Robust Aggregation for Federated Learning

Krishna Pillutla Sham M. Kakade Zaid Harchaoui
University of Washington

May 24, 2019

Abstract

We introduce a robust aggregation approach to make federated learning robust to settings when a fraction of the devices may be sending outlier updates to the server. The proposed approach relies on a new robust aggregation oracle based on the geometric median, which returns an aggregate using a constant number of calls to a secure average oracle. The robust aggregation oracle is privacy-preserving, similar to the secure average oracle. We present experimental results of the proposed approach with linear models and deep neural networks for computer vision and natural language processing using the LEAF benchmark suite. The robust aggregation approach is agnostic to the level of noise and outperforms the classical aggregation approach in terms of robustness in low signal-to-noise ratio regimes, while being competitive in high signal-to-noise ratio regimes.

1 Introduction

The growing use of smartphones, tablets and wearables has led to an explosion of rich data such as photos captured and text typed by users. It comes with a tremendous potential to improve user experience and power intelligent apps, as well as privacy challenges. Strict regulations require a guarantee that a user's private data is not revealed directly, or indirectly in the form of model updates.

The federated averaging algorithm [43] is a distributed learning algorithm designed for this purpose. Here, each device applies a stochastic learning algorithm on its local data, and a coordinating server updates its model parameters based on an element-wise weighted arithmetic mean of the parameters from each of the devices. A user's private data never leaves their device, and individual parameters are never revealed to the server or other devices with the use of a *secure average oracle* [10].

Robustness is a desirable property for any machine learning algorithm. In the federated setting, robustness turns into an important requirement. We would like to be able to guarantee robustness to departures from nominal distributions of the data, as well as outlier model parameters caused by distributing training to potentially untrustworthy participating devices. For instance, this could be caused by a malfunction in low-cost hardware on mobile devices, corrupted data or adversaries.

Existing federated learning approaches are not robust in the sense that a few outliers can cause divergence of the learning algorithm. The use of such algorithms at an increasing scale for tasks such as keyboard suggestions in mobile devices [58] carries a security threat to the service and its users.

We seek to address this issue by designing a privacy-preserving robust aggregation oracle for federated learning which affords us the same generality and privacy of the federated averaging algorithm in locally running a stochastic learning algorithm on each device, but updates the server parameters using a robust aggregate of individual updates sent by devices.

Owing to the complexity of, and the engineering effort entailed in a practical implementation of a new secure aggregation oracle [see, e.g., 10], we seek a robust aggregation oracle which can be implemented by a small number of calls to the secure average oracle.

Further, the federated learning algorithm with robust aggregation must perform similar to existing approaches in settings of high signal-to-noise ratio (i.e., low noise), while displaying a more graceful degradation of performance as the noise level increases, as illustrated qualitatively in Fig. 1.

1.1 Contributions

The main take-away message of this work is:

> *Federated learning can be made robust to outliers by replacing the weighted arithmetic mean in the aggregation step with an approximate geometric median.*

To this end, we make the following concrete contributions.

Robust Aggregation. We design a practical, privacy-preserving and provably robust aggregation oracle based on the classical geometric median. We show how to implement this aggregation routine in a small number of calls to a secure average oracle with an alternating minimization algorithm which empirically exhibits rapid convergence. This algorithm can be interpreted as a numerically stable version of the classical algorithm of Weiszfeld [57], thus shedding new light on it.

Experiments. We demonstrate the breadth of our framework for different noise models including data corruption and model replacement, on federated learning tasks from computer vision and natural language processing, with linear models as well as convolutional and recurrent neural networks. In particular, our results show that the proposed robust aggregation oracle leads to a federated learning algorithm which, (a) outperforms the federated averaging approach in settings of low signal-to-noise ratio, and (b) matches the performance of the federated averaging algorithm in *thrice the communication cost* in high signal-to-noise ratio settings. Moreover, the aggregation algorithm is completely agnostic to the actual level of noise in the problem instance.

Further, our Python code, as well as the scripts used to generate experimental results are available online[1].

1.2 Related Work

Federated Learning. Federated learning was introduced by McMahan et al. [43] as a distributed learning algorithm to handle on-device training, with a secure aggregation oracle given by Bonawitz et al. [9]. Extensions were proposed in [31, 46, 50, 54]. We address the issue of robustness, which is more broadly applicable in these settings.

Distributed Optimization. Distributed optimization has a long history [see e.g., 7]. Recent work includes a primal-dual framework for distributed convex optimization called COCOA [40, 55] and its decentralized version [20], as well as asynchronous incremental algorithms [34], distributed optimization in networks [51, 52] and distributed mean estimation [30].

Robust Estimation. Robust estimation was pioneered by Huber [24, 25]. Robust median-of-means approaches were introduced in [48], with more recent work in [23, 35, 38, 39, 45]. Robust mean estimation, in particular, received much attention [14, 17, 44]. These works consider the statistics of robust estimation in the i.i.d. case, while we focus on distributed learning with privacy-preservation.

[1] https://github.com/krishnap25/RFA

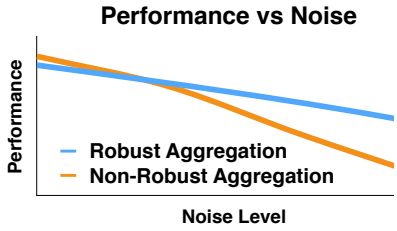


Figure 1: (Qualitative) Ideal behavior of a robust aggregation algorithm compared to its non-robust counterpart.

Algorithm 1 Federated Learning: Meta-Algorithm

Input: F from (1) over K devices, devices per round m

for $t = 1, 2, \cdots$ **do** ▷ Run on the server
  Let S_t denote a random set of m devices
  Broadcast $w^{(t)}$ to each device $k \in S_t$
  for each device $k \in S_t$ **in parallel do**
    $w_k^{(t)} \leftarrow \mathit{LocalUpdate}(k, w^{(t)})$
  $w^{(t+1)} = \mathit{SecureAggregate}(\{w_k^{(t)}\}_{k \in S_t})$

Geometric Median Algorithms. The classical algorithm of Weiszfeld [57] has received much attention [29, 32, 56] - see [4] for a history. However, all these variants are *not* numerically stable, while our variant is (cf. discussion in Sec. 4). The Weiszfeld algorithm is a special case of alternating minimization [3] and majorization-minimization [41, 42]. A landmark theoretical construction led to a nearly-linear time algorithm for the geometric median [16], but its practical applicability is unclear.

Privacy. Differential privacy [18] is a popular framework to guarantee privacy of user data. It is an orthogonal direction to ours, and could be used in conjunction. An alternate approach is homomorphic encryption [19] - see [9] for a discussion on privacy and federated learning.

Byzantine Fault-Tolerance. Byzantine fault-tolerance of a distributed system is resilience to arbitrary, even adversarial behavior of some workers [33]. Recent work includes Byzantine fault-tolerance in gradient based updates [1, 8, 12, 13, 59]. As we discuss in Sec. 3, (a) we consider a more nuanced outlier model which goes beyond the worst-case of the Byzantine setting, and, (b) we aggregate parameters in the federated setting of this work, which is irreconcilably different from aggregating gradients of these related works. Moreover, it is unclear how to implement their aggregation algorithms using only a secure average oracle.

2 Problem Setup

Consider the finite-sum optimization problem

$$\min_{w \in \mathbb{R}^d} \left[F(w) := \frac{1}{n} \sum_{i=1}^{n} f_i(w) \right] . \tag{1}$$

In the setting of federated learning, the functions $f_i : \mathbb{R}^d \to \mathbb{R}$ are distributed across K client devices so that the k^{th} device holds $n_k = |\mathcal{D}_k|$ functions, where $\mathcal{D}_1, \cdots, \mathcal{D}_K$ is an arbitrary partition of $\mathcal{D} = \{f_1, \cdots, f_n\}$. A central server can communicate with each of these devices.

In the supervised learning setting, each function f_i is given by an input-output pair (x_i, y_i) as $f_i(w) = \ell(y_i, \varphi(x_i; w))$, where ℓ is a loss function such as the square loss, and φ maps an input x_i to a prediction using parameters w, for instance, as $\varphi(x_i, w) = x_i^\top w$. The goal of federated learning is to find the best model parameters $w^\star$ in as little communication between the devices and the server as possible, while the cost of local computation on each device is negligible in comparison.

Privacy Constraints. Owing to the privacy-sensitive nature of the data (x_i, y_i), each device is not allowed to directly share any function f_i with the server, but can train locally and share its model parameters only.

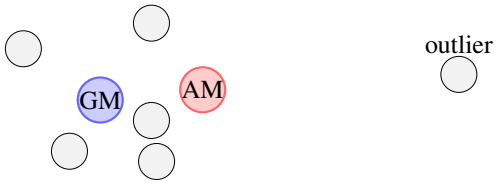


Figure 2: Effect of a single outlier on the geometric median (GM) and arithmetic mean (AM) of the black points in $\mathbb{R}^2$. Not to scale.

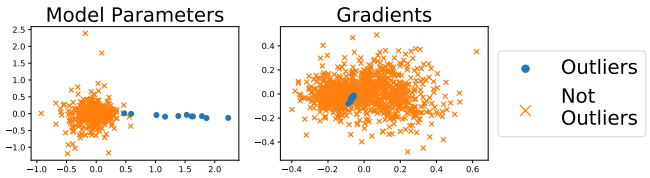


Figure 3: PCA projection on $\mathbb{R}^2$ of model parameters and gradients for a linear model on the EMNIST dataset under data noise at level $\delta = 0.01$ — see Appendix G.4.1 for details.

Furthermore, the server can only access an aggregate of parameters sent by the devices, while individual model parameters of a device are private. We now make this precise.

Definition 1. Given m devices with each device k containing $w_k \in \mathbb{R}^d$ and $\beta_k > 0$, a *secure average oracle* computes the average $\sum_{k=1}^m \beta_k w_k / \sum_{k=1}^m \beta_k$ at a total communication of $\mathcal{O}(d)$ such that no w_k or β_k are revealed to either the server or any other device. Any function $A : (\mathbb{R}^d)^m \to \mathbb{R}^d$ which can be computed by finitely many calls to a secure average oracle is said to be a *secure aggregate*.

An example secure average oracle is the cryptographic protocol of Bonawitz et al. [9].

Federated Learning Meta-Algorithm. We now make precise the general federated learning (FL) meta-algorithm as a distributed algorithm, which runs in synchronized rounds of communication between the server and the devices, consisting in two steps: (a) each device receives the server's model parameters, performs some local computation and sends its update, and, (b) the server updates its parameters based on a secure aggregate (in the sense of Def. 1) of the parameter updates sent by the devices. See Algo. 1.

A concrete FL algorithm must specify (a) the *LocalUpdate* method, the local computation on each device, and, (b) the *SecureAggregate* method, the aggregation strategy of the server. In this framework, the federated averaging (FedAvg) algorithm [43] uses as *LocalUpdate* a few passes of stochastic gradient descent (SGD), while the aggregation strategy in *SecureAggregate* in Line 6 is simply a weighted arithmetic mean with w_k weighted by n_k and requires one call to the secure average oracle.

