, we bound the convergence of LocalUpdate to the empirical minimizer x^*.

Corollary 1. *Under the same settings as Theorem 3, for some hyperparameter setting of* ServerOpt, *the iterates* $\{x_t\}_{t\geq 1}$ *of* LocalUpdate *satisfy*

$$\|x_T - x^*\| \leq \rho^T \|x_0 - x^*(\alpha, \gamma, \Theta)\| + 8C \frac{\sqrt{\kappa_0} - \sqrt{\kappa}}{\sqrt{\kappa_0} + \sqrt{\kappa}}$$

where $\kappa = \kappa(\alpha, \gamma, \Theta)$ *is given in* (11) *and* (13), *and* ρ *is given in Table 2.*

Here we see the benefit of local update methods in *communication-limited* settings. When T is small and $\|x_0 - x^*\|$ is large, we can achieve better convergence by decreasing ρ and leaving the second term fixed. In such settings, FedAvg can arrive at a neighborhood of a critical point in fewer communication rounds than mini-batch SGD, but may not ever actually reach the critical point. If $\|x_0 - x^*\|$ is small, we may be better served by using mini-batch SGD instead.

5 Comparing Local Update Methods

Comparing optimization algorithms is a fundamental theoretical effort. Many past works compare local update methods based on their convergence to critical points of the empirical loss. By Theorem 1, LocalUpdate is only guaranteed to converge to critical points of f if $\gamma = 0$ or $K(\Theta) = 1$. Thus, existing analyses ignore many useful cases of LocalUpdate.

To remedy this, we compare local update algorithms on the basis of both convergence and accuracy. Instead of fixing γ and Θ, we analyze LocalUpdate as γ and $K(\Theta)$ vary. To do so, we use our theory from Section 4. Given α, γ and Θ, we define the **convergence rate** $\rho(\alpha, \gamma, \Theta)$ as the infimum over all ρ such that for all $T \geq 1$, (14) holds. Values of ρ when ServerOpt is gradient descent are given in Table 2. For $\Theta = \Theta_{1:K}$ or Θ_K, we define the **suboptimality** $\Delta(\alpha, \gamma, \Theta)$ by

$$\Delta(\alpha, \gamma, \Theta) := \frac{\sqrt{\kappa_0} - \sqrt{\kappa(\alpha, \gamma, \Theta)}}{\sqrt{\kappa_0} + \sqrt{\kappa(\alpha, \gamma, \Theta}}. \tag{17}$$

By Theorem 4, this captures the asymptotic worst-case suboptimality of LocalUpdate.

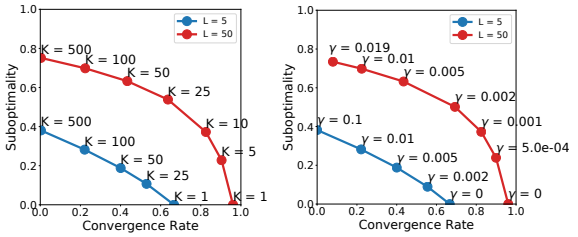


Figure 1: Pareto frontiers for $\mu = 1, \alpha = 0, \Theta = \Theta_{1:K}$ and $L \in \{5, 50\}$. We fix $\gamma = 0.01$ and vary K (left), and fix $K = 100$ and vary γ (right).

Note that $\rho, \Delta \in [0, 1]$. Therefore, fixing μ, L and ServerOpt, we obtain a **Pareto frontier** in $[0, 1]^2$ by plotting (ρ, Δ) for various γ and $K(\Theta)$. This curve represents the worst-case convergence/accuracy trade-off of a class of local update methods. We generally want the curve to be as close to $(0, 0)$ as possible.

For example, in Figure 1 we let ServerOpt be gradient descent and set $\alpha = 0, \Theta = \Theta_{1:K}$. We plot (ρ, Δ) as we vary K and fix γ, and vice-versa. When $L = 5$, we obtain nearly identical curves. The curves for $L = 50$ are similar, except that when we fix K and vary γ, we do not reach $\rho \approx 0$. While γ and K have similar impacts on convergence-accuracy trade-offs, varying K leads a larger set of attainable (ρ, Δ). Formally, this is because in (11), $\lim_{\gamma \to L^{-1}}(\kappa) \neq 0$. Intuitively, $K \to \infty$ recovers one-shot averaging while $\gamma \to L^{-1}$ does not. Notably, the convergence-accuracy trade-off becomes closer to a linear trade-off as L/μ decreases.

The Pareto frontiers contain more information than just the convergence rate to a critical point (the curve's intersection with the x-axis). This information is useful in communication-limited regimes, where we wish to minimize the number of rounds needed to attain a given accuracy. The curves also help visualize various hyperparameter settings of an algorithms simultaneously. To illustrate this, we use the Pareto frontiers to derive novel findings regarding server momentum, proximal client updates, and qualitative differences between FedAvg and MAML. The results are all given below. For more results, see Appendix D.

Impact of Server Momentum As shown empirically by Hsu et al. (2019) and as reflected in Table 2, server momentum can improve convergence. To understand this, in Figure 2 we compare Pareto frontiers where $\Theta = \Theta_{1:K}$ and ServerOpt is gradient descent with various types of momentum (Nesterov, heavy-ball, or no momentum). We see a strict ordering of the server optimization methods. Heavy-ball momentum is better than Nesterov momentum, which is better than no momentum.

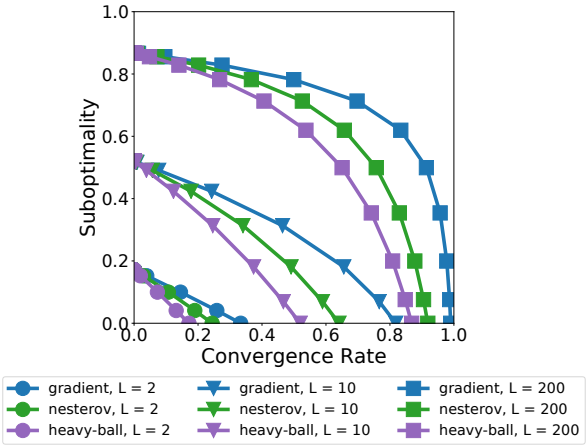


Figure 2: Pareto frontiers for $\mu = 1$, varying L, and where ServerOpt is gradient descent with different types of momentum. We let $\alpha = 0, \gamma = (2L)^{-1}$ and $\Theta = \Theta_{1:K}$ for varying $K \in [1, 10^6]$.

One important finding is that the benefit of momentum is more pronounced as L/μ increases. On the other hand, the benefit of server momentum diminishes for sufficiently large K: In Figure 4, the various types of momentum lead to similar suboptimality when the convergence rate is close to 0. Intuitively, as $K \to \infty$, we recover one-shot averaging, which converges in a single communication round with or without momentum.

Another intriguing observation: The Pareto frontiers for heavy-ball momentum appear to be symmetric about the line $\rho = \Delta$. We conjecture this is true for any μ, L. While we believe that this may be provable by careful algebraic manipulation of our results above, ideally a proof would explain the root causes of this symmetry. Thus, we leave a proof to future work.

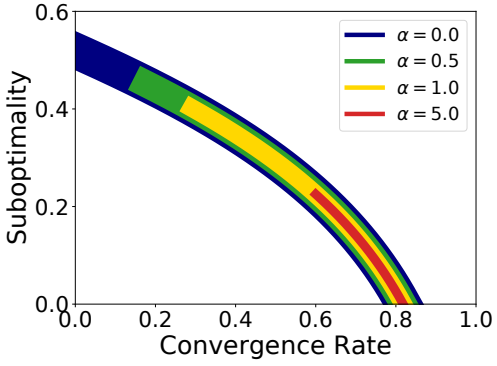


Figure 3: Pareto frontiers for $\mu = 1, L = 10, \Theta = \Theta_{1:K}$. We set $\gamma = 1/2(L + \alpha)^{-1}$, ServerOpt as gradient descent, vary α and K over $\{0, 0.5, 1.0, 5.0\}$ and $[1, 10^6]$.

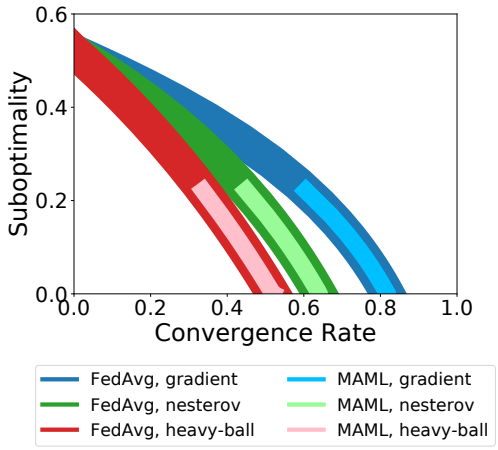


Figure 4: Pareto frontiers for $\mu = 1$, $L = 10$, $\alpha = 0$, $\Theta = \Theta_{1:K}$ (FedAvg) and Θ_K (MAML) for varying $K \in [0, 10^6]$. ServerOpt is gradient descent, with Nesterov, heavy-ball, or no momentum. For $\Theta_{1:K}$, we use $\gamma = 0.001$, and for Θ_K, we use $\gamma = (2LK)^{-1}$.

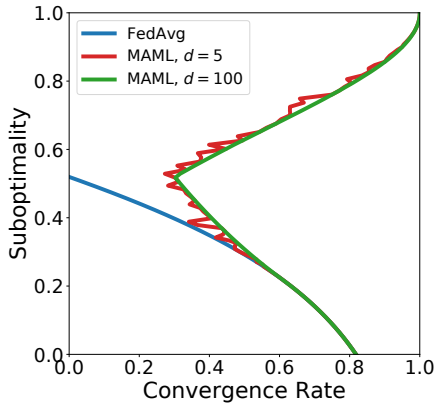


Figure 5: Simulated Pareto frontiers for $\mu = 1$, $L = 10$, $\alpha = 0$, $\gamma = 0.001$, $\Theta = \Theta_K$ (MAML). ServerOpt is gradient descent. We randomly sample $A \in \mathbb{R}^{d \times d}$ with $\mu \preceq A \preceq L$, and compute (ρ, Δ) for various $K \in [1, 10^6]$ and $d \in \{5, 100\}$. We compare to the Pareto frontier for $\Theta_{1:K}$ (FedAvg).

Proximal Client Updates So far we have only considered $\alpha = 0$. One might posit that as α varies, the Pareto frontier moves closer to the origin. This appears to not be the case. In all settings we examined, changing α did not bring the Pareto frontier closer to 0. Instead, the frontier for $\alpha > 0$ was simply a subset of the frontier for $\alpha = 0$.

To illustrate this, we plot Pareto frontiers for varying α in Figure 3. As α increases, the frontier becomes a smaller subset of the frontier for $\alpha = 0$. Thus, proximal client updates may not enable faster convergence. Rather, their benefit may be in guarding against setting γ too small or K too large. Figure 3 shows that FedAvg can always attain the same (ρ, Δ) as FedProx, but it may require different hyperparameters. The reverse is not true, as FedProx cannot recover one-shot averaging. Our findings are consistent with work by Wang et al. (2020), who show that FedProx can reduce the "objective inconsistency" of FedAvg, at the expense of increasing convergence time.

Comparing MAML to FedAvg We now turn our attention to comparing FedAvg-style algorithms ($\Theta = \Theta_{1:K}$) to MAML-style algorithms ($\Theta = \Theta_K$). We plot Pareto frontiers for the ρ, Δ guaranteed by Theorems 3 and 4. The results are in Figure 4.

For each ServerOpt, the MAML frontier is a subset of the FedAvg frontier. Recall that in Theorem 3, we require $\gamma < (L + \alpha)^{-1}$ for FedAvg, but $\gamma < (KL + \alpha)^{-1}$ for MAML. In Figure 4 this causes the frontier for Θ_K to be more restrictive than for $\Theta_{1:K}$. However, it is still notable that these two fundamentally different methods, attain the same frontier when ρ is large.

By Lemma 1, ρ and Δ are still well-defined for Θ_K when $(KL + \alpha)^{-1} \leq \gamma < (L + \alpha)^{-1}$. To understand what happens in this regime, we generate random symmetric $A \in \mathbb{R}^{d \times d}$ satisfying $\text{cond}(A) = L/\mu$ and compute $Q(\alpha, \gamma, \Theta_K)$ as in (6). We then compute κ via (9), and plug this into Table 2 and (17) to get ρ, Δ. This gives us a simulated Pareto frontier for Θ_K, which we compare to $\Theta_{1:K}$ in Figure 5. For details and additional experiments, see Appendix D.1.