Overview. In Sec. 3, we design a new FL algorithm by proposing a robust aggregation oracle to use for *SecureAggregate*, while Sec. 4 gives a concrete implementation of the robust aggregation oracle using only a small number of calls to the secure average oracle. Finally, in Sec. 5, we present comprehensive experimental studies comparing the proposed federated learning algorithm to FedAvg and distributed SGD. A rigorous presentation of the material, as well as proofs can be found in the supplementary material.

3 Outlier Updates and Robust Aggregation Oracle

We now define our outlier model and design a robust aggregation oracle using the geometric median.

Beyond the Byzantine Setting. The worst-case assumption of the Byzantine setting [33] that a client device may behave arbitrarily is too stringent for secure aggregation, since concrete implementations of the secure average oracle rely on secret sharing protocols and fundamentally require some trust in the behavior of each device [9]. Therefore, given the widespread practical use of secure aggregation [10], we consider here robustness to a more nuanced outlier model which arises, e.g., under data corruption or model replacement.

Outlier Model. We consider a set $\mathcal{C} \subseteq [K]$ of "corrupted" devices which, unbeknownst to the server, send outlier parameter updates. In particular, in the context of Algo. 1, we assume that the output of *LocalUpdate*(k, w) is an arbitrary vector in $\mathbb{R}^d$ if $k \in \mathcal{C}$, while each device k, including $k \in \mathcal{C}$, behaves in a nominal manner during the run of *SecureAggregate*. Here, we define the noise level δ as $\delta := (1/n) \sum_{k \in \mathcal{C}} n_k$ and the signal-to-noise ratio as $(1-\delta)/\delta$.

The Need for Robust Aggregation. It is a classical fact that the weighted arithmetic mean, as used by FedAvg, is not robust to outliers, where even a single outlier can make the aggregate arbitrarily bad (see Fig. 2). This makes FedAvg sensitive to outliers, especially since privacy requirements do not allow us to inspect any individual update, and motivates the need for robust aggregation strategies.

Aggregating Model Parameters, not Gradients. In contrast to prior work on robust aggregation in distributed learning, which aggregates gradients, we focus on aggregating model parameters because it is more natural in the FL setting. Indeed, aggregating parameters allows us to make more progress at the same communication cost by increasing local computation on each device. In addition, we find aggregation of parameters appealing because assumptions on the distributions of parameters can be easier to interpret than assumptions on distributions of gradients. See Fig. 3 for an illustration.

3.1 Robust Aggregation Oracle based on the Geometric Median

The geometric median (GM) of $w_1, \cdots, w_m \in \mathbb{R}^d$ with weights $\alpha_1, \cdots, \alpha_m > 0$ is the minimizer of

$$g(z) := \sum_{k=1}^{m} \alpha_k \|z - w_k\|, \tag{2}$$

where $\|\cdot\| = \|\cdot\|_2$ is the Euclidean norm. As a robust aggregation oracle, we use an ϵ-approximate minimizer $\widehat{z}$ of g which satisfies $g(\widehat{z}) - \min_z g(z) \leq \epsilon$. We denoted it by $\widehat{z} = \mathrm{GM}\left((w_k)_{k=1}^m, (\alpha_k)_{k=1}^m, \epsilon\right)$.

The GM is known to be robust to arbitrary corruption of points with at most half the total weight [37, Thm. 2.2]. See Fig. 2 for an illustration. We defer to Sec. 4 the actual implementation of robust aggregation oracle using only a secure average oracle.

We assume that $w_1, \cdots w_k$ are non-collinear, which is reasonable in the FL setting. Then, g admits a unique minimizer. Further, we suppose that $\sum_k \alpha_k = 1$ w.l.o.g.

3.2 Robust Federated Aggregation: The RFA Algorithm

We now present RFA in Algo. 2 as an instantiation of the FL meta-algorithm (Algo. 1) using this GM-based robust aggregation oracle. In particular, the local computation *LocalUpdate* is same as in the case of FedAvg, i.e., a few passes of minibatch SGD, while the aggregate is an approximate GM. Note that RFA is completely agnostic to the actual level of noise in the problem. The robust aggregation step enjoys the following robustness guarantee.

Proposition 2. *Consider the setting of Algo. 2 with tolerance $\epsilon \geq 0$. Suppose F from Eq. (1) is L-smooth*[2]*. For any $w^\star \in \arg\min_w F(w)$ and any $S' \subseteq S$ such that $\theta := \sum_{k \in S'} \alpha_k < 1/2$, the output $\widehat{z}$ of the SecureAggregate procedure from Algo. 2 satisfies,*

$$F(\widehat{z}) - F(w^\star) \leq \frac{L}{(1-2\theta)^2} \left(4 \max_{k \in S \setminus S'} \|w_k - w^\star\|^2 + \epsilon^2 \right).$$

[2] A function f is L-smooth if it is continuously differentiable and its gradient ∇f is Lipschitz w.r.t. $\|\cdot\|$.

Algorithm 2 The RFA algorithm

Input: Tolerance ϵ

Instantiate Algo. 1 with
function *SecureAggregate*$(S, \{w_k\}_{k\in S})$
  $\alpha_k \leftarrow n_k / \sum_{j\in S} n_j$ for $k \in S$
  $\widehat{z} \leftarrow \mathrm{GM}\left((w_k)_{k\in S}, (\alpha_k)_{k\in S}, \epsilon\right)$ using Algo. 3 or 10
  return $\widehat{z}$

Algorithm 3 Smoothed Weiszfeld's Algorithm

Input: $w_1, \cdots, w_m \in \mathbb{R}^d$ with w_k on device k, $\alpha_1, \cdots, \alpha_m > 0$, $\nu > 0$, budget T, $z^{(0)} \in \mathbb{R}^d$, secure average oracle $\mathcal{A}$

for $t = 0, 1, \cdots, T-1$ **do**
  Server broadcasts $z^{(t)}$ to devices $1, \cdots, m$
  Device k computes $\beta_k^{(t)} = \frac{\alpha_k}{\nu \vee \|z^{(t)} - w_k\|}$
  $z^{(t+1)} \leftarrow \frac{\sum_{k=1}^m \beta_k^{(t)} w_k}{\sum_{k=1}^m \beta_k^{(t)}}$ using $\mathcal{A}$
return $z^{(T)}$

By letting S' denote the set of outlier devices, we see that the GM is insensitive to outlier updates. Note that Prop. 2 does not require any convexity assumptions. The proof, based on the classical robustness property of the GM [37, Thm. 2.2], is in Appendix B.

3.3 Convergence of RFA

We make two simplifications to analyze the convergence of RFA: (i) we consider linear regression where $f_i(w) = (y_i - x_i^\top w)^2$ for example (x_i, y_i), and, (ii) we suppose that the local data on each device is identical, i.e., $\mathcal{D}_k = \mathcal{D}_{k'}$ for all devices k, k'. In Appendix C, we relax (ii) so that each device draws examples from i.i.d. streams. Note that in the non-i.i.d. case, no convergence analysis is known, even for vanilla FedAvg.

RFA in the Absence of Outliers. We first verify that RFA converges in the absence of outliers — see Appendix C for a proof.

Proposition 3. *Assume that F defined in (1) is μ-strongly convex with unique minimizer $w^\star \in \mathbb{R}^d$ and $\|x_i\| \le R$ for each $i \in [n]$ and that $\mathcal{C} = \varnothing$, i.e., no device sends outlier updates. Then, a variant of RFA (given as Algo. 5 in Appendix C) such that (a) LocalUpdate returns the tail-averaged iterate from N iterations of SGD with constant step-size $\gamma = 1/(2R^2)$ and batch size 1, and, (b) the GM computation is performed using Algo. 3, produces a sequence $(w^{(t)})$ which satisfies*

$$\mathbb{E}F(w^{(t)}) - F(w^\star) \le 2^{-t}\left(F(w^{(0)}) - F(w^\star)\right) + \frac{16 d\sigma^2}{N},$$

where σ^2 quantifies the variance of the stochastic gradients and $N > (8\log 2)(R^2/\mu)\log(R^2/\mu)$.

Remark 4. *The bound in the previous proposition is independent of m. In particular, it shows that aggregation with the geometric median is no worse than running the same algorithm with a single device with $1/m^{th}$ of the total data. The noise term $d\sigma^2/N$ can be improved by increasing the number N_t of SGD iterations in LocalUpdate in the t^{th} outer iteration of RFA — see Prop. 10 in Appendix C. The assumption $\|x\| \le R$ can be relaxed — see Assumption 8 in Appendix C.*

RFA with Outliers. Under the limit of infinite local computation in *LocalUpdate*, we can use Prop. 2 to show that one outer iteration of RFA converges to the solution $w^\star$ despite the outliers, as long as the total weight of the outliers is strictly less than $1/2$. The details are given in Appendix C.3.

4 Implementation of Robust Average Oracle

While the GM is a natural robust aggregation oracle, the key challenge in the federated setting is to design an algorithm to compute it with a small number of calls to a secure average oracle only. We now consider an alternating minimization approach which satisfies this criteria.

Smoothing and Surrogate. Since the GM objective g is non-smooth, we consider the smoothing

$$g_\nu(z) := \sum_{k=1}^m \alpha_k \|z - w_k\|_{(\nu)}\,, \quad \text{where} \quad \|z\|_{(\nu)} := \begin{cases} \frac{1}{2\nu}\|z\|^2 + \frac{\nu}{2}\,, & \|z\| \le \nu \\ \|z\|\,, & \|z\| > \nu \end{cases}. \tag{3}$$

It is known that g_ν is a $(1/\nu)$-smooth approximation to g and that g_ν approximates g to $\nu/2$ [5]. We minimize g_ν with a surrogate $G : \mathbb{R}^d \times \mathbb{R}^m_{++} \to \mathbb{R}$ defined using $\eta = (\eta_1, \cdots, \eta_m) \in \mathbb{R}^m$ as

$$G(z, \eta) := \frac{1}{2}\sum_{k=1}^m \alpha_k \left(\frac{\|z - w_k\|^2}{\eta_k} + \eta_k \right).$$

It is easy to see that G is jointly convex in (z, η) and the following holds (see Appendix E):

$$\min_{(z,\eta)\in\mathbb{R}^d\times\mathcal{E}_\nu} G(z, \eta) = \min_{z\in\mathbb{R}^d} g_\nu(z)\,, \quad \text{where} \quad \mathcal{E}_\nu := \{\eta \in \mathbb{R}^m \,:\, \eta_1, \cdots, \eta_m \ge \nu\}\,.$$

Algorithm. We consider an alternating minimization algorithm in G which iterates

$$\eta^{(t)} = \arg\min_{\eta\in\mathcal{E}_\nu} G(z^{(t)}, \eta)\,, \quad \text{and,} \quad z^{(t+1)} = \arg\min_{z\in\mathbb{R}^d} G(z, \eta^{(t)})\,.$$

Both updates can be computed in closed form — see Algo. 3. Each iteration requires one call to the secure average oracle. We call it the smoothed Weiszfeld algorithm for its resemblance to Weiszfeld [57]'s classical algorithm, which is a special case of Algo. 3 with $\nu = 0$ and $z \ne w_k$ for all k.