The Pareto frontiers are identical for small K (mirroring Figure 4), but diverge when $\gamma \geq (KL + \alpha)^{-1}$. The frontier for MAML then moves further from 0. Intuitively, FedAvg tries to learn a global model, while MAML tries to learn a model that adapts quickly to new tasks (Finn et al., 2017); MAML need not minimize (ρ, Δ). The MAML frontier is noisy for $d = 5$ (as ρ, Δ depend on random eigenvalues of A), but stabilizes for $d = 100$. While we posit that this reflects a semi-circle law for eigenvalues of random matrices (Alon et al., 2002), we leave an analysis to future work.

One final observation that highlights the similarities and differences of FedAvg- and MAML-style methods: In Figure 4, the curve for MAML when $d = 100$ has a clear cusp. This seems to occur at the same suboptimality (ie. y-value) as the intersection of the FedAvg curve with the y-axis. In other words, the behavior of MAML diverges substantially from FedAvg, but only after it reaches the same suboptimality as FedAvg for $K \to \infty$ (which corresponds to one-shot averaging). We are unsure why the suboptimality of one-shot averaging corresponds to a cuspidal operating point of MAML, but this observation highlights significant nuance in the behavior of these methods.

6 Limitations and Discussion

Our convergence-accuracy framework and the resulting Pareto frontiers can be useful tools in understanding how algorithmic choices impact local update methods. The obvious limitation is that they only apply to quadratic models. While this is restrictive, we show empirically in Appendix F that even for non-convex functions, the client learning rate governs a convergence-accuracy trade-off for FedAvg.

Our framework may also be useful in identifying important phenomena underlying LocalUpdate, even in non-quadratic settings. To demonstrate this, we show that many of the observations in Section 5 hold in non-convex settings. We train a CNN on the FEMNIST dataset (Caldas et al., 2018) using LocalUpdate where $\Theta = \Theta_{1:50}$. We tune client and server learning rates. See Appendix E for full details. In Figure 6, we illustrate how server momentum and α change convergence. Our results match the Pareto frontiers in Figures 2 and 3: Server momentum improves convergence, while α has little to no effect, provided we tune learning rates.

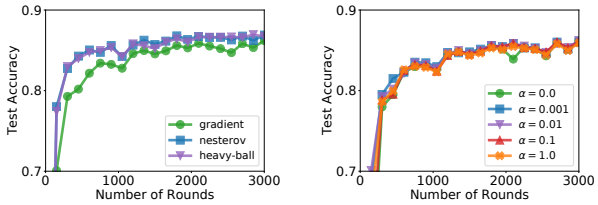


Figure 6: Test accuracy of LocalUpdate with $\Theta = \Theta_{1:50}$ on FEMNIST with tuned learning rates. (Left) Varying types of server momentum, $\alpha = 0$. (Right) No momentum and varying α.

This brief example illustrates that our framework can identify crucial facets of local update methods. While our framework may not capture all relevant details of such methods, we believe it greatly simplifies their analysis, comparison, and design. In the future, we hope to extend this framework to more general loss functions. Other important extensions include stochastic settings with partial client participation, as well as trade-offs between convergence and post-adaptation accuracy of local update methods.

References

Noga Alon, Michael Krivelevich, and Van H Vu. On the concentration of eigenvalues of random symmetric matrices. *Israel Journal of Mathematics*, 131(1): 259–267, 2002.

Maria-Florina Balcan, Mikhail Khodak, and Ameet Talwalkar. Provable guarantees for gradient-based meta-learning. In *International Conference on Machine Learning*, pages 424–433. PMLR, 2019.

Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations. In *Advances in Neural Information Processing Systems*, pages 14668–14679, 2019.

Rajendra Bhatia and Chandler Davis. A better bound on the variance. *The American Mathematical Monthly*, 107(4):353–357, 2000.

Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečný, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1082–1092. PMLR, 26–28 Aug 2020a. URL http://proceedings.mlr.press/v108/fallah20a.html.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020b.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR, 2017.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

Andrew Hard, Kurt Partridge, Cameron Nguyen, Niranjan Subrahmanya, Aishanee Shah, Pai Zhu, Ignacio Lopez Moreno, and Rajiv Mathews. Training keyword spotting models on non-IID data with federated learning. *arXiv preprint arXiv:2005.10406*, 2020.

Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip B Gibbons. The non-IID data quagmire of decentralized machine learning. *arXiv preprint arXiv:1910.00189*, 2019.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Alex Ingerman and Krzys Ostrowski. Introducing TensorFlow Federated, 2019. URL https://medium.com/tensorflow/introducing-tensorflow-federated-a4147aa20041.

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

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

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for on-device federated learning. *arXiv preprint arXiv:1910.06378*, 2019.

Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. Adaptive gradient-based meta-learning methods. In *Advances in Neural Information Processing Systems*, pages 5915–5926, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *arXiv preprint arXiv:1908.07873*, 2019.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems 2020*, pages 429–450, 2020a.

Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. Fair resource allocation in federated learning. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=ByexElSYDr.

Grigory Malinovsky, Dmitry Kovalev, Elnur Gasanov, Laurent Condat, and Peter Richtarik. From local SGD to local fixed point methods for federated learning. *arXiv preprint arXiv:2004.01442*, 2020.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, pages 1273–1282, 2017.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Reese Pathak and Martin J Wainwright. FedSplit: An algorithmic framework for fast federated optimization. *arXiv preprint arXiv:2005.05238*, 2020.

Tiberiu Popoviciu. Sur les équations algébriques ayant toutes leurs racines réelles. *Mathematica*, 9:129–145, 1935.

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

Sebastian U. Stich. Local SGD converges fast and communicates little. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1g2JnRcFX.

Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *arXiv preprint arXiv:2007.07481*, 2020.

Blake Woodworth, Kumar Kshitij Patel, Sebastian U Stich, Zhen Dai, Brian Bullins, H Brendan McMahan, Ohad Shamir, and Nathan Srebro. Is local SGD better than minibatch SGD? *arXiv preprint arXiv:2002.07839*, 2020.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

Cong Xie, Oluwasanmi Koyejo, Indranil Gupta, and Haibin Lin. Local AdaAlter: Communication-efficient stochastic gradient descent with adaptive learning rates. *arXiv preprint arXiv:1911.09030*, 2019.

Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

Honglin Yuan and Tengyu Ma. Federated accelerated stochastic gradient descent. *Advances in Neural Information Processing Systems*, 33, 2020.

Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps for-

ward, 1 step back. In *Advances in Neural Information Processing Systems*, pages 9593–9604, 2019.

Pan Zhou, Xiaotong Yuan, Huan Xu, Shuicheng Yan, and Jiashi Feng. Efficient meta learning via minibatch proximal update. In *Advances in Neural Information Processing Systems*, pages 1534–1544, 2019.

Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.

A Relations between FedAvg, FedProx, and LocalUpdate

We focus on the following (simplified) version of FedAvg, otherwise known as Local SGD (Zinkevich et al., 2010; Stich, 2019): At each iteration t, we sample some set of clients I_t of size M from the client population $\mathcal{I}$. Each client $i \in I_t$ receives the server's model x_t, and applies K steps of mini-batch SGD to its local model, resulting in an updated local model x_t^i. The server receives these models from the sampled clients, and updates its model via

$$x_{t+1} = \frac{1}{M} \sum_{i \in I_t} x_t^i.$$

Fix t, and let g_k^i denote the k-th mini-batch gradient of client i. Suppose we use a learning rate of γ on each client when performing mini-batch SGD. Then we have

$$\begin{aligned} x_{t+1} &= \frac{1}{M} \sum_{i \in I_t} x_t^i \\ &= x_t - \frac{1}{M} \sum_{i \in I_t} \left(x_t - x_t^i \right) \\ &= x_t - \frac{1}{M} \sum_{i \in I_t} x_t - \left(x_t - \gamma \sum_{k=1}^{K} g_k^i \right) \\ &= x_t - \gamma \frac{1}{M} \sum_{i \in I_t} \sum_{k=1}^{K} g_k^i. \end{aligned}$$

A similar analysis holds for FedProx, but with the usage of a proximal term with parameter $\alpha > 0$. In both cases, this is exactly LocalUpdate (see Algorithms 1 and 2) with $\Theta = \Theta_{1:K}$, $\eta = \gamma$, and where ServerOpt is gradient descent. However, by instead using a server learning rate of η that is allowed to vary independently of γ in LocalUpdate, we can obtain markedly different convergence behavior. We note that a form of this decoupling has previously been explored by Karimireddy et al. (2019) and Reddi et al. (2020). However, these versions instead perform averaging on the so-called "model delta", in which the server model is updated via

$$\begin{aligned} x_{t+1} &= x_t - \frac{\eta}{M} \sum_{i \in I_t} (x_t - x_t^i) \\ &= x_t - \frac{\eta}{M} \sum_{i \in I_t} \left(x_t - \left(x_t - \gamma \sum_{k=1}^{K} g_k^i \right) \right) \\ &= x_t - \frac{\eta\gamma}{M} \sum_{i \in I_t} \sum_{k=1}^{K} g_k^i. \end{aligned}$$

Thus, while this does decouple η and γ to some degree, it does not fully do so. In particular, if we set $\gamma = 0$, then $x_{t+1} = x_t$, in which case we can make no progress overall. This is particularly important because, as implied by Theorem 1, for $K(\Theta) > 1$ we can only guarantee that the surrogate loss has the same critical points as the true loss by setting $\gamma = 0$. More generally, we see that the effective learning rate used in the model-delta approach is the product $\eta\gamma$. This can result in conflations between the effect of changing the server learning rate γ and changing the client learning rate η. By disentangling these, we can better understand differences in the impact of these parameters on the underlying optimization dynamics.

B Omitted Proofs

Notation For a matrix $A \in \mathbb{R}^{m \times n}$ we let $A^\intercal \in \mathbb{R}^{n \times m}$ denote its transpose. Similarly, given a vector $v \in \mathbb{R}^{d \times 1}$, we let $v^\intercal \in \mathbb{R}^{1 \times d}$ denote its transpose. We let $\|\cdot\|$ denote the ℓ_2 norm for vectors, and the spectral norm for matrices. For a real, symmetric matrix A, we let $\lambda_{\max}(A), \lambda_{\min}(A)$ denote the maximum and minimum eigenvalue of a matrix. We let $\preceq$ denote the Loewner order on symmetric positive semi-definite matrices.

B.1 Proofs of Theorems 1 and 2

We first prove a general result about stochastic local updates on quadratic functions. Let $A \in \mathbb{R}^{d \times d}$ be symmetric and positive definite, and let $c \in \mathbb{R}^d$. Let

$$h(x) = \frac{1}{2} \left\| A^{1/2}(x - c) \right\|^2.$$

Suppose we perform K iterations of SGD on h with learning rate γ. That is, starting at x_1 we generate a sequence of independent random vectors $\{g_k\}_{k=1}^K$ and corresponding SGD iterates $\{x_k\}_{k=1}^{K+1}$ satisfying, for $1 \leq k \leq K$,

$$\mathbb{E}[g_k] = \nabla h(x_k), \tag{18}$$

$$x_{k+1} = x_k - \gamma g_k. \tag{19}$$

We then have the following lemma regarding the g_k.

Lemma 7. *For all $k \geq 1$,*

$$\mathbb{E}[g_{k+1}] = (I - \gamma A)\, \mathbb{E}[g_k]. \tag{20}$$

In particular, this implies

$$\mathbb{E}[g_k] = (I - \gamma A)^{k-1} A(x_1 - c). \tag{21}$$

Proof. By (18), the law of total expectation, and the independence of the g_k,

$$\mathbb{E}[g_k] = \mathbb{E}\left[\mathbb{E}[\nabla h(x_k) \mid g_1, \ldots, g_{k-1}]\right] = \mathbb{E}[A(x_k - c)].$$

By linearity of expectation,

$$\mathbb{E}[g_k] = A(\mathbb{E}[x_k] - c). \tag{22}$$

Combining (19) and (22), we have

$$\begin{aligned}
\mathbb{E}[g_{k+1}] &= A(\mathbb{E}[x_{k+1}] - c) \\
&= A(\mathbb{E}[x_k] - \gamma\, \mathbb{E}[g_k] - c) \\
&= A(A^{-1}\, \mathbb{E}[g_k] - \gamma\, \mathbb{E}[g_k]) \\
&= (I - \gamma A)\, \mathbb{E}[g_k]
\end{aligned}$$

This completes the first part of the proof. The second follows from noting that $\mathbb{E}[g_1] = A(x_1 - c)$. □

Recall that for $i \in \mathcal{I}$, $\alpha \geq 0, \gamma \geq 0$ and $\Theta = (\theta_1, \ldots, \theta_{K(\Theta)})$ we define the matrix $Q_i(\alpha, \gamma, \Theta)$ by

$$Q_i(\alpha, \gamma, \Theta) := \sum_{k=1}^{K(\Theta)} \theta_k (I - \gamma(A_i + \alpha I))^{k-1}. \tag{23}$$

We can now prove Theorem 1. For convenience, we restate the theorem here.