Convergence. The algorithm enjoys the following rate of convergence. It is proved in Appendix E.4.

Proposition 5. *The iterate $z^{(t)}$ of Algo. 3 with input $z^{(0)} \in \mathrm{conv}\{w_1, \cdots, w_m\}$ and $\nu > 0$ satisfies*

$$g(z^{(t)}) - g(z^\star) \le \frac{2\|z^{(0)} - z^\star\|^2}{\overline{\nu} t} + \frac{\nu}{2}\,,$$

where $z^\star = \arg\min g$ and $\overline{\nu} = \min_{\tau\in[t],k\in[m]} \nu \vee \|z^{(\tau-1)} - w_k\| \ge \nu$. Furthermore, if $z^\star$ does not coincide with any w_k, and ν satisfies $\nu \le \min_{k=1,\cdots,m}\|z^\star - w_k\|$, then it holds that $g(z^{(t)}) - g(z^\star) \le 2\|z^{(0)} - z^\star\|^2/\overline{\nu} t$.

If we want an ϵ-approximate GM, we set $\nu = \mathcal{O}(\epsilon)$ to get a $\mathcal{O}(1/\epsilon^2)$ rate. However, if the GM $z^\star$ is not too close to any w_k, then the same algorithm automatically enjoys a faster $\mathcal{O}(1/\epsilon)$ rate.

Discussion. Weiszfeld's original algorithm and its variants are numerically unstable when an iterate z approaches some w_k, because of division by $\|z - w_k\|$. This is combated in practice by heuristically using $\nu \vee \|z - w_k\|$ for some small ν, which results exactly in Algo. 3. Here, we use the smooth version g_ν to rigorously and directly analyze the algorithm we run in practice.

Cor. 26 of Prop. 5, given in Appendix E.5, gives a bound on the Weiszfeld algorithm without smoothing. While Prop. 5 proves a global sublinear rate, it is known that the Weiszfeld algorithm exhibits locally linear convergence [29]. Indeed, we find in Fig. 4 (details in Appendix G.4.2) that Algo. 3 displays rapid convergence, giving a high quality solution in *3-5 iterations*. This also obviates the need for acceleration, which is possible in principle [4, 41], and algorithms such as mirror-prox [47] (see Appendix F) which enjoy faster convergence in theory — see Appendix G.4.2 for empirical evidence.

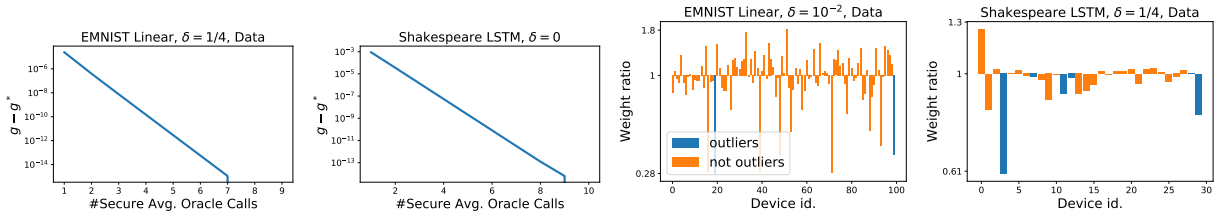


Figure 4: Left: Convergence of the smoothed Weiszfeld algorithm. Right: Visualization of the re-weighting β_k/α_k, where β_k is the weight of w_k in $\mathrm{GM}((w_k), (\alpha_k)) = \sum_k \beta_k w_k$. See Appendix G.4.2 for details.

5 Experiments

We now conduct experimental studies to compare RFA with other federated learning algorithms. The experiments were run using TensorFlow and the data was preprocessed using the LEAF [11]. The full details from this section and more experimental results are given in Appendix G.

5.1 Datasets and Tasks

We consider two tasks with non-i.i.d. data distributions — see Table 1 for a description of the datasets.

Character Recognition. We use the EMNIST dataset [15], where the input is an image of a handwritten character and the output is its identification (0-9, a-z, A-Z). Each device is the set of characters written by the same writer. We use a linear model and a convolutional neural network (ConvNet), both trained with the cross-entropy loss, and evaluated with the classification accuracy.

Character-Level Language Modeling. We learn a character-level language model over the Complete Works of Shakespeare [53]. Given a window of 20 characters, the task is to predict the next character. Each device is a role from a play. We use a long-short term memory model (LSTM) [22], train it with the cross-entropy loss and evaluate with the accuracy of next-character prediction.

5.2 Noise Models

We consider two noise models in *LocalUpdate* of devices $k \in \mathcal{C}$ which send outlier updates.

Data Noise. The data fed into the model is modified, while the training procedure is not. For the EMNIST dataset, we take the negative of an image, while we reverse the text for the Shakespeare dataset. In both cases, the labels are unchanged.

Omniscient Noise. The data is not modified, but the parameters $w_k^{(t)}$ returned by devices $k \in \mathcal{C}$ are modified so that the weighted arithmetic mean of the parameters is set to the negative of what it ought to have been. This is an adversarial noise designed to hurt the weighted arithmetic mean.

5.3 Performance of RFA

We compare RFA with FedAvg and SGD.

Table 1: Dataset description and statistics.

Dataset	Task	#Classes	#Train	#Test	#Devices	#Train per Device		
						Median	Max	Min
EMNIST	Image Classification	62	$204K$	$23K$	1000	160	418	92
Shakespeare	Character-level Language Modeling	53	$2.2M$	$0.25M$	628	1170	70600	90

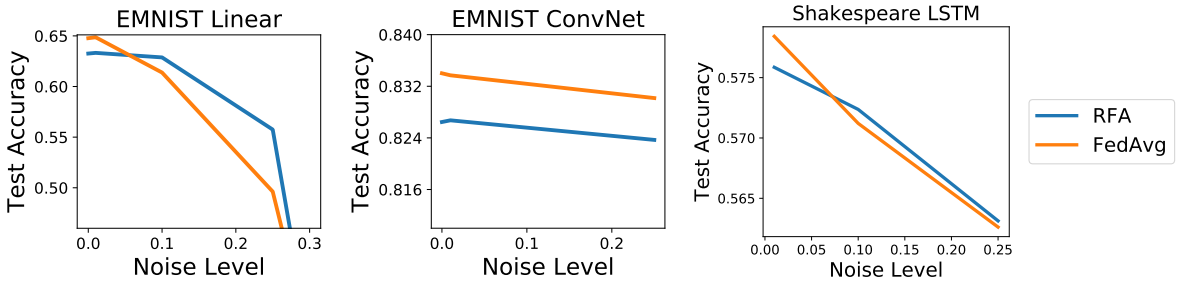


Figure 5: Comparison of robustness of RFA and FedAvg under data noise.

Hyperparameters. The hyperparameters of FedAvg and RFA are chosen similar to the defaults of [43] — see Appendix G.2.3. The learning rate and its decay were tuned on a validation set for FedAvg without any noise and we used the same values for all settings. Further, the aggregation step of RFA is implemented using the smoothed Weiszfeld algorithm with a budget of 3 calls to the secure average oracle, thanks to its rapid empirical convergence; see also Fig. 15 in Appendix G for study of effect of this budget. All experiments are repeated 5 times and the shaded area denotes the minimum and maximum over these runs.

Robustness. Fig. 5 compares the maximum test accuracy of RFA and FedAvg. We observe that RFA gives us improved robustness in case of the linear model, like Fig. 1. FedAvg is better for the ConvNet model, which shows good performance even at high noise levels. On the LSTM model, both FedAvg and RFA give identical curves. We note that the behavior of the training of a neural network when the data is noisy is not well-understood in general; see e.g., [60].

Performance Across Iterations. Next, we plot in Fig. 6 the performance of competing methods versus the number of rounds of communication as measured by the number of calls to the secure average oracle.

We note that FedAvg is better in the high signal-to-noise case of $\delta = 0$ or $\delta = 10^{-2}$ under data noise when measured in the number of calls to the secure average oracle. However, it matches the performance of RFA when measured in terms of the number of outer iterations (cf. Figures 11 to 13 in Appendix G) .

Further, both FedAvg and SGD diverge under omniscient noise (cf. Property 6). An exception, however, is that FedAvg with the LSTM model does not diverge under omniscient noise at $\delta = 10^{-2}$. Indeed, the probability of encountering omniscient noise in an iteration is $\sim 5\%$. Note that RFA still converges with no change in accuracy. Fig. 14 in Appendix G shows experiments with omniscient noise at level $\delta = 0.25$. Furthermore, we observe that RFA is qualitatively more stable than FedAvg in its behavior over time in the high data noise setting of $\delta = 1/4$.

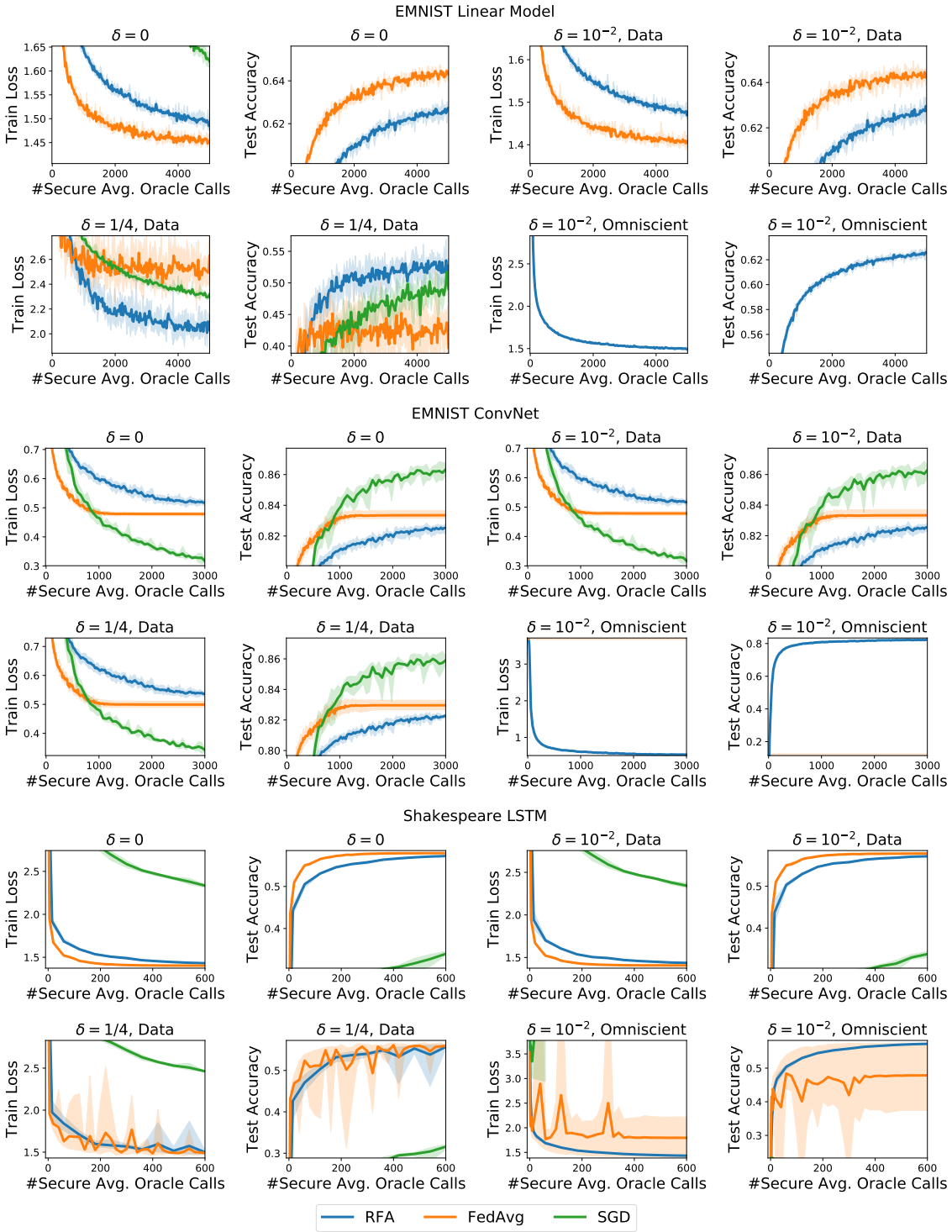


Figure 6: Comparison of methods plotted against number of calls to the secure average oracle for different noise settings. For the case of omniscient noise, FedAvg and SGD are not shown in the plot if they diverge.

References

[1] D. Alistarh, Z. Allen-Zhu, and J. Li. Byzantine stochastic gradient descent. In *Advances in Neural Information Processing Systems 31*, pages 4618–4628, 2018.

[2] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov. How to Backdoor Federated Learning. *arXiv preprint arXiv:1807.00459*, 2018.

[3] A. Beck. On the Convergence of Alternating Minimization for Convex Programming with Applications to Iteratively Reweighted Least Squares and Decomposition Schemes. *SIAM Journal on Optimization*, 25(1):185–209, 2015.

[4] A. Beck and S. Sabach. Weiszfeld's Method: Old and New Results. *J. Optimization Theory and Applications*, 164(1):1–40, 2015.

[5] A. Beck and M. Teboulle. Smoothing and first order methods: A unified framework. *SIAM Journal on Optimization*, 22(2):557–580, 2012.

[6] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[7] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

[8] P. Blanchard, R. Guerraoui, E. M. El Mhamdi, and J. Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems 30*, pages 119–129, 2017.

[9] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[10] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan, et al. Towards Federated Learning at Scale: System Design. *arXiv preprint arXiv:1902.01046*, 2019.

[11] S. Caldas, P. Wu, T. Li, J. Konecný, H. B. McMahan, V. Smith, and A. Talwalkar. LEAF: A benchmark for federated settings. *CoRR*, abs/1812.01097, 2018.

[12] L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos. DRACO: Byzantine-resilient Distributed Training via Redundant Gradients. In *International Conference on Machine Learning*, pages 902–911, 2018.

[13] Y. Chen, L. Su, and J. Xu. Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2): 44, 2017.

[14] Y. Cheng, I. Diakonikolas, and R. Ge. High-Dimensional Robust Mean Estimation in Nearly-Linear Time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2755–2771. SIAM, 2019.

[15] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. EMNIST: an extension of MNIST to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

[16] M. B. Cohen, Y. T. Lee, G. L. Miller, J. Pachocki, and A. Sidford. Geometric median in nearly linear time. In *Symposium on Theory of Computing*, pages 9–21, 2016.

[17] I. Diakonikolas, G. Kamath, D. M. Kane, J. Li, A. Moitra, and A. Stewart. Robust Estimators in High Dimensions without the Computational Intractability. In *Symposium on Foundations of Computer Science*, pages 655–664, 2016.

[18] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*, pages 265–284, 2006.

[19] C. Gentry. Computing arbitrary functions of encrypted data. *Commun. ACM*, 53(3):97–105, 2010.

[20] L. He, A. Bian, and M. Jaggi. COLA: Decentralized Linear Learning. In *Advances in Neural Information Processing Systems 31*, pages 4541–4551, 2018.

[21] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I: Fundamentals*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 1996. ISBN 9783540568506.

[22] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780, 1997.

[23] D. J. Hsu and S. Sabato. Loss minimization and parameter estimation with heavy tails. *Journal of Machine Learning Research*, 17:18:1–18:40, 2016.

[24] P. J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1): 73–101, 03 1964.

[25] P. J. Huber. *Robust Statistics*. Springer, 2011.

[26] P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, V. K. Pillutla, and A. Sidford. A Markov Chain Theory Approach to Characterizing the Minimax Optimality of Stochastic Gradient Descent (for Least Squares). In *Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 2:1–2:10, 2017.

[27] P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford. Parallelizing stochastic gradient descent for least squares regression: Mini-batching, averaging, and model misspecification. *Journal of Machine Learning Research*, 18:223:1–223:42, 2017.

[28] A. Juditsky and A. Nemirovski. First-Order Methods for Nonsmooth Convex Large-Scale Optimization, II: Utilizing Problem's Structure. *Optimization for Machine Learning*, pages 149–183, 2011.

[29] I. N. Katz. Local convergence in Fermat's problem. *Mathematical Programming*, 6(1):89–104, 1974.

[30] J. Konečnỳ and P. Richtárik. Randomized distributed mean estimation: Accuracy vs. communication. *Frontiers in Applied Mathematics and Statistics*, 4:62, 2018.

[31] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated Learning: Strategies for Improving Communication Efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[32] H. W. Kuhn. A note on Fermat's problem. *Mathematical Programming*, 4(1):98–107, Dec 1973.

[33] L. Lamport, R. E. Shostak, and M. C. Pease. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[34] R. Leblond, F. Pedregosa, and S. Lacoste-Julien. Improved Asynchronous Parallel Optimization Analysis for Stochastic Incremental Methods. *Journal of Machine Learning Research*, 19, 2018.

[35] G. Lecué and M. Lerasle. Robust machine learning by median-of-means: theory and practice. *arXiv preprint arXiv:1711.10306*, 2017.

[36] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[37] H. P. Lopuhaa and P. J. Rousseeuw. Breakdown points of affine equivariant estimators of multivariate location and covariance matrices. *Ann. Statist.*, 19(1):229–248, 03 1991.

[38] G. Lugosi and S. Mendelson. Risk minimization by median-of-means tournaments. *arXiv preprint arXiv:1608.00757*, 2016.

[39] G. Lugosi and S. Mendelson. Regularization, sparse recovery, and median-of-means tournaments. *arXiv preprint arXiv:1701.04112*, 2017.

[40] C. Ma, J. Konecný, M. Jaggi, V. Smith, M. I. Jordan, P. Richtárik, and M. Takác. Distributed optimization with arbitrary local solvers. *Optimization Methods and Software*, 32(4):813–848, 2017.

[41] J. Mairal. Optimization with First-Order Surrogate Functions. In *International Conference on Machine Learning*, pages 783–791, 2013.

[42] J. Mairal. Incremental Majorization-Minimization Optimization with Application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.

[43] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

[44] S. Minsker. Uniform Bounds for Robust Mean Estimators. *arXiv preprint arXiv:1812.03523*, 2018.

[45] S. Minsker et al. Geometric median and robust estimation in Banach spaces. *Bernoulli*, 21(4):2308–2335, 2015.

[46] M. Mohri, G. Sivek, and A. T. Suresh. Agnostic Federated Learning. *arXiv preprint arXiv:1902.00146*, 2019.

[47] A. Nemirovski. Prox-Method with Rate of Convergence $O(1/T)$ for Variational Inequalities with Lipschitz Continuous Monotone Operators and Smooth Convex-Concave Saddle Point Problems. *SIAM J. on Optimization*, 15(1):229–251, 2005. ISSN 1052-6234.

[48] A. S. Nemirovski and D. B. Yudin. Problem Complexity and Method Efficiency in Optimization. 1983.

[49] Y. Nesterov. *Introductory Lectures on Convex Optimization Vol. I: Basic course*, volume 87. Springer Science & Business Media, 2013.

[50] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith. On the Convergence of Federated Optimization in Heterogeneous Networks. *arXiv preprint arXiv:1812.06127*, 2018.

[51] K. Scaman, F. R. Bach, S. Bubeck, Y. T. Lee, and L. Massoulié. Optimal Algorithms for Smooth and Strongly Convex Distributed Optimization in networks. In *International Conference on Machine Learning*, pages 3027–3036, 2017.

[52] K. Scaman, F. Bach, S. Bubeck, L. Massoulié, and Y. T. Lee. Optimal Algorithms for Non-Smooth Distributed Optimization in Networks. In *Advances in Neural Information Processing Systems 31*, pages 2745–2754, 2018.

[53] W. Shakespeare. The Complete Works of William Shakespeare. URL https://www.gutenberg.org/ebooks/100.

[54] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems 30*, pages 4424–4434, 2017.

[55] V. Smith, S. Forte, M. Chenxin, M. Takáč, M. I. Jordan, and M. Jaggi. COCOA: A General Framework for Communication-Efficient Distributed Optimization. *Journal of Machine Learning Research*, 18:230, 2018.

[56] Y. Vardi and C.-H. Zhang. A modified Weiszfeld algorithm for the Fermat-Weber location problem. *Mathematical Programming*, 90(3):559–566, 2001.

[57] E. Weiszfeld. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal, First Series*, 43:355–386, 1937.

[58] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays. Applied Federated Learning: Improving Google Keyboard Query Suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

[59] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5636–5645, 2018.

[60] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

Supplementary Material:
Robust Aggregation for Federated Learning

Table of Contents

A Mathematical Description of Noise Model

Here, we given a mathematical description of the federated learning (FL) meta-algorithm as well as the outlier model.

A.1 Federated Learning Meta-Algorithm

We describe the components of Algo. 1.

Local Computation. In the noise-free setting, the local computation Φ_k on device k is modeled as $\Phi_k(w, H) := \Phi(w, H; \mathcal{D}_k) \in \mathbb{R}^d$, where $w \in \mathbb{R}^d$ are the model parameters sent by the server, H represents other information sent by the server, e.g., a random seed or hyperparameters such as the learning rate, and, Φ is a deterministic function depending only on the local data $\mathcal{D}_k$[3].

Aggregation. In each step t, the server weighs the parameters sent by the selected devices S_t with a probability distribution p_t given by $(p_t)_k \propto n_k$. The aggregation must be secure in the sense of Def. 1. Note that FedAvg aggregates using one call to a secure average oracle as $w^{(t+1)} = \mathbb{E}_{k\sim p_t}\left[\Phi_k(w^{(t)}, H_k)\right]$.

A.2 Outliers and Signal-to-Noise Ratio

We allow a set $\mathcal{C} \subseteq [K]$ of "corrupted" devices to, unbeknownst to the server, send outlier parameter updates for instance, due to malfunction or malintent. Formally, we modify the local computation model as

$$\Phi_k(w, H) = \begin{cases} \Phi(w, H, \mathcal{D}_k)\,, & k \notin \mathcal{C} \\ \Psi(k, w, H, \mathcal{D}_k, \mathcal{D}, \mathcal{S}) & k \in \mathcal{C}\,, \end{cases}$$

where Ψ is an arbitrary $\mathbb{R}^d$-valued function which is allowed to depend on the entire dataset, as well as the server state $\mathcal{S}$. Recall that we define the noise level δ as

$$\delta := \frac{1}{n}\sum_{k\in\mathcal{C}} n_k\,,$$

and the signal-to-noise ratio as $(1-\delta)/\delta$.