Theorem 1. *For all $i \in \mathcal{P}$,*

$$\mathbb{E}[\textsc{ClientUpdate}(i, x, \alpha, \gamma, \Theta)] = \nabla \tilde{f}_i(x, \alpha, \gamma, \Theta).$$

Proof. Fix $i \in \mathcal{I}$, x_1, and $\alpha \geq 0$. For $z \in \mathcal{Z}$, define

$$h_\alpha(x; z) := f(x; z) + \frac{\alpha}{2} \|x - x_1\|^2, \quad h_\alpha(x) := \mathop{\mathbb{E}}_{z \sim \mathcal{D}_i}[h_\alpha(x; z)].$$

Further define:

$$B_{z,\alpha} := B_z + \alpha I, \quad b_{z,\alpha} := B_z c_z + \alpha x_1, \quad \tau_{z,\alpha} := \frac{1}{2} c_z^\intercal B_z c_z + \frac{\alpha}{2} \|x_1\|^2.$$

By (1), we have

$$\begin{aligned} h_\alpha(x;z) &= \frac{1}{2}\left\|B_z^{1/2}(x-c_z)\right\|^2 + \frac{\alpha}{2}\|x-x_1\|^2 \\ &= \frac{1}{2}x^\intercal(B_z+\alpha I)x - x^\intercal(B_z c_z + \alpha x_1) + \frac{1}{2}c_z^\intercal B_z c_z + \frac{\alpha}{2}\|x_1\|^2 \\ &= \frac{1}{2}x^\intercal B_{z,\alpha}x - x^\intercal b_{z,\alpha} + \tau_{z,\alpha}. \end{aligned}$$

Therefore,

$$h_\alpha(x) = \frac{1}{2}x^\intercal \mathop{\mathbb{E}}_{z\sim\mathcal{D}_i}[B_{z,\alpha}]x - x^\intercal \mathop{\mathbb{E}}_{z\sim\mathcal{D}_i}[b_{z,\alpha}] + \mathop{\mathbb{E}}_{z\sim\mathcal{D}_i}[\tau_{z,\alpha}]. \tag{24}$$

Define

$$A_\alpha := \mathop{\mathbb{E}}_{z\sim\mathcal{D}_i}[B_{z,\alpha}], \quad c_\alpha := A_\alpha^{-1}\mathop{\mathbb{E}}_{z\sim\mathcal{D}_i}[b_{z,\alpha}], \quad \tau_\alpha := \mathop{\mathbb{E}}_{z\sim\mathcal{D}_i}[\tau_{z,\alpha}].$$

Note that $A_\alpha = A_i + \alpha I$, where A_i is as in (3). Straightforward manipulation of (24) implies

$$h_\alpha(x) = \frac{1}{2}\left\|A_\alpha^{1/2}(x-c_\alpha)\right\|^2 + \tau_\alpha. \tag{25}$$

Note that the stochastic gradients $g_1,\dots,g_K$ computed in Algorithm 2 are therefore independent stochastic gradients of h_α. By applying Lemma 7 and noting that the constant term τ_α does not impact these stochastic gradients, we have that for $k \geq 1$,

$$\mathbb{E}[g_k] = (I-\gamma A_\alpha)^{k-1}A_\alpha(x-c_\alpha). \tag{26}$$

Expanding and using the fact that in Algorithm 2, $x_1 = x$, we have

$$\begin{aligned} \mathbb{E}[g_k] &= (I-\gamma A_\alpha)^{k-1}A_\alpha\left(x - A_\alpha^{-1}\mathop{\mathbb{E}}_{z\sim\mathcal{D}_i}[B_z c_z + \alpha x]\right) \\ &= (I-\gamma A_\alpha)^{k-1}\left(A_\alpha x - \mathop{\mathbb{E}}_{z\sim\mathcal{D}_i}[B_z c_z] - \alpha x\right) \\ &= (I-\gamma(A_i+\alpha I))^{k-1}A_i(x-c_i). \end{aligned}$$

This last step follows from (3). Taking a sum and using the linearity of expectation,

$$\begin{aligned} \mathbb{E}[\textsc{ClientUpdate}(i,x,\alpha,\gamma,\Theta)] &= \sum_{k=1}^{K(\Theta)}\mathbb{E}[g_k] \\ &= \sum_{k=1}^{K(\Theta)}\theta_k(I-\gamma(A_i+\alpha I))^{k-1}A_i(x-c_i) \\ &= Q_i(\alpha,\gamma,\Theta)A_i(x-c_i) \\ &= \nabla\tilde{f}_i(x,\alpha,\gamma,\Theta). \end{aligned}$$

□

A similar analysis using Lemma 7 can be used to derive Theorem 2, which we also restate.

Theorem 2. *If $X_K^i(x)$ is the function that runs K steps of gradient descent on $\mathcal{D}_i$ with learning rate γ starting at x, then*

$$\nabla m_K^i(x) = \nabla_x\tilde{f}_i(x,0,\gamma,\Theta_{2K+1}).$$

Proof of Theorem 2. Fix $i\in\mathcal{I}$, and for convenience of notation, let $X_k := X_k^i(x), X_1 = x$. Thus, for $k\geq 1$ we have

$$X_{k+1} = X_k - \gamma\nabla_{X_k}f_i(X_k).$$

Since $\nabla^2 f_i(y) = A_i$ for all y, we have

$$\nabla_{X_k} X_{k+1} = I - \gamma A_i. \tag{27}$$

By Lemma 7, we also have

$$\nabla_{X_k} f_i(X_k) = (I - \gamma A_i)^{K-1} A_i (x - c_i). \tag{28}$$

For a function $\psi : \mathbb{R}^a \to \mathbb{R}^b$, let its Jacobian at a point $x \in \mathbb{R}^a$ be denoted by $J_x(\psi)$. Applying (27), (28) and the chain rule, we have

$$\begin{aligned}
(\nabla m_K^i(x))^\intercal &= J_{X_1}(f_i(X_{K+1})) \\
&= J_{X_{K+1}}(f_i(X_{K+1})) J_{X_1}(X_{K+1}) \\
&= J_{X_{K+1}}(f_i(X_{K+1})) \prod_{k=1}^{K} J_{X_k}(X_{k+1}) \\
&= [(I - \gamma A_i)^K A_i (x - c_i)]^\intercal \prod_{k=1}^{K} (I - \gamma A_i)^\intercal \\
&= \left((I - \gamma A_i)^{2K} A_i (x - c_i)\right)^\intercal.
\end{aligned}$$

The result follows from applying (23) for $\Theta = \Theta_{2K+1}$. □

B.2 Proofs of Lemmas 1, 2, 3, 4, and 6

These lemmas will follow from a spectral analysis of $Q_i(\alpha, \gamma, \Theta)$. We defer the proof of Lemma 5 to Appendix C due to its more elaborate nature. We first state a general result about eigenvalues of expected values of matrices.

Lemma 8. *Suppose we have a probability space $(\Omega, \mathcal{F}, \mathcal{P})$ where Ω is the set of symmetric matrices in $\mathbb{R}^{d \times d}$, and let B be a random matrix drawn from $\mathcal{P}$. Suppose that $\mathbb{E}[B]$ exists and is finite. Define*

$$\tau_1 := \mathbb{E}[\lambda_{\min}(B)], \quad \tau_2 := \mathbb{E}[\lambda_{\max}(B)].$$

Then $\tau_1 I \preceq \mathbb{E}[B] \preceq \tau_2 I$.

Proof. By standard properties of expectations, $\mathbb{E}[B]$ is a symmetric matrix. Fix $v \in \mathbb{R}^d$ such that $\|v\| = 1$. By the linearity of expectation,

$$v^\intercal \mathbb{E}[B] v = \mathbb{E}[v^\intercal B v] \leq \mathbb{E}[\lambda_{\max}(B)].$$

An analogous argument shows $v^\intercal \mathbb{E}[B] v \geq \mathbb{E}[\lambda_{\min}(B)]$. The result follows. □

We also compute the spectrum of $Q_i(\alpha, \gamma, \Theta)$ and $Q_I(\alpha, \gamma, \Theta) A_i$.

Lemma 9. *For each eigenvalue λ of A_i, $Q_i(\alpha, \gamma, \Theta)$ has an eigenvalue*

$$\sum_{k=1}^{K(\Theta)} \theta_k (1 - \gamma(\lambda + \alpha))^{k-1} \tag{29}$$

and $Q_i(\alpha, \gamma, \Theta) A_i$ has an eigenvalue

$$\sum_{k=1}^{K(\Theta)} \theta_k (1 - \gamma(\lambda + \alpha))^{k-1} \lambda, \tag{30}$$

both with the same multiplicity as λ.

Proof. Let v be an eigenvector of A_i with eigenvalue λ. Then v is an eigenvector of $(I - \gamma(A_i + \alpha I))^{k-1}$ with eigenvalue $(1 - \gamma(\lambda + \alpha))^{k-1}$, implying (29) is an eigenvalue of $Q_i(\alpha, \gamma, \Theta)$ with eigenvector v. Similarly, we note that v is an eigenvector of $(I - \gamma(A_i + \alpha I))^{k-1} A_i$ with eigenvalue $(1 - \gamma(\lambda + \alpha))^{k-1} \lambda$, implying (30) is an eigenvalue of $Q_i(\alpha, \gamma, \Theta) A_i$ with eigenvector v. The statement about multiplicities follows directly. □

Lemma 9 can be used in a straightforward manner to prove Lemma 1, which we restate and prove below.

Lemma 1. *Suppose that $\gamma < (L+\alpha)^{-1}$. Then for all i, $Q_i(\alpha,\gamma,\Theta)$ is positive definite and $\tilde{f}(x,\alpha,\gamma,\Theta)$ is strongly convex.*

Proof. By Assumption 1, we have that for every eigenvalue λ of A_i, $\lambda \le L$. Therefore, for any such λ,

$$1-\gamma(\lambda+\alpha) \ge 1-\gamma(L+\alpha) > 0.$$

Since the θ_k are nonnegative and not all zero by assumption, we see that (29) is a sum of nonnegative terms, at least one of which must be positive. Therefore, all eigenvalues of $Q_i(\alpha,\gamma,\Theta)$ are positive, and $Q_i(\alpha,\gamma,\Theta)$ is therefore symmetric and positive definite.

By Assumption 1, A_i is also symmetric and positive definite, hence the product $Q_i(\alpha,\gamma,\Theta)A_i$ is symmetric positive definite. However, since

$$\nabla_x^2 \tilde{f}_i(x,\alpha,\gamma,\Theta) = Q_i(\alpha,\gamma,\Theta)A_i \succ 0$$

we see that $\tilde{f}_i$ is a strongly convex quadratic function. □

We can also use 9 to derive Lemma 2. In fact, we will show a slightly stronger version, where we also bound the Lipschitz and strong convexity parameters of $\tilde{f}$.

Lemma 10. *Suppose $\gamma < (L+\alpha)^{-1}$. Define*

$$\tilde{L}(\alpha,\gamma,\Theta) := \mathop{\mathbb{E}}_i[\lambda_{\max}(Q_i(\alpha,\gamma,\Theta)A_i)], \tag{31}$$

$$\tilde{\mu}(\alpha,\gamma,\Theta) := \mathbb{E}[\lambda_{\min}(Q_i(\alpha,\gamma,\Theta)A_i)]. \tag{32}$$

Then $\tilde{f}(\alpha,\gamma,\Theta)$ is $\tilde{L}(\alpha,\gamma,\Theta)$-Lipschitz and $\tilde{\mu}(\alpha,\gamma,\Theta)$-strongly convex.

Proof. Recall that the smoothness and strong convexity parameters of are the largest and smallest eigenvalue of its Hessian. Therefore, it suffices to bound the eigenvalues of $\tilde{f}(\alpha,\gamma,\Theta)$. By (7), we have

$$\nabla^2 \tilde{f}_i(x,\alpha,\gamma,\Theta) = Q_i(\alpha,\gamma,\Theta)A_i.$$

By (8), we also have

$$\nabla^2 \tilde{f}(\alpha,\gamma,\Theta) = \mathop{\mathbb{E}}_i[\nabla^2 \tilde{f}_i(\alpha,\gamma,\Theta)] = \mathop{\mathbb{E}}_i[Q_i(\alpha,\gamma,\Theta)A_i].$$

Applying Lemma 8, we conclude the proof. □

Note that $\kappa(\alpha,\gamma,\Theta)$ in (9) is simply the ratio of $\tilde{L}(\alpha,\gamma,\Theta)$ to $\tilde{\mu}(\alpha,\gamma,\Theta)$, so we immediately derive Lemma 2 as a corollary to Lemma 10. Lemmas 3 and 4 are also straightforward consequences of Lemma 10, as we show below. For posterity, we state and prove each one separately.

Recall that as in (10), we define

$$\phi(\lambda,\alpha,\gamma,K) := \sum_{k=1}^{K}(1-\gamma(\lambda+\alpha))^{k-1}\lambda.$$

We can now restate and prove Lemma 3.

Lemma 3. *If $\gamma < (L+\alpha)^{-1}$, $\tilde{f}(x,\alpha,\gamma,\Theta_{1:K})$ is $\phi(L,\alpha,\gamma,K)$-smooth, $\phi(\mu,\alpha,\gamma,K)$-strongly convex, and* $\mathrm{cond}(\tilde{f}) \le \kappa(\alpha,\gamma,\Theta_K)$ *where*

$$\kappa(\alpha,\gamma,\Theta_{1:K}) \le \frac{\phi(L,\alpha,\gamma,K)}{\phi(\mu,\alpha,\gamma,K)}. \tag{11}$$

Proof. By Lemma 10, it suffices to derive upper and lower bounds on the eigenvalues of $Q_i(\alpha,\gamma,\Theta_{1:K})A_i$. Fix $i \in \mathcal{I}$. By Lemma 9, we see that the eigenvalues of $Q_i(\alpha,\gamma,\Theta_{1:K})A_i$ are exactly of the form $\phi(\lambda,\alpha,\gamma,K)$, where λ is an eigenvalue of A_i. By Assumption 1, each such λ satisfies $\lambda \in [\mu, L]$ where $\mu > 0$.

Fix α,γ,K, and define $g(\lambda) := \phi(\lambda,\alpha,\gamma,K)$. Since $\gamma(L+\alpha) < 1$, basic properties of geometric sums imply that for $\lambda \in [\mu, L]$,

$$g(\lambda) = \phi(\lambda,\alpha,\gamma,K) = \frac{1-(1-\gamma(\lambda+\alpha))^K}{\gamma}\frac{\lambda}{\lambda+\alpha}. \tag{33}$$

For a given λ, let $\xi = 1 - \gamma(\lambda + \alpha)$. Simple but tedious computations show that if we take a derivative with respect to λ, we have

$$\gamma(\lambda + \alpha)g'(\lambda) = K\lambda\gamma\xi^{K-1} + (1 - \xi^K)\frac{\alpha}{\lambda + \alpha}.$$

Note that since $\gamma < (L + \alpha)^{-1}$ by assumption, $0 \leq \xi \leq 1$ for $\lambda \in [\mu, L]$. Therefore, $g'(\lambda) \geq 0$ for $\lambda \in [\mu, L]$, so any eigenvalue χ of $Q_i(\alpha, \gamma, \Theta_{1:K})A_i$ must satisfy

$$\phi(\mu, \alpha, \gamma, K) = g(\mu) \leq \chi \leq g(L) = \phi(L, \alpha, \gamma, K).$$

The result then follows by Lemma 10. □

Recall that in (12), we defined

$$\psi(\lambda, \alpha, \gamma, K) := (1 - \gamma(\lambda + \alpha))^{K-1}\lambda.$$

We use a similar proof as that of Lemma 3 to prove Lemma 4, which we restate and prove below.

Lemma 4. *If $\gamma < (KL + \alpha)^{-1}$, $\tilde{f}(x, \alpha, \gamma, \Theta_K)$ is $\psi(L, \alpha, \gamma, K)$-smooth, $\psi(\mu, \alpha, \gamma, K)$-strongly convex, and $\text{cond}(\tilde{f}) \leq \kappa(\alpha, \gamma, \Theta_K)$ where*

$$\kappa(\alpha, \gamma, \Theta_K) \leq \left(\frac{1 - \gamma(L + \alpha)}{1 - \gamma(\mu + \alpha)}\right)^{K-1}\frac{L}{\mu}. \tag{13}$$

Proof. By Lemma 10, it suffices to bound the eigenvalues of $Q_i(\alpha, \gamma, \Theta_K)A_i$. Fix $i \in \mathcal{I}$. By Lemma 9, we see that the eigenvalues of $Q_i(\alpha, \gamma, \Theta_K)$ are of the form $\psi(\lambda, \alpha, \gamma, K)$ where λ is an eigenvalue of A_i. Note that by Assumption 1, any such λ satisfies $\lambda \in [\mu, L]$.

Fix α, γ, K, and define $h(\lambda) := \psi(\lambda, \alpha, \gamma, K)$. Let $\zeta = 1 - \gamma(\lambda + \alpha)$. Straightforward computations show

$$h'(\lambda) = \zeta^{K-2}(1 - \gamma(K\lambda + \alpha)).$$

Since $\gamma < (KL + \alpha)^{-1}$, we in particular have $\gamma < (L + \alpha)^{-1}$ so $0 \leq \zeta \leq 1$ for $\lambda \in [\mu, L]$. Since $\gamma < (KL + \alpha)^{-1}$, we also have the term $1 - \gamma(K\lambda + \alpha) \geq 0$ for $\lambda \in [\mu, L]$. Thus, $h'(\lambda) \geq 0$ for $\lambda \in [\mu, L]$. Thus, any eigenvalue χ of $Q_i(\alpha, \gamma, \Theta_K)A_i$ must satisfy

$$\psi(\mu, \alpha, \gamma, K) = h(\mu) \leq \chi \leq h(L) = \psi(L, \alpha, \gamma, K).$$

The result then follows by Lemma 10. □

Finally, we are now equipped to prove Lemma 6, which we restate here for posterity.

Lemma 6. *Suppose that either (I) $\gamma < (L + \alpha)^{-1}$ and $\Theta = \Theta_{1:K}$ or (II) $\gamma < (KL + \alpha)^{-1}$ and $\Theta = \Theta_K$. Then for all $i \in \mathcal{P}$, $\text{cond}(Q_i(\alpha, \gamma, \Theta)) \leq \kappa_0\kappa(\alpha, \gamma, \Theta)^{-1}$.*

Proof. This will follow almost immediately from Lemmas 3, 4, and 9. First, consider the case $\Theta = \Theta_{1:K}$. Then by Lemma 9 and Assumption 1, we see that

$$\lambda_{\max}(Q_i(\alpha, \gamma, \Theta_{1:K})) \leq \sum_{i=1}^{K}(1 - \gamma(\mu + \alpha))^{K-1} = \frac{\phi(\mu, \alpha, \gamma, K)}{\mu}$$

and

$$\lambda_{\min}(Q_i(\alpha, \gamma, \Theta_{1:K})) \geq \sum_{i=1}^{K}(1 - \gamma(L + \alpha))^{K-1} = \frac{\phi(L, \alpha, \gamma, K)}{L}.$$

Therefore,

$$\text{cond}(Q_i(\alpha, \gamma, \Theta_{1:K})) \leq \frac{\phi(\mu, \alpha, \gamma, K)}{\phi(L, \alpha, \gamma, K)}\frac{L}{\mu} = \kappa_0\kappa(\alpha, \gamma, \Theta_{1:K})^{-1}.$$

Here we used the fact that $\kappa_0 := L/\mu$ and Lemma 3. An almost identical proof gives the analogous result for $\Theta = \Theta_K$. □

B.3 Tightness of Lemmas 3 and 4

In fact, Lemmas 3 and 4 are tight. Fix any $\alpha \geq 0$ and $\gamma < (\alpha + L)^{-1}$. Let $\mathcal{P}$ be supported on a single client i, and let this client's dataset be supported on a single example z where

$$B_z = \begin{pmatrix} L & 0 \\ 0 & \mu \end{pmatrix}, c_z = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Note that by (3), we then have $A_i = B_i, c_i = 0$. In fact, we will show that in this case, the bounds on the condition numbers given in Lemma 3 and 4 are tight. By direct computation,

$$Q_i(\alpha, \gamma, \Theta_{1:K})A_i = \begin{pmatrix} \phi(L, \alpha, \gamma, K) & 0 \\ 0 & \phi(\mu, \alpha, \gamma, K) \end{pmatrix}.$$

If $\gamma < (L + \alpha)^{-1}$ then similar reasoning to the proof of Lemma 3 implies that the condition number satisfies

$$\text{cond}(\tilde{f}_i(\alpha, \gamma, \Theta_{1:K})) = \frac{\lambda_{\max}(Q_i(\alpha, \gamma, \Theta_{1:K})A_i)}{\lambda_{\min}(Q_i(\alpha, \gamma, \Theta_{1:K})A_i)} = \frac{\phi(L, \alpha, \gamma, K)}{\phi(L, \alpha, \gamma, K)}.$$

Similarly, we have that

$$Q_i(\alpha, \gamma, \Theta_K)A_i = \begin{pmatrix} \psi(L, \alpha, \gamma, K) & 0 \\ 0 & \psi(\mu, \alpha, \gamma, K) \end{pmatrix}.$$

By analogous reasoning to the proof of Lemma 4, if $\gamma < (KL + \alpha)^{-1}$, we have

$$\text{cond}(\tilde{f}_i(\alpha, \gamma, \Theta_{1:K})) = \frac{\lambda_{\max}(Q_i(\alpha, \gamma, \Theta_K)A_i)}{\lambda_{\min}(Q_i(\alpha, \gamma, \Theta_K)A_i)} = \frac{\psi(L, \alpha, \gamma, K)}{\psi(L, \alpha, \gamma, K)}.$$

C Proof of Lemma 5

In order to prove the results in this section, we will use the following straightforward lemma regarding the structure of $x^*(\alpha, \gamma, \Theta)$.

Lemma 11. *Suppose $\gamma < (L + \alpha)^{-1}$. Then*

$$x^*(\alpha, \gamma, \Theta) = \mathbb{E}\left[Q_i(\alpha, \gamma, \Theta)A_i\right]^{-1} \mathbb{E}\left[Q_i(\alpha, \gamma, \Theta)A_i c_i\right].$$

Proof. By definition,

$$\tilde{f}(x, \alpha, \gamma, \Theta) = \mathbb{E}\left[\frac{1}{2}\left\|(Q_i(\alpha, \gamma, \Theta)A_i)^{1/2}(x - c_i)\right\|^2\right].$$

Therefore,

$$\nabla \tilde{f}(x, \alpha, \gamma, \Theta) = \mathbb{E}[Q_i(\alpha, \gamma, \Theta)A_i(x - c_i)] = \mathbb{E}[Q_i(\alpha, \gamma, \Theta)A_i]x - \mathbb{E}[Q_i(\alpha, \gamma, \Theta)A_i c_i].$$

Since $\gamma < (L + \alpha)^{-1}$, we know by Lemma 1 that $\tilde{f}$ is strongly convex in x. It then follows that

$$x^*(\alpha, \gamma, \Theta) = \mathbb{E}[Q_i(\alpha, \gamma, \Theta)A_i]^{-1} \mathbb{E}[Q_i(\alpha, \gamma, \Theta)A_i c_i].$$

□

To prove Lemma 5, we will reduce it to a statement about mean absolute deviations of bounded random variables. We define the mean absolute deviation of a random variable below.

Definition 1. *Let X be a random variable in some Banach space over $\mathbb{R}$. The mean absolute deviation of X is*

$$D(X) := \mathbb{E}[\|X - \mathbb{E}[X]\|].$$

C.1 One-dimensional Case

We first proceed for $d = 1$, so that $Q_i(\alpha, \gamma, \Theta), A_i, c_i \in \mathbb{R}$. We do this both for expository purposes, as when $d = 1$ we can rely on classical versions of the mean absolute deviation, and because we actually derive tight bounds for $d = 1$.