B Robust Aggregation: Missing Details

Here, we provide the missing details from Sec. 3.

B.1 Failure of the Arithmetic Mean

We recall the classical fact that the weighted arithmetic mean as used by FedAvg can be made arbitrarily bad by a single corruption.

Property 6. *Fix $w_2, \cdots, w_m \in \mathbb{R}^d$, $\alpha_1, \cdots, \alpha_m > 0$ such that $\sum_k \alpha_k = 1$ and consider any $\overline{w} \in \mathbb{R}^d$. Then, it is possible to choose $w_1 \in \mathbb{R}^d$ so that $\sum_{k=1}^m \alpha_k w_k = \overline{w}$.*

Proof. Choose $w_1 = (\overline{w} - \sum_{k=2}^m \alpha_k w_k)/\alpha_1$. □

[3]While SGD is a random process, it is deterministic from a procedural viewpoint if the pseudo-random generator uses a fixed seed.

B.2 Robustness of the Geometric Median

The geometric median (GM) is known to be robust to arbitrary corruption of point with at most half the total weight. In particular, Lopuhaa and Rousseeuw [37, Thm. 2.2] show that if a large fraction of the w_k's are close to any $z \in \mathbb{R}^d$, then the GM cannot be far from z. In other words, the small fraction of w_k's far from this z cannot arbitrarily affect the GM. This is stated below.

Property 7. *Fix $\epsilon \geq 0$. Suppose $\widehat{z}$ satisfies $g(\widehat{z}) \leq \min_z g(z) + \epsilon$. Then, for any $S \subset [m]$ such that $\theta := \sum_{k \in S} \alpha_k < 1/2$ and any $z^\star \in \mathbb{R}^d$, we have that*

$$\|\widehat{z} - z^\star\| \leq 2\left(\frac{1-\theta}{1-2\theta}\right) \max_{k \notin S} \|w_k - z^\star\| + \frac{\epsilon}{1-2\theta}\,.$$

For example, if we let S be the set of outliers and $z^\star$ be the mean of the non-outliers, then Property 7 bounds how far the GM can be from this $z^\star$.

B.3 The RFA Algorithm

The full version of RFA is given in Algo. 4. The local computation $\Phi_k(w, H)$ is same as in the case of FedAvg, i.e., a few passes of minibatch SGD, while the aggregation step finds an approximate GM of the updates it receives.

We now prove Prop. 2.

Proposition 2. *Consider the setting of Algo. 4 with input $\epsilon \geq 0$. Suppose F from Eq. (1) is L-smooth. For any $w^\star \in \arg\min_w F(w)$ and any $S \subseteq S_t$ such that $\theta := \sum_{k \in S} \alpha_k^{(t)} < 1/2$, we have,*

$$F(w^{(t+1)}) - F(w^\star) \leq \frac{L}{(1-2\theta)^2}\left(4 \max_{k \in S_t \setminus S} \left\|w_k^{(t+1)} - w^\star\right\|^2 + \epsilon^2\right)\,.$$

Proof. By smoothness [e.g., 49, Thm. 2.1.5],

$$F(w^{(t+1)}) - F(w^\star) \leq (L/2)\|w^{(t+1)} - w^\star\|^2\,.$$

Now, plug in Property 7 with $z = w^\star$ and use $(a+b)^2 \leq 2(a^2+b^2)$ for reals a, b. □

C Convergence Analysis of RFA for Linear Regression

In this section, we consider the case of linear regression with i.i.d. data first in the absence of outliers, and then with outliers. We show that the RFA algorithm does indeed converge as expected in this case.

C.1 Setting

We now define the setting in the absence of outliers. Consider $F : \mathbb{R}^d \to \mathbb{R}$ defined as

$$F(w) = \frac{1}{2}\mathbb{E}_{(x,y)\sim\mathbb{P}}(y - w^\top x)^2\,. \tag{4}$$

where $\mathbb{P}$ is a fixed distribution on $\mathbb{R}^d \times \mathbb{R}$. We let $\mathbb{P}_X$ denote the marginal distribution on $\mathbb{R}^d$ and $\mathbb{P}(\cdot|x)$ denote the conditional distribution of $\mathbb{R}$ given $x \in \mathbb{R}^d$.

In the federated setting, we suppose that each device has access to independent samples from $\mathbb{P}$, and the overall goal is to minimize F. Note that Problem (1) is a special case of this setting where the distribution $\mathbb{P}$ is a discrete distribution on n points.

Algorithm 4 The RFA algorithm

Input: Function F as in Eq. (1) distributed over K devices, $\epsilon \geq 0$, fraction c, minibatch size b, number of local epochs n_e, learning rate sequence (γ_t), initial iterate $w^{(0)}$, secure average oracle $\mathcal{A}$

Server executes:

for $t = 1, 2, \cdots$ **do**
  $m \leftarrow \max\{cK, 1\}$
  Let S_t denote a random set of m devices
  Broadcast $w^{(t)}$ to each device $k \in S_t$
  for each device $k \in S_t$ **in parallel do**
    $w_k^{(t)} \leftarrow \textit{LocalUpdate}(k, w^{(t)})$
  $\alpha_k^{(t)} \leftarrow n_k / \sum_{j \in S_t} n_j$ for $k \in S_t$
  $w^{(t+1)} \leftarrow \mathrm{GM}\left((w_k^{(t)})_{k \in S_t}, (\alpha_k^{(t)})_{k \in S_t}, \epsilon\right)$ using Algo. 3 or Algo. 10

function *LocalUpdate*(k, w) ▷ Run on device k
  for $i = 1, \cdots, n_k n_e / b$ **do**
    $B_i \leftarrow$ random subset of $\mathcal{D}_k$ of size b
    $w \leftarrow w - \gamma_t \sum_{f \in B_i} \nabla f(w) / b$
  return w

Assumptions. We make the following assumptions. The first assumption is about the marginal $\mathbb{P}_X$.

Assumption 8. For some $R > 0$, it holds that[4] $\mathbb{E}_{x \sim \mathbb{P}_X}\left[\|x\|^2 x x^\top\right] \preceq R^2 \mathbb{E}[x x^\top]$. In particular, this holds if we have $\mathbb{P}_X$-almost surely that $\|x\| \leq R$. Furthermore, the smallest eigenvalue $\mu = \lambda_{\min}\left(\mathbb{E}_{x \sim \mathbb{P}_X}[x x^\top]\right)$ of the hessian $\mathbb{E}_{x \sim \mathbb{P}_X}[x x^\top]$ of F is strictly positive.

The first part of Assumption 8 is standard in linear regression [see e.g., 27], while the second part means that F is μ-strongly convex. We denote a condition number κ as

$$\kappa := \frac{R^2}{\mu} . \tag{5}$$

The next assumption is about the conditional $\mathbb{P}(\cdot|x)$.

Assumption 9. There exists a vector $w^\star \in \mathbb{R}^d$ such that for all $x \in \mathbb{R}^d$, the random variable y_x distributed according to the law $\mathbb{P}(\cdot|x)$ satisfies $y_x = x^\top w^\star + \zeta$, where ζ is a random variable satisfying (a) ζ is independent of x, (b) $\mathbb{E}\zeta = 0$, and, (c) $\mathbb{E}\zeta^2 \leq \sigma^2$, for some $\sigma^2 < \infty$.

Under Assumptions 8 and 9, $w^\star$ is the unique minimizer of $F(w)$. The independent of ζ and x from Assumption 9 implies a well-specified model. The results presented in this section continue to hold with suitable modifications in the case of model misspecification. See [26, 27] for a discussion.

C.2 Convergence in the Absence of Outliers

We now show the convergence of federated learning with tail-averaged SGD as *LocalUpdate* in Algo. 1, while the aggregation strategy is only required to return a point in $\mathrm{conv}\{w_1, \cdots, w_m\}$. In this setting, we assume that we have no outliers. The overall algorithm is presented in Algo. 5.

[4] For symmetric matrices $A, B \in \mathbb{R}^{d \times d}$, the notation $A \preceq B$, means that $B - A$ is positive semidefinite.

Proposition 10. *Consider F from (4) and suppose that Assumptions 8 and 9 hold. Suppose the aggregation strategy SecureAggregate satisfies that SecureAggregate $\left((w_k)_{k\in[m]}, (\alpha_k)_{k\in[m]}\right) \in \operatorname{conv}\{w_1, \cdots, w_m\}$ for all $w_1, \cdots, w_m \in \mathbb{R}^d$ and $\alpha_1, \cdots, \alpha_m > 0$ with $\sum_{k=1}^m \alpha_k = 1$, this being true if*

- *SecureAggregate $\left((w_k)_{k\in[m]}, (\alpha_k)_{k\in[m]}\right) = \mathrm{GM}\left((w_k)_{k\in[m]}, (\alpha_k)_{k\in[m]}, 0\right)$, or,*
- *SecureAggregate $\left((w_k)_{k\in[m]}, (\alpha_k)_{k\in[m]}\right) = \mathrm{GM}\left((w_k)_{k\in[m]}, (\alpha_k)_{k\in[m]}, \epsilon\right)$ which is computed using the Smoothed Weiszfeld algorithm (Algo. 3) for any $\epsilon > 0$.*

In addition, suppose that $N > (8\log 2)\kappa\log\kappa$, where $\kappa = R^2/\mu$. Then, the output $w^{(T)}$ of Algo. 5 with learning rate $\gamma = 1/(2R^2)$ and number of SGD iterations $N_t = N$ for all t satisifies,

$$\mathbb{E}F(w^{(T)}) - F(w^\star) \le 2^{-T}\left(F(w^{(0)}) - F(w^\star)\right) + \frac{16d\sigma^2}{N}\,.$$

Instead, if the number of SGD iterations is chosen as $N_t = 2^t N$, then we have that,

$$\mathbb{E}F(w^{(T)}) - F(w^\star) \le 2^{-T}\left(F(w^{(0)}) - F(w^\star)\right) + \frac{8d\sigma^2 T}{2^{T-1}N}\,.$$

Proof. Let $\mathcal{F}_t$ be sigma algebra generated by $w^{(t)}$. Denote $w_k^{(t)} = \overline{z}_{N_t':N_t}$ with $N_t' = N_t/2$ for each selected device k, as obtained from Line 12 of Algo. 5. Under Assumptions 8 and 9, the output of *LocalUpdate*$(k, w^{(t)}, N_t)$ (tail-averaged SGD with batch size 1) of Algo. 5 satisfies the following guarantee, cf. [26, Thm. 1] or [27, Cor. 2]:

$$\mathbb{E}\left[F(w_k^{(t)})\middle|\mathcal{F}_t\right] - F(w^\star) \le 2\exp\left(-\frac{N_t}{4\kappa\log\kappa}\right)\left(F(w^{(t)}) - F(w^\star)\right) + \frac{4d\sigma^2}{N_t}\,.$$