To derive our results, we bound the mean absolute deviation of bounded random variables. Our result is inspired by the bound by Bhatia and Davis (2000) on the variance of bounded random variables.

Theorem 5. *Suppose X is a discrete random variable taking values in $[a, b] \subseteq \mathbb{R}$ for $a < b$. Then*

$$D(x) \leq \frac{2(b - \mathbb{E}[X])(\mathbb{E}[X] - a)}{b - a}.$$

Moreover, this holds with equality iff X is supported on $\{a, b\}$.

Proof. Suppose X takes on values $x_1, \ldots, x_n$ with probabilities $p_1, \ldots p_n$. We will first show that there is a random variable Y supported on $\{a, b\}$ such that $D(X) \leq D(Y)$.

Without loss of generality, suppose $x_1 \in (a, b)$. Define

$$s := p_1 \frac{b - x_1}{b - a}, \quad t := p_1 \frac{x_1 - a}{b - a}.$$

First note that $s, t \in [0, 1]$. Simple analysis also shows

$$s + t = p_1 \tag{34}$$

$$sa + tb = p_1 x_1. \tag{35}$$

Let X' be the random variable taking on values $a, x_2, \ldots, x_n, b$ with probabilities $s, p_2, \ldots p_n, t$. Note that these probabilities are nonnegative and sum to 1 by (34). By (35), we also have $\mathbb{E}[X'] = \mathbb{E}[X]$.

It is straightforward to show that $D(X') \geq D(X)$. Suppose $x_1 \geq \mathbb{E}[X]$. Then we have

$$\begin{aligned}
D(X') - D(X) &= s(\mathbb{E}[X] - a) + t(b - \mathbb{E}[X]) - p_1(x_1 - \mathbb{E}[X]) \\
&= p_1 \left(\frac{(b - x_1)(\mathbb{E}[X] - a)}{b - a} + \frac{(x_1 - a)(b - \mathbb{E}[X])}{b - a} - (x_1 - \mathbb{E}[X]) \right) \\
&= \frac{p_1}{b - a} \left((b - x_1)(\mathbb{E}[X] - a) + (x_1 - a)(b - \mathbb{E}[X]) - (b - a)(x_1 - \mathbb{E}[X]) \right) \\
&= \frac{p_1}{b - a} \left((b - x_1)(2\,\mathbb{E}[X] - a - x_1) + (x_1 - a)(b - x_1) \right) \\
&= \frac{2p_1(b - x_1)(\mathbb{E}[X] - a)}{b - a} \\
&\geq 0.
\end{aligned}$$

An analogous argument shows that $D(X') \geq D(X)$ when $x_1 \leq \mathbb{E}[X]$. A thorough examination of the derivation above shows that this inequality is strict if and only if $x_1 \in (a, b)$ and $p_1 > 0$. Thus, $D(X') \geq D(X)$ and X' has one fewer possible outcome in the range (a, b).

By iterating this procedure, (which is guaranteed to terminate after at most n iterations), we obtain some random variable Y supported on $\{a, b\}$ such that $D(X) \leq D(Y)$, with equality if and only if X is already supported on $\{a, b\}$. Suppose Y takes on a, b with probabilities $(1 - p), p$. Straightforward calculation shows

$$D(Y) = 2(1 - p)p(b - a) = \frac{2(b - \mathbb{E}[X])(\mathbb{E}[X] - a)}{b - a}.$$

□

Using the fact that for any real x, $(b - a)^2 \geq 4(b - x)(x - a)$ (with equality iff $x = (b + a)/2$) we arrive at the following corollary, analogous to Popoviciu's inequality on variances (Popoviciu, 1935).

Corollary 2. *If X is a discrete random variable on $[a, b]$, then*

$$D(x) \leq \frac{1}{2}(b-a),$$

with equality iff X takes on the values a and b, each with probability $1/2$.

We can now prove a stronger version of Lemma 5 when $d = 1$. For simplicity, we assume $\mathcal{P}$ is a discrete distribution on some finite $\mathcal{I}$, though the analysis can be generalized to arbitrary distributions.

Lemma 12. *Let $b = \max_{i\in\mathcal{I}} \lambda_{\max}(Q_i(\alpha, \gamma, \Theta))$, $a = \min_{i\in\mathcal{I}} \lambda_{\min}(Q_i(\alpha, \gamma, \Theta))$. Then for $d = 1$,*

$$\|x^*(\alpha, \gamma, \Theta) - x^*\| \leq 2C\frac{\sqrt{b} - \sqrt{a}}{\sqrt{b} + \sqrt{a}}.$$

Proof. Suppose $|\mathcal{I}| = [n]$ and $\mathcal{P}$ is the discrete distribution on $\mathcal{I}$ with associated probabilities $p_1, \ldots, p_n$. Without loss of generality, assume $p_1, \ldots, p_n > 0$. For brevity, we will let Q_i denote $Q_i(\alpha, \gamma, \Theta)$. Note that since $d = 1$, Q_i, A_i, c_i are all elements of $\mathbb{R}$. By Assmptions 1 and 2, and by definition of a, b, we have that for all i, $0 < \mu \leq A_i \leq L$, $Q_i \in [a, b]$, and $|c_i| \leq C$. Moreover, by Lemma 1, we must have $a > 0$.

Let $q_i = p_i A_i$. Note that $q_i \geq p_i\mu > 0$. Thus, the q_i define a unique distribution $\mathcal{Q}$ over $\mathcal{I}$ where i is sampled with probability proportional to q_i. Let Z denote the random variable Q_i where $i \sim \mathcal{Q}$ and let $v = \mathbb{E}[Z]$. Thus,

$$v = \mathbb{E}[Z] = \frac{\sum_{i=1}^n q_i Q_i}{\sum_{i=1}^n q_i} = \frac{\sum_{i=1}^n p_i Q_i A_i}{\sum_{i=1}^n p_i A_i}.$$

Since each $Q_i \in [a, b]$, we also have $v \in [a, b]$.By Lemma 11,

$$\begin{aligned}
\|x^*(\alpha, \gamma, \Theta) - x^*\| &= \left\| \frac{\sum_{i=1}^n q_i Q_i c_i}{\sum_{i=1}^n q_i Q_i} - \frac{\sum_{i=1}^n q_i c_i}{\sum_{i=1}^n q_i} \right\| \\
&= \left\| \frac{\sum_{i=1}^n q_i Q_i c_i - v\sum_{i=1}^n q_i c_i}{\sum_{i=1}^n q_i Q_i} \right\| \\
&= \left\| \frac{\sum_{i=1}^n (Q_i - v) q_i c_i}{\sum_{i=1}^n q_i Q_i} \right\| \\
&= \left\| \frac{\sum_{i=1}^n (Q_i - v) q_i c_i}{\sum_{i=1}^n q_i Q_i} \frac{\sum_{i=1}^n q_i}{\sum_{i=1}^n q_i} \right\| \\
&= \frac{\|\mathbb{E}_{i\sim\mathcal{Q}}(Q_i - v)c_i\|}{v} \\
&\leq \frac{\mathbb{E}_{i\sim\mathcal{Q}}[\|(Q_i - v)c_i\|]}{v} \\
&\leq \frac{C\,\mathbb{E}[\|Z - v\|]}{v}.
\end{aligned}$$

Since $v = \mathbb{E}[Z]$, we can apply Theorem 5, finding

$$\|x^*(\alpha, \gamma, \Theta) - x^*\| \leq \frac{2C(b - v)(v - a)}{v(b - a)}.$$

Maximizing the right-hand side for $v \in [a, b]$, we get

$$\|x^*(\alpha, \gamma, \Theta) - x^*\| \leq 2C\frac{\sqrt{b} - \sqrt{a}}{\sqrt{b} + \sqrt{a}}.$$

□

C.2 Tightness of Lemma 12

In this section, we will show that in some sense, Lemma 12 is tight. Specifically, we will show that for all $\epsilon > 0$, there are $A_1, A_2 \in \mathbb{R}$, $c_1, c_2 \in \mathbb{R}$ such that $\|c_i\| \leq 1$, a distribution $\mathcal{P}$ over $\{1, 2\}$, and α, γ, Θ such that for

$$a = \min\{\lambda_{\min}(Q_1(\alpha, \gamma, \Theta)|i = 1, 2\}, \quad b = \max\{\lambda_{\max}(Q_i(\alpha, \gamma, \Theta)|i = 1, 2\},$$

we have

$$\|x^*(\alpha, \gamma, \Theta) - x^*\| \geq 2\frac{\sqrt{b} - \sqrt{a}}{\sqrt{b} + \sqrt{a}} - \epsilon.$$

Let $A_1 = 4, A_2 = 1, c_1 = 1, c_2 = -1$. Let $\alpha = 0, \gamma = 1/8$. Then by (6), we have

$$Q_1(\alpha, \gamma, \Theta_K) = \left(\frac{1}{2}\right)^K, \quad Q_2(\alpha, \gamma, \Theta_K) = \left(\frac{7}{8}\right)^K.$$

Therefore, in this setting, $b = (7/8)^K, a = 2^{-K}$. By applying L'Hopital's rule, we find

$$\lim_{K \to \infty} 2\frac{\sqrt{b} - \sqrt{a}}{\sqrt{b} + \sqrt{a}} = 2. \tag{36}$$

Let $\mathcal{P}$ be the distribution that selects $i = 1$ with probability p and $i = 2$ with probability $1 - p$. Then

$$\begin{aligned}\|x^*(\alpha, \gamma, \Theta_K) - x^*\| &= \left|\frac{4pQ_1 - (1-p)Q_2}{4pQ_1 + (1-p)Q_2} - \frac{5p-1}{3p+1}\right| \\ &= \left|\frac{4p\left(\frac{1}{2}\right)^K - (1-p)\left(\frac{7}{8}\right)^K}{4p\left(\frac{1}{2}\right)^K + (1-p)\left(\frac{7}{8}\right)^K} - \frac{5p-1}{3p+1}\right|.\end{aligned}$$

For any fixed $p \in (0, 1)$, straightforward but tedious applications of L'Hopital's rule yields the fact that

$$\lim_{K \to \infty} \|x^*(\alpha, \gamma, \Theta_K) - x^*\| = \frac{8p}{3p+1} \tag{37}$$

By (36) and (37), we see that by selecting K sufficiently large and p sufficiently close to 1, we can ensure that

$$\|x^*(\alpha, \gamma, \Theta_K) - x^*\| \geq 2 - \epsilon$$

and that

$$2\frac{\sqrt{b} - \sqrt{a}}{\sqrt{b} + \sqrt{a}} \geq 2 - \epsilon$$

thus implying that

$$\|x^*(\alpha, \gamma, \Theta_K) - x^*\| \geq 2\frac{\sqrt{b} - \sqrt{a}}{\sqrt{b} + \sqrt{a}} - \epsilon.$$

C.3 General Case

In this case, we will use a similar proof strategy. However, we will use a matrix-weighted version of the mean absolute deviation. Suppose we have positive-definite symmetric matrices $X_1, \ldots, X_n, Y_1, \ldots Y_n \in \mathbb{R}^{d \times d}$ such that for all i, X_i and Y_i commute. Let $Y = \sum_{i=1}^n Y_i$.

Definition 2. *The matrix-weighted mean of $\{X_1, \ldots X_n\}$ with respect to $\{Y_1, \ldots Y_n\}$ is given by*

$$f(X_1, \ldots, X_n | Y_1, \ldots, Y_n) := \left(\sum_{i=1}^n X_i Y_i\right) Y^{-1}.$$

When $d = 1$, and the $Y_i > 0$, this gives the standard mean of a discrete random variable X taking values $X_1, \ldots X_n$ with probabilities $Y_1, \ldots Y_n$. When the context is clear, we will simply denote this by $f(X|Y)$. We first prove a simple lemma regarding the Loewner ordering and matrix-weighted means.

Lemma 13. *Suppose that for all i, $aI \preceq X_i \preceq bI$. Then $aI \preceq f(X|Y) \preceq bI$.*

This generalizes the fact that if X is a random variable taking values in $[a, b]$, then $a \leq \mathbb{E}[X] \leq b$ to symmetric positive-definite matrices.