Since $N_t > (8\log 2)\kappa\log\kappa$ for all t, we have that

$$\mathbb{E}\left[F(w_k^{(t)})\middle|\mathcal{F}_t\right] - F(w^\star) \le \frac{1}{2}\left(F(w^{(t)}) - F(w^\star)\right) + \frac{4d\sigma^2}{N_t}\,. \tag{6}$$

Since the aggregation strategy selects a point in the convex hull of $\{w_1^{(t)}, \cdots, w_k^{(t)}\}$, it follows from convexity of F that

$$F(w^{(t+1)}) \le F(w_k^{(t)})\,, \tag{7}$$

for each selected device k. We now take an expectation of (6) over $\mathcal{F}_t$ and use (7) to get

$$\mathbb{E}F(w^{(t+1)}) - F(w^\star) \le \frac{1}{2}\left(\mathbb{E}F(w^{(t)}) - F(w^\star)\right) + \frac{4d\sigma^2}{N_t}\,.$$

We unroll this sum over t to get

$$\mathbb{E}F(w^{(T)}) - F(w^\star) \le 2^{-T}\left(F(w^{(0)}) - F(w^\star)\right) + 4d\sigma^2\left(\frac{1}{N_{T-1}} + \frac{1}{2N_{T-2}} + \cdots + \frac{1}{2^{T-1}N_0}\right)\,.$$

In the case that $N_t = N$, for all t, using $1 + 1/2 + \cdots + 1/2^{T-1} \le 2$ completes the first part. In the case that $N_t = 2^t N$, the second term sums to $4d\sigma^2 T/(2^{T-1}N)$. □

Algorithm 5 Federated Learning: i.i.d. Case for Linear Regression

Input: F from (4) in K devices, initial iterate $w^{(0)}$, devices per round m, number of rounds T, sequence (N_t) of device SGD iterations per round, aggregation strategy *SecureAggregate*

Server executes:

for $t = 0, 1, \cdots, T-1$ **do**
  Let S_t denote a random set of m devices
  Broadcast $w^{(t)}$ to each device $k \in S_t$
  for each device $k \in S_t$ **in parallel do**
    $w_k^{(t)} \leftarrow$ *LocalUpdate*$(k, w^{(t)}, N_t)$ on device k
  $w^{(t+1)} =$ *SecureAggregate* $\left((w_k^{(t)})_{k \in S_t}, (n_k/n^{(t)})_{k \in S_t}\right)$, where $n^{(t)} = \sum_{j \in S_t} n_j$

function *LocalUpdate*(k, w, N) ▷ Run on device k
  Set $z_0 \leftarrow w$
  for $\tau = 0, 1, \cdots, N-1$ **do**
    Sample $(x_\tau, y_\tau) \sim \mathbb{P}$
    Update $z_{\tau+1} \leftarrow z_\tau - \gamma x_\tau (x_\tau^\top z_\tau - y_\tau)$
  return $\overline{z}_{N':N} = \frac{1}{N-N'} \sum_{\tau=N'+1}^{N} z_\tau$, where $N' = N/2$

Discussion. The bound in the previous proposition is independent of m. In particular, it shows that aggregation in the convex hull is no worse than running the same algorithm with a single device with $1/m^{\text{th}}$ of the total data. If the model were misspecified (i.e., Assumption 9 did not hold), the same result can be obtained with using a smaller learning rate and larger number n of local iterations — see [27].

Furthermore, the analysis is not tight in the sense that every iteration of RFA adds to the contribution of the noise variance. The goal of this bound was to show as a sanity check that RFA converges. We leave a refined analysis of RFA (and indeed of FedAvg as well) for future work.

C.3 Convergence in the Presence of Outliers

A general convergence proof in the presence of outliers is complicated by the fact that we do not have a tight analysis of FedAvg even in the case of linear regression. For this reason, we present here a proof of convergence of a single outer iteration of RFA under the limit of infinite local computation.

C.3.1 Setup

We specialize the notation of Sec. C.1 to consider a single iteration of the RFA algorithm.

Consider m devices $k = 1, \cdots, m$, each of which has access to independent samples from $\mathbb{P}$. Let $\alpha_k \propto n_k$ be the weight of device k, as set in Algo. 2. Suppose now that a set $S' \subseteq [m]$ returns outlier updates. Define θ to be the total weight of the outliers as

$$\theta := \sum_{k \in S'} \alpha_k \,.$$

C.3.2 Convergence Result

For convenience, a single iteration of RFA is given in Algo. 6. It satisfies the following convergence in probability upto a tolerance ϵ of the approximate geometric median computation.

Proposition 11. *Consider F from (4) and suppose that Assumptions 8 and 9 hold. Further, assume that the total weight of the outliers θ satisfies $\theta < 1/2$. In this case, the output $w^{(1)}(N)$ of Algo. 6 with N iterations of device SGD satisfies*

$$\lim_{N\to\infty} \mathbb{P}\left[F(w^{(1)}(N)) - F(w^\star) > \frac{L\epsilon^2}{(1-2\theta)^2}\right] = 0\,,$$

where $L = \lambda_{\max}(\mathbb{E}_{x\sim\mathbb{P}_X}[xx^\top])$ is the smoothness of F.

Proof. Let N be the number of iterations of SGD. We denote $w_k(N)$ to be output of *LocalUpdate* on device k after N iterations of tail-averaged SGD. We start analogously as in the proof of Prop. 10 to get for each non-outlier device $k \in S \setminus S'$ that

$$\mathbb{E}\left[F(w_k(N))\right] - F(w^\star) \leq 2\exp\left(-\frac{N}{4\kappa\log\kappa}\right)\left(F(w^{(0)}) - F(w^\star)\right) + \frac{4d\sigma^2}{N}\,.$$

For some $\delta'_N > 0$, let $\mathcal{E}_N$ denote the event that for every $k \in S \setminus S'$, it holds that

$$F(w_k(N)) - F(w^\star) \leq \frac{m}{\delta'_N}\left(2\exp\left(-\frac{N}{4\kappa\log\kappa}\right)\left(F(w^{(0)}) - F(w^\star)\right) + \frac{4d\sigma^2}{N}\right)\,.$$

Using Markov's inequality and a union bound over each of the $\leq m$ devices in $S \setminus S'$, we get that $\mathcal{E}_N$ holds with probability at least $1 - \delta'_N$.

From μ-strong convexity of F, whenever $\mathcal{E}_N$ holds, it follows for every $k \in S \setminus S'$ that

$$\|w_k(N) - w^\star\|^2 \leq \frac{2m}{\mu\delta'_N}\left(2\exp\left(-\frac{N}{4\kappa\log\kappa}\right)\left(F(w^{(0)}) - F(w^\star)\right) + \frac{4d\sigma^2}{N}\right)\,.$$

Since $\theta < 1/2$, we invoke Prop. 2 to assert that whenever $\mathcal{E}_N$ holds, we can bound the function value of the aggregation step as

$$F(w^{(1)}(N)) - F(w^\star) \leq \frac{8mL}{\mu\delta'_N(1-2\theta)^2}\left(2\exp\left(-\frac{N}{4\kappa\log\kappa}\right)\left(F(w^{(0)}) - F(w^\star)\right) + \frac{4d\sigma^2}{N}\right) + \frac{L\epsilon^2}{(1-2\theta)^2}\,,$$

where $w^{(1)}(N)$ is the output of Algo. 6 where N denotes the number of local iterations of SGD on each device k. Setting $\delta'_N = C/\sqrt{N}$, where C is a universal constant completes the proof. □

Remark 12. *Note that we can generalize the analysis of this section to the setting where the number of devices $m = m_N \to \infty$, the number of outliers $|S'_N| \to \infty$, such that $\theta_N = \sum_{k\in S'_N}\alpha_k < 1/2$ strictly. The bound in Prop. 11 continues to hold as long as $m_N/N \to 0$, i.e., $m = o(N)$.*

D Geometric Median: Smoothing, Setup and Notation

We are given distinct points $w_1, \cdots, w_m \in \mathbb{R}^d$ and scalars $\alpha_1, \cdots, \alpha_m > 0$ such that $\sum_{k=1}^m \alpha_k = 1$. We make the following non-degenerateness assumption, which is assumed to hold throughout this work.

Assumption 13. The points $w_1, \cdots, w_k$ are not collinear.

Algorithm 6 Single iteration of RFA (i.i.d. Case for Linear Regression)

Input: F from (4) in K devices, initial iterate $w^{(0)}$, set S of selected devices such that $|S| = m$, number N of device SGD iterations, tolerance ϵ for the geometric median computation

Server executes:

1: Broadcast $w^{(0)}$ to each device $k \in S$

2: **for** each device $k \in S$ **in parallel do**

3: $w_k \leftarrow$ *LocalUpdate*$(k, w^{(0)}, N)$ on device k, where *LocalUpdate* is defined in Algo. 5

4: Using $\widehat{n} = \sum_{j \in S} n_j$, set

$$w^{(1)} = \mathrm{GM}\left((w_k)_{k \in S}, (n_k/\widehat{n})_{k \in S}, \epsilon\right) ,$$

5: **return** $w^{(1)}$

This assumption is reasonable in the federated learning setting we consider.

The geometric median is defined as any minimizer of

$$g(z) := \sum_{k=1}^{m} \alpha_k \|z - w_k\| . \tag{8}$$

Here, $\|\cdot\|$ refers to the Euclidean norm unless mentioned otherwise.

Under Assumption 13, g is known to have a unique minimizer - we denote it by $z^\star$.

Given a smoothing parameter $\nu > 0$, its smoothed variant g_ν is

$$g_\nu(z) := \sum_{k=1}^{m} \alpha_k \|z - w_k\|_{(\nu)} , \tag{9}$$

where

$$\|z\|_{(\nu)} := \max_{u^\top u \leq 1} \left\{ u^\top z - \tfrac{\nu}{2} u^\top u \right\} + \tfrac{\nu}{2} = \begin{cases} \frac{1}{2\nu}\|z\|^2 + \frac{\nu}{2} , & \|z\| \leq \nu \\ \|z\| , & \|z\| > \nu \end{cases} . \tag{10}$$

In case $\nu = 0$, we define $g_0 \equiv g$. Beck and Teboulle [5] show that $\|\cdot\|_{(\nu)}$ is $(1/\nu)$-smooth and that

$$0 \leq \|\cdot\|_{(\nu)} - \|\cdot\| \leq \nu/2 \tag{11}$$

Under Assumption 13, g_ν has a unique minimizer as well, denoted by $z_\nu^\star$. We call $z_\nu^\star$ as the ν-smoothed geometric median.

We let R denote the diameter of the convex hull of $\{w_1, \cdots, w_m\}$, i.e.,

$$R := \mathrm{diam}(\mathrm{conv}\{w_1, \cdots, w_m\}) = \max_{z, z' \in \mathrm{conv}\{w_1, \cdots, w_m\}} \|z - z'\| . \tag{12}$$

We also assume that $\nu < R$, since for all $\nu \geq R$, the function g_ν is simply a quadratic for all $z \in \mathrm{conv}\{w_1, \cdots, w_m\}$.

E The Smoothed Weiszfeld Algorithm

In this section, we elaborate on the smoothed Weiszfeld's algorithm. The notation used is given in Appendix D.