Proof. Since the X_i, Y_i are commuting positive definite matrices and Y is positive definite, $f(X|Y)$ is similar to the matrix

$$P = Y^{-1/2} \left(\sum_{i=1}^{n} Y_i^{1/2} X_i Y_i^{1/2} \right) Y^{-1/2}.$$

By assumption, $X_i \preceq bI$. Therefore,

$$Y_i^{1/2} X_i Y_i^{1/2} \preceq Y_i^{1/2} b I Y_i^{1/2} = b Y_i.$$

Hence, we have

$$\sum_{i=1}^{n} Y_i^{1/2} X_i Y_i^{1/2} \preceq bY.$$

Thus,

$$P \preceq bY^{-1/2} Y Y^{-1/2} = bI.$$

An analogous argument shows that $P \succeq aI$. By basic properties of matrix similarity, we therefore find $aI \preceq f(X|Y) \preceq bI$. □

We can use this matrix-weighted mean to define a normalized, matrix-weighted version of the mean absolute deviation.

Definition 3. *The normalized matrix-weighted discrepancy of $\{X_1, \dots X_n\}$ with respect to $\{Y_1, \dots Y_n\}$ is given by*

$$M(X_1, \dots X_n | Y_1, \dots, Y_n) = \sum_{i=1}^{n} \left\| Y^{-1} f(X|Y)^{-1} (X_i - f(X|Y)) Y_i \right\|$$

where $\|\cdot\|$ is the operator norm.

Note that when $d = 1$, $M(X|Y) = D(X)/|\mathbb{E}[X]|$, where $D(X)$ is as in Definition 1 and X takes on values $X_1, \dots X_n > 0$ with probabilities $Y_1, \dots Y_n > 0$.

We will prove an analog of Theorem 5 for this normalized matrix-weighted discrepancy.

Theorem 6. *Let $X_1, \dots X_n$ be symmetric matrices in $\mathbb{R}^{d \times d}$ satisfying $aI \preceq X_i \preceq bI$ for all i, and suppose $Y_1, \dots Y_n$ are symmetric, positive-definite matrices in $\mathbb{R}^{d \times d}$. Then*

$$M(X_1, \dots X_n | Y_1, \dots Y_n) \leq \frac{2(b-a)}{b}.$$

To prove this, we will require a straightforward lemma regarding eigenvalues of symmetric positive definite matrices.

Lemma 14. *Let P_1, P_2 be symmetric positive definite matrices and let $P = P_1 + P_2$. Then*

$$\left\| P_1 P^{-1} \right\| \leq 1.$$

Proof. By basic properties of the Loewner ordering,

$$P_1 \preceq P \implies P^{-1/2} P_1 P^{-1/2} \preceq P^{-1/2} P P^{-1/2} = I.$$

Since $P_1 P^{-1}$ is similar to $P^{-1/2} P_1 P^{-1/2}$, we have

$$\left\| P_1 P^{-1} \right\| = \left\| P^{-1/2} P_1 P^{-1/2} \right\| \leq 1.$$

□

We can now prove Theorem 6.

Proof of Theorem 6. We will proceed in a similar manner to the proof of Theorem 5. We will first show that we can always find a set of symmetric positive definite matrices $X_1', \dots, X_m', Y_1', \dots, Y_m'$ such that:

1. For all i, $aI \preceq X_i' \preceq bI$.
2. For all i, X_i', Y_i' commute.
3. $\sum_i Y_i' = \sum_i Y_i$.
4. $f(X'|Y') = f(X|Y)$.
5. $M(X'|Y') \geq M(X|Y)$.
6. For $t = |\{X_1, \dots, X_n\} \backslash \{aI, bI\}|$, we have $|\{X_1', \dots, X_m'\} \backslash \{aI, bI\}| \leq \max\{0, t-1\}$.

By iterating this procedure, we can replace $X_1, \dots, X_n, Y_1, \dots, Y_n$ with matrices $X_1'', \dots, X_l'', Y_1'', \dots, Y_l''$ where the X_i'' are all in the set $\{aI, bI\}$. It will then suffice to show that $M(X''|Y'')$ satisfies the desired bound, which we do by a somewhat direct computation, though one that is made much easier due to the fact that aI, bI are diagonal.

We now proceed in detail. Define the matrices

$$S := \frac{bI - X_1}{b-a} Y_1, \quad T := \frac{X_1 - aI}{b-a} Y_1.$$

Note that since X_1, Y_1 commute, S and T are products of symmetric, positive definite, commuting matrices. They are therefore symmetric, positive definite, commuting matrices as well. One can easily verify that

$$S + T = Y_1 \tag{38}$$

and

$$aS + bT = X_1 Y_1. \tag{39}$$

By (38) we have

$$S + Y_2 + \cdots + Y_n + T = Y. \tag{40}$$

and combining (39) and (40), we have

$$f(aI, X_2, \dots, X_n, bI | S, Y_2, \dots Y_n, T) = f(X_1, \dots, X_n | Y_1, \dots, Y_n). \tag{41}$$

We will use Z to denote the matrices in (41). Note that by Lemma 13, we know that $aI \preceq Z \preceq bI$. We will show that this replacement of (X_1, Y_1) by (aI, S) and (bI, T) does not decrease the mean absolute deviation. By (40) and (41),

$$\begin{aligned}
&M(aI, X_2, \dots, X_n, bI | S, Y_2, \dots, Y_n, T) - M(X_1, \dots, X_n | Y_1, \dots, Y_n) \\
&= \left\| Y^{-1} Z^{-1} (Z - aI) S \right\| + \left\| Y^{-1} Z^{-1} (bI - Z) T \right\| - \left\| Y^{-1} Z^{-1} (Z - X_1) Y_1 \right\| \\
&= \frac{\left\| Y^{-1} Z^{-1} (Z - aI)(bI - X_1) Y_1 \right\| + \left\| Y^{-1} Z^{-1} (bI - Z)(X_1 - aI) Y_1 \right\| - \left\| Y^{-1} Z^{-1} (bI - aI)(Z - X_1) Y_1 \right\|}{b-a}
\end{aligned}$$

Define

$$\begin{aligned}
T_1 &= Y^{-1} Z^{-1} (Z - aI)(bI - X_1) Y_1, \\
T_2 &= Y^{-1} Z^{-1} (bI - Z)(X_1 - aI) Y_1, \\
T_3 &= Y^{-1} Z^{-1} (bI - aI)(Z - X_1) Y_1.
\end{aligned}$$

Note that since $aI \preceq Z \preceq bI, aI \preceq X_1 \preceq bI$ and the Y_i are positive definite, T_1 and T_2 are positive semi-definite matrices. Simple algebraic manipulation implies that $T_3 = T_1 - T_2$. Since T_1, T_2 are positive definite matrices, we have

$$\|T_1 - T_2\| \leq \max\left\{\lambda_{\max}(T_1) - \lambda_{\min}(T_2), \lambda_{\max}(T_2) - \lambda_{\min}(T_1)\right\} \leq \lambda_{\max}(T_1) + \lambda_{\max}(T_2) = \|T_1\| + \|T_2\|.$$

Thus,

$$M(aI, X_2, \ldots, X_n, bI | S, Y_2, \ldots, Y_n, T) \geq M(X_1, \ldots, X_n | Y_1, \ldots, Y_n).$$

We therefore exhibit exactly the matrices satisfying properties (1)-(6) described above. By iterating this procedure, we obtain positive definite, symmetric matrices $(X_1'', \ldots X_m''), (Y_1'', \ldots, Y_m'')$ such that X_i'', Y_i'' commute, each X_i'' is equal to aI or bI, and such that

$$M(X_1'', \ldots, X_m'' | Y_1'' \ldots Y_m'') \geq M(X_1, \ldots X_n | Y_1 \ldots Y_n).$$

By consolidating X_i'' that are equal, we can assume without loss of generality that we have matrices (aI, bI) with associated symmetric positive definite matrices (C_1, C_2). Let $C = C_1 + C_2$, and let $R = aC_1 + bC_2$. We then have

$$f(aI, bI | C_1, C_2) = RC^{-1}.$$

Let $Z = RC^{-1}$. Then by direct computation,

$$\begin{aligned} M(aI, bI | C_1, C_2) &= \left\|R^{-1}(aI - Z)C_1\right\| + \left\|R^{-1}(bI - Z)C_2\right\| \\ &= \left\|(aR^{-1} - C^{-1})C_1\right\| + \left\|(C^{-1} - bR^{-1})C_2\right\|. \end{aligned}$$

After some straightforward but tedious algebraic manipulation, we find

$$\begin{aligned} M(aI, bI | C_1, C_2) &= (b-a)\left\|R^{-1}C_2C^{-1}C_1\right\| + (b-a)\left\|C^{-1}C_1R^{-1}C_2\right\| \\ &\leq (b-a)\left\|R^{-1}C_2\right\|\left\|C^{-1}C_1\right\| + (b-a)\left\|C^{-1}C_1\right\|\left\|R^{-1}C_2\right\|. \end{aligned}$$

Since C_1, C_2 are symmetric positive definite matrices, we have

$$\left\|C^{-1}C_1\right\| = \lambda_{\min}(CC_1^{-1})^{-1} = \frac{1}{\lambda_{\min}(I + C_2C_1^{-1})} = \frac{1}{1 + \lambda_{\min}(C_2C_1^{-1})}.$$

An analogous computation shows that

$$\left\|R^{-1}C_2\right\| = \frac{1}{a\lambda_{\min}(C_1C_2^{-1}) + b}.$$

Letting p, q denote $\lambda_{\min}(C_2C_1^{-1}), \lambda_{\min}(C_1C_2^{-1})$ respectively, and noting that we therefore have $p, q > 0$, we have

$$M(aI, bI | C_1, C_2) \leq \frac{2(b-a)}{(1+q)(ap+b)} \leq \frac{2(b-a)}{b}.$$

□

We can now prove Lemma 5.

Proof. Suppose $|\mathcal{I}| = n$ and $\mathcal{P}$ is the discrete distribution on $\mathcal{I}$ with associated probabilities p_i. For brevity, we

will let Q_i denote $Q_i(\alpha, \gamma, \Theta)$. We will let $Y_i = p_i A_i$ and $Y = \sum_{i=1}^n Y_i$. By Lemma 11, we have

$$\begin{aligned}
\|x^*(\alpha,\gamma,\Theta) - x^*\| &= \left\| \left(\sum_{i=1}^n Q_i Y_i\right)^{-1} \left(\sum_{i=1}^n Q_i Y_i c_i\right) + \left(\sum_{i=1}^n Y_i\right)^{-1} \left(\sum_{i=1}^n Y_i c_i\right) \right\| \\
&= \left\| \left(\sum_{i=1}^n Q_i Y_i\right)^{-1} \left(\left(\sum_{i=1}^n Q_i Y_i c_i\right) + \left(\sum_{i=1}^n Q_i Y_i\right) Y^{-1} \left(\sum_{i=1}^n Y_i c_i\right) \right) \right\| \\
&= \left\| \left(\sum_{i=1}^n Q_i Y_i\right)^{-1} \left(\sum_{i=1}^n (Q_i - f(Q|Y)) Y_i c_i\right) \right\| \\
&\leq \sum_{i=1}^n \left\| Y^{-1} f(Q|Y)^{-1} (Q_i - f(Q|Y)) Y_i \right\| \|c_i\| \\
&\leq CM(Q|Y).
\end{aligned}$$

Here we used Lemma 9, which in particular shows that since $\gamma < (L+\alpha)^{-1}$, all the Q_i are positive definite. Moreover, Lemma 9 shows that Q_i, A_i share the same eigenvectors, and therefore commute with one another. Hence, the Q_i commute with the Y_i. Moreover, by Assumption 1, the Y_i are positive definite symmetric matrices, and by Assumption 2, $\|c_i\| \leq C$. Applying Theorem 6, we have

$$\|x^*(\alpha,\gamma,\Theta) - x^*\| \leq 2C\frac{b-a}{b} \leq 8C\frac{\sqrt{b}-\sqrt{a}}{\sqrt{b}+\sqrt{a}}$$

The last inequality holds from simple algebraic manipulation.

□

D Additional Pareto Frontiers

D.1 Simulated MAML-style Pareto Frontiers

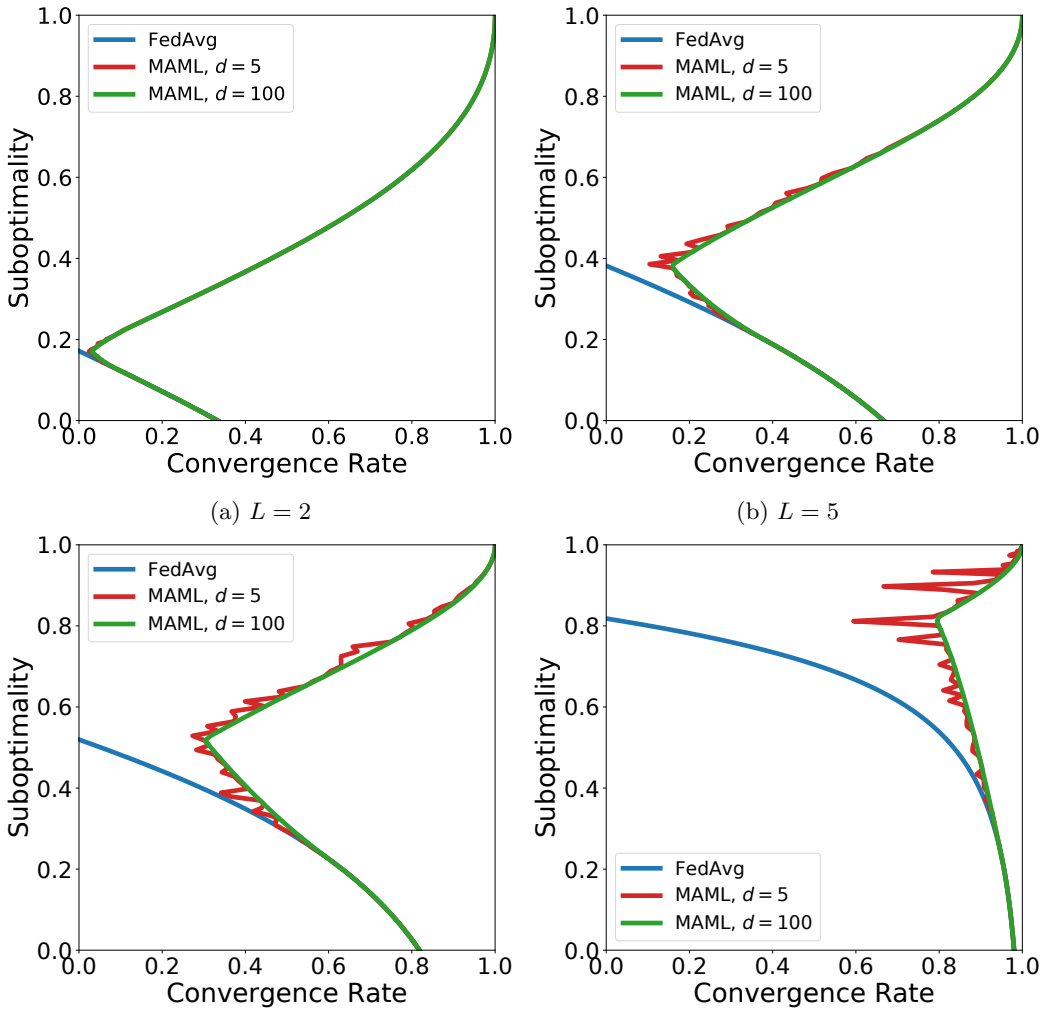
In order to plot Pareto frontiers for MAML-style methods ($\Theta = \Theta_K$) when $\gamma > (KL+\alpha)^{-1}$, we generate random symmetric matrices $A \in \mathbb{R}^{d\times d}$ satisfying $\mu I \preceq A \preceq LI$. We then compute the associated matrix $Q(\alpha, \gamma, \Theta_K)$. Once we have these matrices, we can compute the condition number κ of $Q(\alpha, \gamma, \Theta_K)A$. We can then compute ρ, Δ by substituting κ into Table 2 and Lemma 5.

To generate A, we generate B by sampling its entries independently from $\mathcal{N}(0, 1)$. We then set $A = \beta_1 B^\intercal B + \beta_2 I$, where β_1, β_2 are the unique scalars such that

$$\mu = \lambda_{\min}(A) \leq \lambda_{\max}(A) = L.$$

We plot the resulting Pareto frontiers for varying L and fixed μ in Figure 7. We see that as L increases with respect to μ, the discrepancy between the MAML curves and the FedAvg curves grows. In particular, for small L/μ, we see that the MAML curve recovers most of the FedAvg curve before diverging, while for large L/μ, the two diverge almost immediately. Again, we see that when $d = 5$, there is some noise in (ρ, Δ), which seems to approach some limiting behavior for $d = 100$.

We perform a similar experiment, but where we fix $L = 10$ and vary ServerOpt over gradient descent with no momentum, with Nesterov momentum, and with heavy-ball momentum. The results are given in Figure 8. While the differences are not huge, we see that momentum helps convergence in all cases, FedAvg or MAML. Moreover, we see an interesting phenomenon where the type of momentum changes the concavity of the MAML Pareto frontier for $d = 100$. As we add momentum, the region to the right of the MAML curve becomes more convex, becoming more rounded for heavy-ball momentum than for Nesterov momentum.



(a) $L = 2$ (b) $L = 5$

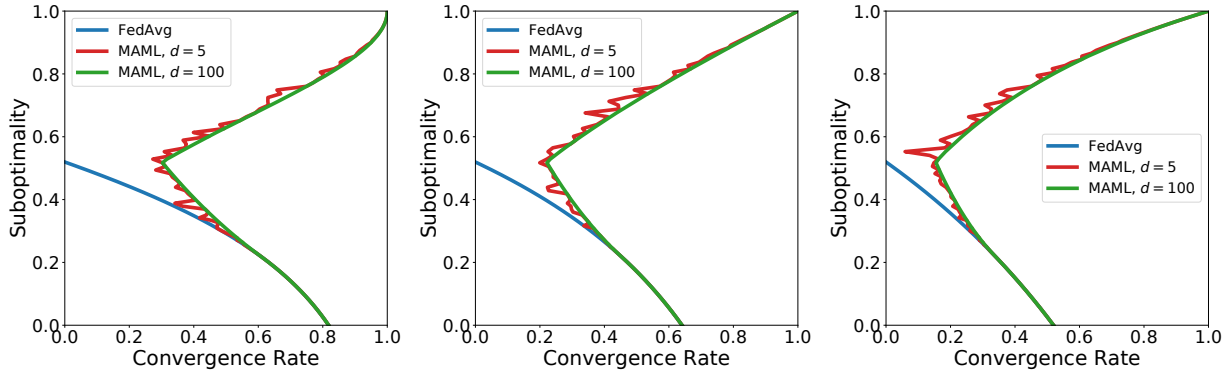
(c) $L = 10$ (d) $L = 100$

Figure 7: Simulated MAML Pareto frontiers for $\mu = 1, \gamma = 0.001, \Theta = \Theta_K$, varying L, and where ServerOpt is gradient descent. We vary $L \in \{2, 5, 10, 100\}$ and $K \in [1, 10^6]$. We randomly generate $A \in \mathbb{R}^{d \times d}$ with $\mu I \preceq A \preceq LI$ and compute the associated (ρ, Δ). We also plot the Pareto frontier for $\Theta_{1:K}$ and the same L.

D.2 Proximal MAML-style Pareto Frontiers

In Figure 9 we plot the analog of the Pareto frontiers in Figure 3, but for MAML-style algorithms where $\Theta = \Theta_K$. We see a similar, though more subdued, version of the behavior in Figure 3. That is, adding a proximal term simply alters how much of the Pareto frontier is traversed; it does not change the fundamental shape. Note that here we only used γ satisfying $\gamma < (KL + \alpha)^{-1}$, as required by Theorem 3. In particular, the only restriction on the shape of the curve seems to be coming from the fact that larger α reduces the set of γ satisfying $\gamma < (KL + \alpha)^{-1}$.

To see the effects of α when $\gamma \geq (KL + \alpha)^{-1}$, we use the same simulated approach as in Figure 5. We do this for varying α in Figure 10. We see that while increasing α shrinks the space of the Pareto curve of FedAvg, it does not seem to change the MAML curves by a meaningful amount.



(a) ServerOpt: Gradient descent with no momentum

(b) ServerOpt: Gradient descent with Nesterov momentum

(c) ServerOpt: Gradient descent with heavy-ball momentum

Figure 8: Simulated MAML Pareto frontiers for $\mu = 1, L = 10, \gamma = 0.001, \Theta = \Theta_K$, and where ServerOpt is gradient descent with various types of momentum. We vary $K \in [1, 10^6]$. We randomly generate $A \in \mathbb{R}^{d \times d}$ with $\mu I \preceq A \preceq LI$ and compute the associated (ρ, Δ). We also compare to the Pareto frontier for $\Theta_{1:K}$ with the same ServerOpt.

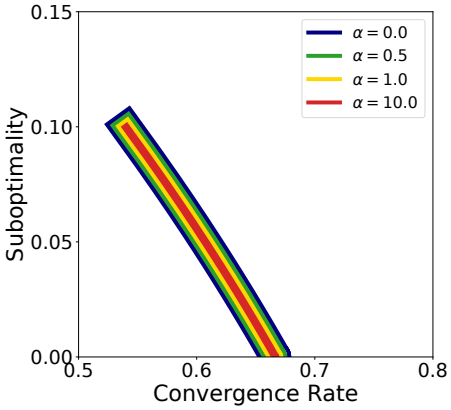
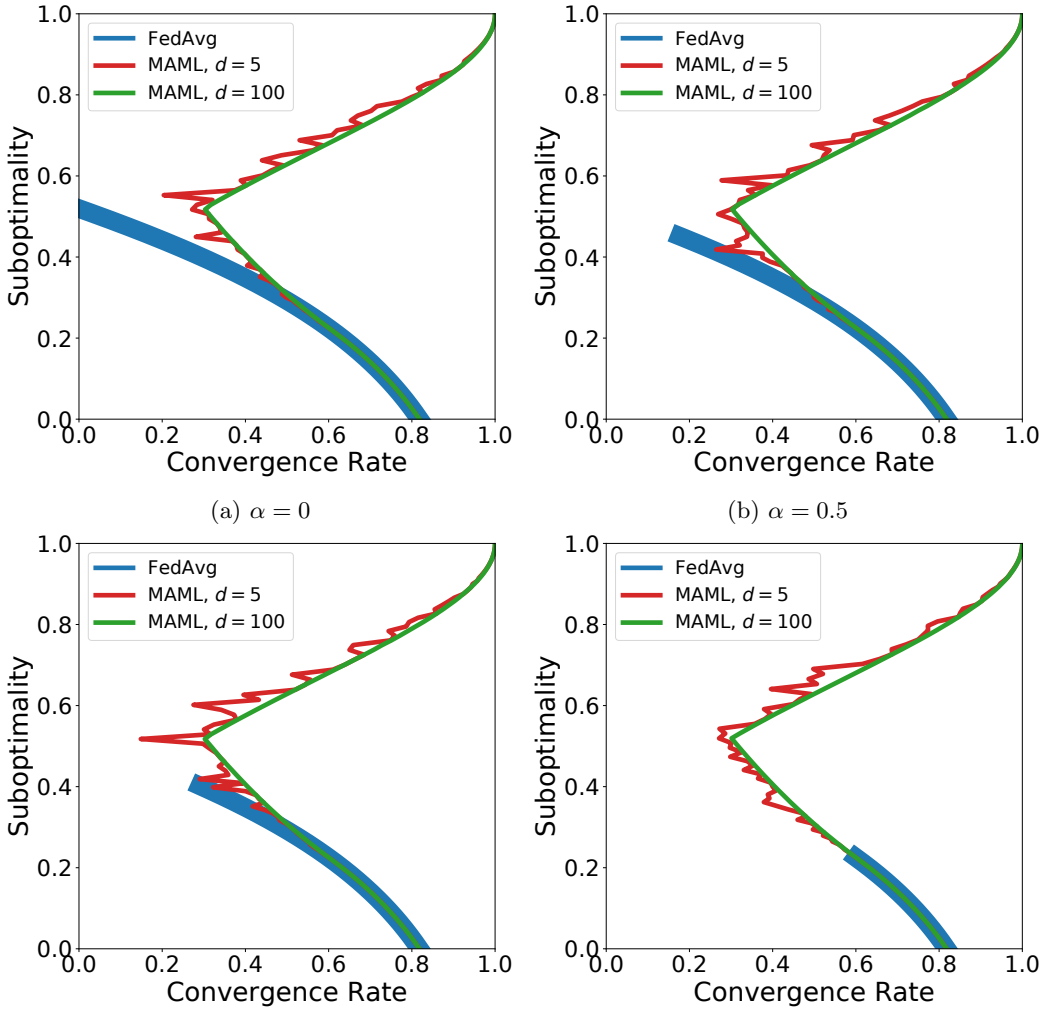


Figure 9: Pareto frontiers for $\mu = 1$, $L = 10, \gamma = 10^{-7}, \Theta = \Theta_K$ and varying α. We generate the frontiers by varying $K \in [1, 10^6]$. We let ServerOpt be gradient descent. For clarity, we used plots of varying width.