Algorithm 7 The Smoothed Weiszfeld Algorithm

Input: $w_1, \cdots, w_m \in \mathbb{R}^d$, $\alpha_1, \cdots, \alpha_m > 0$ with $\sum_{k=1}^m \alpha_k = 1$, $\nu > 0$, Number of iterations T, $z^{(0)} \in \operatorname{conv}\{w_1, \cdots, w_m\}$.

1: **for** $t = 0, 1, \cdots, T-1$ **do**

2: Set $\eta_k^{(t)} = \max\left\{\nu, \|z^{(t)} - w_k\|\right\}$ and $\beta_k^{(t)} = \alpha_k / \eta_k^{(t)}$ for $k = 1, \cdots, m$.

3: Set $z^{(t+1)} = \left(\sum_{k=1}^m \beta_k^{(t)} w_k\right) / \left(\sum_{k=1}^m \beta_k^{(t)}\right)$.

Output: $z^{(T)}$.

E.1 Weiszfeld Algorithm: Review

The Weiszfeld [57] algorithm performs the iterations

$$z^{(t+1)} = \begin{cases} \left(\sum_{k=1}^m \beta_k^{(t)} w_k\right) / \left(\sum_{k=1}^m \beta_k^{(t)}\right), & \text{if } z^{(t)} \notin \{w_1, \cdots, w_k\}, \\ w_k, & \text{if } z^{(t)} = w_k \text{ for some } k, \end{cases} \tag{13}$$

where $\beta_k^{(t)} = \alpha_k / \|z^{(t)} - w_k\|$. Kuhn [32, Thm. 3.4] showed that the sequence $\left(z^{(t)}\right)_{t=0}^{\infty}$ converges to the minimizer of g from (8), provided no iterate coincides with one of the w_k's. We modify Weiszfeld's algorithm to find the smoothed geometric median by considering

$$z^{(t+1)} = \frac{\sum_{k=1}^m \beta_k^{(t)} w_k}{\sum_{k=1}^m \beta_k^{(t)}}, \quad \text{where,} \quad \beta_k^{(t)} = \frac{\alpha_k}{\max\left\{\nu, \|z^{(t)} - w_k\|\right\}}. \tag{14}$$

This is also stated in Algo. 7. Since each iteration of Weiszfeld's algorithm or its smoothed variant consists in taking a weighted average of the w_k's, the time complexity is $\mathcal{O}(md)$ floating point operations per iteration.

E.2 Derivation of the Smoothed Weiszfeld Algorithm

We now derive the Weiszfeld algorithm with smoothing as as an alternating minimization algorithm or as an iterative minimization of a majorizing objective.

Surrogate Definition Consider $\eta = (\eta_1, \cdots, \eta_m) \in \mathbb{R}^m$ and define $G : \mathbb{R}^d \times \mathbb{R}^m_{++} \to \mathbb{R}$ as

$$G(z, \eta) = \frac{1}{2} \sum_{k=1}^m \alpha_k \left(\frac{\|z - w_k\|^2}{\eta_k} + \eta_k\right). \tag{15}$$

Note firstly that G is jointly convex in z, η over its domain.

The first claim shows how to recover g and g_ν from G.

Claim 14. *Consider g, g_ν and G defined in Equations (8), (9) and (15), and fix $\nu > 0$. Then we have the following:*

$$g(z) = \inf_{\eta_1, \cdots, \eta_k > 0} G(z, \eta), \quad \textit{and,} \tag{16}$$

$$g_\nu(z) = \min_{\eta_1, \cdots, \eta_k \geq \nu} G(z, \eta). \tag{17}$$

Proof. Define $G_k : \mathbb{R}^d \times \mathbb{R}_{++} \to \mathbb{R}$ by

$$G_k(z, \eta_k) := \frac{1}{2}\left(\frac{\|z - w_k\|^2}{\eta_k} + \eta_k\right),$$

so that $G(z,\eta) = \sum_{k=1}^m \alpha_k G_k(z,\eta_k)$.

Since $\eta_k > 0$, the arithmetic-geometric mean inequality implies that $G_k(z,\eta_k) \geq \|z - w_k\|$ for each k. When $\|z - w_k\| > 0$, the inequality above holds with equality when $\|z - w_k\|^2/\eta_k = \eta_k$, or equivalently, $\eta_k = \|z - w_k\|$. On the other hand, when $\|z - w_k\| = 0$, let $\eta_k \to 0$ to conclude that

$$\inf_{\eta_k > 0} G_k(z,\eta_k) = \|z - w_k\| .$$

For the second part, we note that if $\|z - w_k\| \geq \nu$, then $\eta_k = \|z - w_k\| \geq \nu$ minimizes $G_k(z,\eta_k)$, so that $\min_{\eta_k \geq \nu} G_k(z,\eta_k) = \|z - w_k\|$. On the other hand, when $\|z - w_k\| < \nu$, we note that $G_k(z,\cdot)$ is minimized over $[\nu,\infty)$ at $\eta_k = \nu$, in which case we get $G_k(z,\eta) = \|z - w_k\|^2/(2\nu) + \nu/2$. From (10), we conclude that

$$\min_{\eta_k \geq \nu} G_k(z,\eta_k) = \|z - w_k\|_{(\nu)} .$$

The proof is complete since $G(z,\eta) = \sum_{k=1}^m \alpha_k G_k(z,\eta_k)$. □

Claim 14 now allows us to consider the following problem in lieu of minimizing g_ν from (9).

$$\min_{\substack{z \in \mathbb{R}^d, \\ \eta_1, \cdots, \eta_m \geq \nu}} \quad G(z,\eta) . \tag{18}$$

Alternating Minimization Next, we consider an alternating minimization algorithm to minimize G in z, η. The classical technique of alternating minimization method, known also as the block-coordinate method [see, e.g., 6], minimizes a function $f : X \times Y \to \mathbb{R}$ using the updates

$$x^{(t+1)} = \arg\min_{x \in X} f(x, y^{(t)}) \quad \text{and, } y^{(t+1)} = \arg\min_{y \in Y} f(x^{(t+1)}, y) .$$

Application of this method to Problem (18) yields the updates

$$\begin{aligned} \eta^{(t)} &= \arg\min_{\eta_1,\cdots,\eta_m \geq \nu} G(z^{(t)},\eta) = \left(\arg\min_{\eta_k \geq \nu} \left\{ \frac{\|z^{(t)} - w_k\|^2}{\eta_k} + \eta_k \right\} \right)_{k=1}^m , \\ z^{(t+1)} &= \arg\min_{z \in \mathbb{R}^d} G(z,\eta^{(t)}) = \arg\min_{z \in \mathbb{R}^d} \sum_{k=1}^m \frac{\alpha_k}{\eta_k^{(t)}} \|z - w_k\|^2 . \end{aligned} \tag{19}$$

These updates can be written in closed form as

$$\begin{aligned} \eta_k^{(t)} &= \max\{\nu, \|z^{(t)} - w_k\|\} , \\ z^{(t+1)} &= \left(\sum_{k=1}^m \frac{\alpha_k}{\eta_k^{(t)}} w_k \right) \Big/ \left(\sum_{k=1}^m \frac{\alpha_k}{\eta_k^{(t)}} \right) . \end{aligned} \tag{20}$$

This gives the smoothed Weiszfeld algorithm, as pointed out by the following claim.

Claim 15. *For any fixed $\nu > 0$ and starting point $z^{(0)} \in \mathbb{R}^d$, the sequences $\left(z^{(t)}\right)$ produced by (14) and (20), and hence, (19) are identical.*

Proof. Follows from plugging in the expression from $\eta_k^{(t)}$ in the update for $z^{(t+1)}$ in (20). □

Majorization-Minimization We now instantiate the smoothed Weiszfeld algorithm as a majorization-minimization scheme. In particular, it is the iterative minimization of a first-order surrogate in the sense of Mairal [41, 42].

Define $g_\nu^{(t)} : \mathbb{R}^d \to \mathbb{R}$ as

$$g_\nu^{(t)}(z) := G(z, \eta^{(t)}) \,, \tag{21}$$

where $\eta^{(t)}$ is as defined in (19). The z-step of (19) simply sets $z^{(t+1)}$ to be the minimizer of $g_\nu^{(t)}$.

We note the following properties of $g_\nu^{(t)}$.

Claim 16. *For $g_\nu^{(t)}$ defined in (21), the following properties hold:*

$$g_\nu^{(t)}(z) \geq g_\nu(z) \,, \quad \textit{for all} z \in \mathbb{R}^d \,, \tag{22}$$

$$g_\nu^{(t)}(z^{(t)}) = g_\nu(z^{(t)}) \,, \quad \textit{and,} \tag{23}$$

$$\nabla g_\nu^{(t)}(z^{(t)}) = \nabla g_\nu(z^{(t)}) \,. \tag{24}$$

Moreover $g^{(t)}$ can also be written as

$$g_\nu^{(t)}(z) = g_\nu(z^{(t)}) + \nabla g_\nu(z^{(t)})^\top \left(z - z^{(t)}\right) + \frac{L^{(t)}}{2} \|z - z^{(t)}\|^2 \,, \tag{25}$$

where

$$L^{(t)} := \sum_{k=1}^m \frac{\alpha_k}{\eta_k^{(t)}} \,. \tag{26}$$

Proof. The first part follows because

$$g_\nu(z) = \min_{\eta_1, \cdots, \eta_m} G(z, \eta) \leq G(z, \eta^{(t)}) = g_\nu^{(t)}(z) \,.$$

For Eq. (23), note that the inequality above is an equality at $z^{(t)}$ by the definition of $\eta^{(t)}$ from (19). To see (24), note that

$$\nabla g_\nu(z) = \sum_{k=1}^m \alpha_m \frac{z - w_k}{\max\{\nu, \|z - w_k\|\}} \,.$$

Then, by the definition of $\eta^{(t)}$ from (20), we get that

$$\nabla g_\nu(z^{(t)}) = \sum_{k=1}^m \frac{\alpha_m}{\eta_k^{(t)}} (z^{(t)} - w_k) = \nabla g_\nu^{(t)}(z^{(t)}) \,.$$

The obtain the expansion (25), we write out the Taylor expansion of the quadratic $g^{(t)}(z)$ around $z^{(t)}$ to get

$$g_\nu^{(t)}(z) = g_\nu^{(t)}(z^{(t)}) + \nabla g_\nu^{(t)}(z^{(t)})^\top \left(z - w^{(t)}\right) + \frac{L^{(t)}}{2} \|z - z^{(t)}\|^2 \,,$$

and complete the proof by plugging in (23) and (24). □

Gradient Descent The next claim rewrites the smoothed Weiszfeld algorithm as gradient descent on g_ν.

Claim 17. *Equation (14) can also be written as*

$$z^{(t+1)} = z^{(t)} - \frac{1}{L^{(t)}} \nabla g_\nu(z^{(t)}) \,, \tag{27}$$

where $L^{(t)}$ is as defined in (26).