(a) $\alpha = 0$ (b) $\alpha = 0.5$

(c) $\alpha = 1.0$ (d) $\alpha = 5.0$

Figure 10: Simulated Pareto frontiers for $\mu = 1, L = 10, \gamma = 0.001, \Theta = \Theta_K$ (MAML), where ServerOpt is gradient descent. We use $\alpha \in \{0, 0.5, 1, 10\}$, $K \in [1, 10^6]$. We randomly generate $A \in \mathbb{R}^{d \times d}$ with $\mu I \preceq A \preceq LI$ and compute the associated (ρ, Δ). We also compare to the Pareto frontier for $\Theta_{1:K}$ (FedAvg) and the same α.

E Experimental Setup

E.1 Datasets and Models

We use three datasets: the federated extended MNIST dataset (FEMNIST) (Caldas et al., 2018), CIFAR-100 (Krizhevsky and Hinton, 2009), and Shakespeare (Caldas et al., 2018). The first two are image datasets, and the third is a language dataset. All datasets are publicly available. We specifically use the versions available in TensorFlow Federated (Ingerman and Ostrowski, 2019), which gives a federated structure to all three. We keep the client partitioning when training, and create a test dataset by taking a union over all test client datasets. Statistics on the number of clients and examples in each dataset are given in Table 3.

Table 3: Dataset statistics.

Dataset	Train Clients	Train Examples	Test Clients	Test Examples
FEMNIST	3,400	671,585	3,400	77,483
CIFAR-100	500	50,000	100	10,000
Shakespeare	715	16,068	715	2,356

FEMNIST The FEMNIST dataset consists of images hand-written alphanumeric characters. There are 62 total alphanumeric characters represented in the dataset. The images are partitioned among clients according to their author. The dataset has natural heterogeneity stemming from the writing style of each person. We train a convolutional network on the dataset (the same one used by Reddi et al. (2020)). The network has two convolutional layers. Each convolutional layer uses 3×3 kernels, max pooling, and then dropout with probability $p = 0.25$. The model has a final dense softmax output layer.

CIFAR-100 The CIFAR-100 dataset is a computer vision dataset consisting of $32 \times 32 \times 3$ images with 100 possible labels. While this dataset does not have a natural partition among clients, a federated version was created by Reddi et al. (2020) using hierarchical latent Dirichlet allocation to enforce moderate amounts of heterogeneity among clients. We train a ResNet-18 on this dataset, where we replace all batch normalization layers with group normalization layers (Wu and He, 2018). The use of group norm over batch norm in federated learning was first advocated by Hsieh et al. (2019).

We perform small amounts of data augmentation and preprocessing, as is standard with CIFAR-100. We first perform a random crop to shape $(24, 24, 3)$, followed by a random horizontal flip. We then normalize the pixel values according to their mean and standard deviation. Thus, given an image x, we compute $(x - \mu)/\sigma$ where μ is the average of the pixel values in x, and σ is the standard deviation.

Shakespeare The Shakespeare dataset is derived from the benchmark designed by Caldas et al. (2018). The dataset corpus is the collected works of William Shakespeare, and the clients correspond to roles in Shakespeare's plays with at least two lines of dialogue. To eliminate confusion, *character* here will refer to alphanumeric and other such symbols, while we will use *client* to denote the various roles in plays. We split each client's lines into sequences of 80 characters, padding if necessary. We use a vocabulary size of 90: 86 characters contained in Shakespeare's work, beginning and end of line tokens, padding tokens, and out-of-vocabulary tokens. We perform next-character prediction on the clients' dialogue using an RNN. The RNN takes as input a sequence of 80 characters, embeds it into a learned 8-dimensional space, and passes the embedding through 2 LSTM layers, each with 256 units. Finally, we use a softmax output layer with 80 units, where we try to predict a sequence of 80 characters formed by shifting the input sequence over by one. Therefore, our output dimension is 80×90. We compute loss using cross-entropy loss.

E.2 Implementation and Hyperparameters

We implement LocalUpdate in TensorFlow Federated (Ingerman and Ostrowski, 2019). We use LocalUpdate with $\Theta = \Theta_{1:K}$ and client learning rate γ. In all experiments, ServerOpt is gradient descent with server learning rate η, with either no momentum, Nesterov momentum, or heavy-ball momentum. We sample $M = 10$ clients

per round. We sample without replacement within a given round, and with replacement across rounds. In order to derive fair comparisons between different hyperparameter settings, we use a random seed to fix which clients are sampled at each round. We use a batch size of $B = 20$ for FEMNIST and CIFAR-100, and $B = 4$ for Shakespeare.

E.3 Details of Figure 6

For posterity's sake, we re-plot Figure 6 in Figure 11. To generate these plots, we perform two distinct experiments. In the first experiment (Figures 6 and 11, left), we fix $\alpha = 0$ and vary ServerOpt. Specifically, we let ServerOpt be gradient descent with no momentum (gradient), gradient descent with Nesterov momentum (nesterov), and gradient descent with heavy-ball momentum (momentum). When ServerOpt uses Nesterov or heavy-ball momentum, we use a momentum parameter of $\beta = 0.9$. In the second experiment (Figures 6 and 11, right), we fix ServerOpt to be gradient descent with no momentum, and vary the proximal strength α. In both cases, we fix $\Theta = \Theta_{1:50}$, and tune γ, η over the range

$$\gamma, \eta \in \{10^{-3}, 10^{-2.5}, \ldots, 10^{0.5}, 10\}.$$

We select the values of γ, η attaining the best average test accuracy over the last 100 rounds.

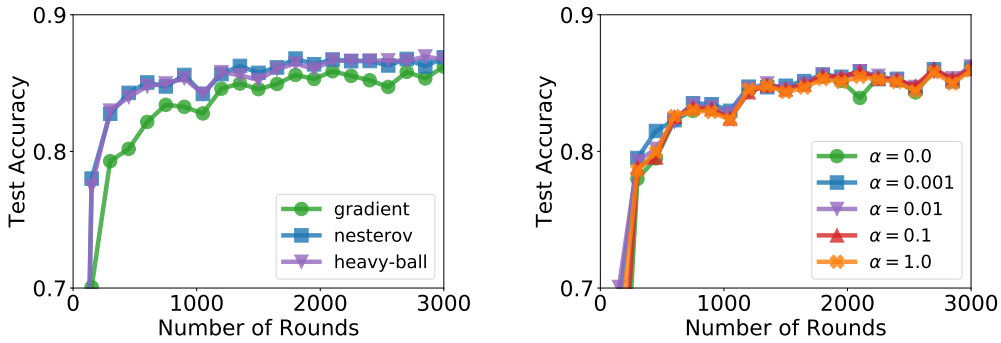


Figure 11: Test accuracy of LocalUpdate on FEMNIST with tuned learning rates. (Left) Varying types of server momentum, $\alpha = 0$. (Right) No momentum and varying α.

F Additional Experiments

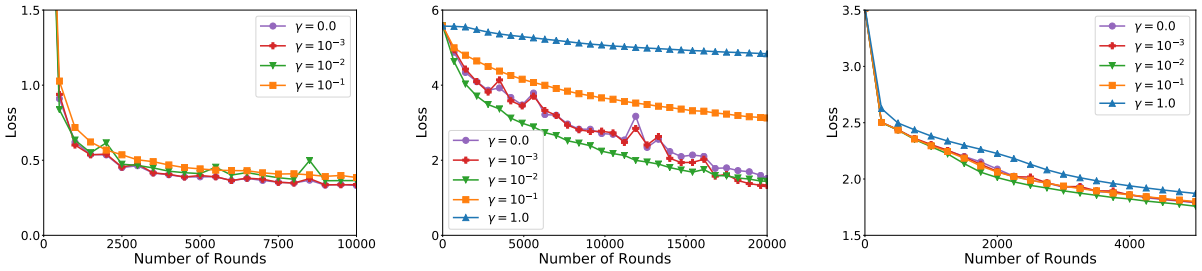
We wish to showcase the convergence-accuracy trade-off discussed in Section 4 in non-convex settings. We train LocalUpdate with $\alpha = 0$, $\Theta = \Theta_{1:10}$, and let ServerOpt be gradient descent with learning rate η. First, we fix $\eta = 0.01$ and vary γ over

$$\gamma \in \{0, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}.$$

We plot the training loss over time on all three datasets in Figure 12, omitting results that diverge due to γ being too large.

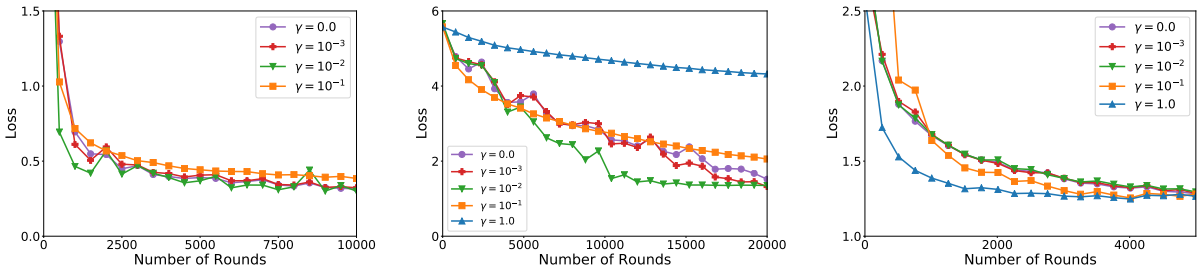
We see that on all three tasks, especially CIFAR-100, the choice of client learning rate can impact not just the speed of convergence, but what point the algorithm converges to. In general, we see very similar behavior to that described in Sections 4 and 5, despite the non-convex loss functions involved in all three tasks. For both FEMNIST and CIFAR-100, smaller client learning rates eventually reach lower training losses than higher learning rates. This is particularly evident in the results for CIFAR-100. While $\gamma = 10^{-2}$ initially performs better than all other methods, it is eventually surpassed by $\gamma = 10^{-3}$, and $\gamma = 0$ ends up obtaining a comparable accuracy. This reflects the idea presented in Section 5 that hyperparameters should be chosen according to the desired convergence-accuracy trade-off. In communication-limited settings, we should use larger γ (or K), while in cases where we can run many communication rounds, we should use smaller γ (or K).

In short, we see clear evidence that the choice of client learning rate γ leads to a trade-off between convergence and accuracy. However, as shown in Lemmas 3 and 4, the condition number of the surrogate loss changes



(a) FEMNIST (b) CIFAR-100 (c) Shakespeare

Figure 12: Training loss of LocalUpdate with $\alpha = 0$, $\Theta = \Theta_{1:10}$, varying client learning rate γ, and where ServerOpt is gradient descent with learning rate $\eta = 0.01$ and no momentum.



(a) FEMNIST (b) CIFAR-100 (c) Shakespeare

Figure 13: Training loss of LocalUpdate with $\alpha = 0, \Theta = \Theta_{1:10}$, varying client learning rate γ, and where ServerOpt is gradient descent with tuned server learning rate η.

depending on parameters such as γ. To derive asymptotically optimal rates for strongly convex functions (such as the ones in Table 2), one must generally set the learning rate η according to the condition number. Thus, we repeat the experiments in Figure 12, but where we tune the server learning rate η instead of fixing it. This helps account for how the optimization dynamics can change as a function of the client learning rate γ. We vary the server learning rate η over

$$\eta \in \{10^{-3}, 10^{-2.5}, \ldots, 10\}$$

and select η that leads to the smallest average training loss over the last 100 rounds. The result is given in Figure 13. Again, we see similar behavior, but see that when the server learning rate is tuned, larger client learning rates may do much better initially. This reflects the fact that in Table 2, the best convergence rates can only be obtained by setting parameters of ServerOpt correctly.

This points to another benefit of the Pareto frontiers proposed in Section 5. Many comparisons of different algorithms, especially empirical ones, can miss good hyperparameter settings. This is heightened by the fact that many FL and ML methods have hyperparameters for both client and server optimizers, comprehensive tuning extremely difficult. This may lead to unfair comparisons between methods. By contrast, the Pareto frontiers showcase convergence-accuracy trade-offs when the hyperparameters of ServerOpt are selected in an "optimal" way, helping derive fair comparisons between methods.