Proof. Use $z^{(t+1)} = \arg\min_{z \in \mathbb{R}^d} g_\nu^{(t)}(z)$, where $g_\nu^{(t)}$ is written using (25). □

E.3 Properties of the Smoothed Weiszfeld Algorithm

The first claim reasons about the iterates $z^{(t)}, \eta^{(t)}$.

Claim 18. *Starting from any $z^{(0)} \in \operatorname{conv}\{w_1, \cdots, w_m\}$, the sequences $(\eta^{(t)})$ and $(z^{(t)})$ produced by Algorithm 7 satisfy*

- *$z^{(t)} \in \operatorname{conv}\{w_1, \cdots, w_m\}$ for all $t \geq 0$, and,*
- *$\nu \leq \eta_k^{(t)} \leq R$ for all $k = 1, \cdots, m$, and $t \geq 1$,*

where $R = \operatorname{diam}(\operatorname{conv}\{w_1, \cdots, w_m\})$. Furthermore, $L^{(t)}$ defined in (26) satisfies $1/R \leq L^{(t)} \leq 1/\nu$ for all $t \geq 0$.

Proof. The first part follows for $t \geq 1$ from the update (14), where Claim 15 shows the equivalence of (14) and (19). Then case of $t = 0$ is assumed. The second part follows from (20) and the first part. The bound on $L^{(t)}$ follows from the second part since $\sum_{k=1}^m \alpha_k = 1$. □

The next result shows that it is a descent algorithm. Note that the non-increasing nature of the sequence $\big(g_\nu(z^{(t)})\big)$ also follows from the majorization-minimization viewpoint [42]. Here, we show that this sequence is strictly decreasing. Recall that $z_\nu^\star$ is the unique minimizer of g_ν.

Lemma 19. *The sequence $(z^{(t)})$ produced by the smoothed Weiszfeld algorithm satisfies $g_\nu(z^{(t+1)}) < g_\nu(z^{(t)})$ unless $z^{(t)} = z_\nu^\star$.*

Proof. Let $S = \{\eta \in \mathbb{R}^m \,:\, \eta_k \geq \nu \text{ for } k = 1, \cdots, m\}$. Starting with (17), we successively deduce,

$$\begin{aligned} g_\nu(z^{(t+1)}) &= \min_{\eta_1, \cdots, \eta_m \geq \nu} G(z^{(t+1)}, \nu) \\ &\leq G(z^{(t+1)}, \eta^{(t)}) \\ &= \min_{z \in \mathbb{R}^d} G(z, \eta^{(t)}) \\ &\leq G(z^{(t)}, \eta^{(t)}) \\ &= \min_{\eta_1, \cdots, \eta_m \geq \nu} G(z^{(t+1)}, \eta) \\ &= g_\nu(z^{(t)})\,. \end{aligned}$$

Here, we used the fact that $z^{(t+1)}$ minimizes $G(\cdot, \eta^{(t)})$ over $\mathbb{R}^d$ and that $\eta^{(t)}$ minimizes $G(z^{(t)}, \cdot)$ over S.

Suppose now that $g_\nu(z^{(t+1)}) = g_\nu(z^{(t)})$. In this case, both the inequalities above hold with equality. Since $G(\cdot, \eta^{(t)})$ is $L^{(t)}$-strongly convex where $L^{(t)} \geq 1/R$ (cf. Claim 18), this implies that $z^{(t)} = \arg\min_{z \in \mathbb{R}^d} G(z, \eta^{(t)})$. By definition then, $\eta^{(t+1)} = \eta^{(t)}$ is the unique minimizer of $G(z^{(t)}, \cdot)$ over S, since $G(z^{(t)}, \cdot)$ is strictly convex. The associated first-order optimality conditions are the following:

$$\nabla_z G(z^{(t)}, \eta^{(t)}) = 0\,, \quad \text{and,} \quad \nabla_\eta G(z^{(t)}, \eta^{(t)})^\top (\eta - \eta^{(t)}) \geq 0 \quad \forall \eta \in S\,.$$

Putting these together, we find that the pair $(z^{(t)}, \eta^{(t)})$ satisfies the first-order optimality conditions for G over the domain $\mathbb{R}^d \times S$. Hence, $z^{(t)} = z_\nu^\star$. □

The next lemma shows that $\|z^{(t)} - z^\star\|$ is non-increasing. This property was shown by Beck and Sabach [4, Corollary 5.1] for the case of Weiszfeld algorithm without smoothing.

Lemma 20. *The sequence $(z^{(t)})$ produced by Algorithm 7 satisfies for all $t \geq 0$,*

$$\|z^{(t+1)} - z_\nu^\star\| \leq \|z^{(t)} - z_\nu^\star\| .$$

Furthermore, if $g_\nu(z^{(t+1)}) \geq g_\nu(z^\star)$, then it holds that

$$\|z^{(t+1)} - z^\star\| \leq \|z^{(t)} - z^\star\| .$$

Proof. We adapt the proof of Beck and Sabach [4]. First note from Claim 17 that

$$\nabla g_\nu(z^{(t)}) = L^{(t)}(z^{(t)} - z^{(t+1)}) , \tag{28}$$

where $L^{(t)}$ is defined in (26). Starting from the results of Claim 16, we observe for any z that,

$$\begin{aligned}
g_\nu(z^{(t+1)}) &\overset{(22)}{\leq} g_\nu^{(t)}(z^{(t+1)}) \\
&\overset{(25)}{=} g_\nu(z^{(t)}) + \nabla g_\nu(z^{(t)})^\top \left(z^{(t+1)} - z^{(t)}\right) + \frac{L^{(t)}}{2}\|z^{(t+1)} - z^{(t)}\|^2 \\
&\overset{(*)}{\leq} g_\nu(z) + \nabla g_\nu(z^{(t)})^\top \left(z^{(t+1)} - z\right) + \frac{L^{(t)}}{2}\|z^{(t+1)} - z^{(t)}\|^2 \\
&\overset{(28)}{\leq} g_\nu(z) + L^{(t)} \left(z^{(t)} - z^{(t+1)}\right)^\top \left(z^{(t+1)} - z\right) + \frac{L^{(t)}}{2}\|z^{(t+1)} - z^{(t)}\|^2 ,
\end{aligned}$$

where $(*)$ following from the convexity of g_ν as $g_\nu(z) \geq g_\nu(z^{(t)}) + \nabla g_\nu(z^{(t)})^\top (z - z^{(t)})$. Next, we use the Pythagorean identity: for any $a, b, c \in \mathbb{R}^d$, it holds that

$$\|b - a\|^2 + 2(b - a)^\top (a - c) = \|b - c\|^2 - \|a - c\|^2 .$$

With this, we get,

$$g_\nu(z^{(t+1)}) \leq g_\nu(z) + \frac{L^{(t)}}{2}\left(\|z^{(t)} - z\|^2 - \|z^{(t+1)} - z\|^2\right) .$$

Plugging in $z = z_\nu^\star$, the fact that $g_\nu(z^{(t+1)}) \geq g_\nu(z^\star)$ implies that $\|z^{(t+1)} - z_\nu^\star\|^2 \leq \|z^{(t)} - z_\nu^\star\|^2$, since $L^{(t)} \geq 1/R$ is strictly positive. Likewise, for $z = z^\star$, the claim holds under the condition that $g_\nu(z^{(t+1)}) \geq g_\nu(z^\star)$. □

E.4 Rate of Convergence

We are now ready to prove the global sublinear rate of convergence of Algorithm 7.

Theorem 21. *Consider the setting of Appendix D. Then, the iterate $z^{(t)}$ produced by Algorithm 7 with input $z^{(0)} \in \operatorname{conv}\{w_1, \cdots, w_k\}$ and $\nu > 0$ satisfies*

$$g_\nu(z^{(t)}) - g_\nu(z_\nu^\star) \leq \frac{2\|z^{(0)} - z_\nu^\star\|^2}{\sum_{\tau=0}^{t-1} 1/L^{(\tau)}} \leq \frac{2\|z^{(0)} - z_\nu^\star\|^2}{\widehat{\nu} t} ,$$

where $L^{(\tau)} = \sum_{k=1}^m \alpha_k/\eta_k^{(\tau)}$ is defined in (26), and

$$\widehat{\nu} = \min_{\tau=0,\cdots,t-1} \min_{k\in[m]} \max\{\nu, \|z^{(\tau)} - w_k\|\} \geq \nu . \tag{29}$$

Furthermore, it holds that

$$g(z^{(t)}) - g(z^\star) \leq \frac{2\|z^{(0)} - z^\star\|^2}{\sum_{\tau=0}^{t-1} 1/L^{(\tau)}} + \frac{\nu}{2} \leq \frac{2\|z^{(0)} - z^\star\|^2}{\widehat{\nu} t} + \frac{\nu}{2} .$$

Proof. With the descent and contraction properties of Lemmas 19 and 20 respectively, the proof now follows the classical proof technique of gradient descent [e.g., 49, Theorem 2.1.13]. Starting from the results of Claim 16, we observe for any z that,

$$\begin{aligned} g_\nu(z^{(t+1)}) &\overset{(22)}{\leq} g_\nu^{(t)}(z^{(t+1)}) \\ &\overset{(25)}{=} g_\nu(z^{(t)}) + \nabla g_\nu(z^{(t)})^\top \left(z^{(t+1)} - z^{(t)}\right) + \frac{L^{(t)}}{2}\|z^{(t+1)} - z^{(t)}\|^2 \\ &\overset{(27)}{=} g_\nu(z^{(t)}) - \frac{1}{2L^{(t)}}\|\nabla g_\nu(z^{(t)})\|^2 \,. \end{aligned} \tag{30}$$

Convergence on g_ν For ease of notation, we let $\widetilde{\Delta}_t := g_\nu(z^{(t)}) - g_\nu(z_\nu^\star)$. We assume now that $\widetilde{\Delta}_{t+1}$ is nonzero, and hence, so is $\widetilde{\Delta}_t$ (Lemma 19). If $\widetilde{\Delta}_{t+1}$ were zero, then the theorem would hold trivially at $t+1$.

Now, from convexity of g_ν and the Cauchy-Schwartz inequality, we get that

$$\widetilde{\Delta}_t \leq \nabla g_\nu(z^{(t)})^\top \left(z^{(t)} - z_\nu^\star\right) \leq \|\nabla g_\nu(z^{(t)})\|\|z^{(t)} - z_\nu^\star\| \,.$$

Plugging this in, we get,

$$\begin{aligned} \widetilde{\Delta}_{t+1} - \widetilde{\Delta}_t &\leq -\frac{1}{2L^{(t)}} \frac{\widetilde{\Delta}_t^2}{\|z^{(t)} - z_\nu^\star\|^2} \\ &\leq -\frac{1}{2L^{(t)}} \frac{\widetilde{\Delta}_t^2}{\|z^{(0)} - z_\nu^\star\|^2} \,, \end{aligned}$$

where we invoked Lemma 20.

Now, we divide by $\widetilde{\Delta}_t\widetilde{\Delta}_{t+1}$, which is nonzero by assumption, and use $\widetilde{\Delta}_t/\widetilde{\Delta}_{t+1} \geq 1$ (Lemma 19) to get

$$\begin{aligned} \frac{1}{\widetilde{\Delta}_t} - \frac{1}{\widetilde{\Delta}_{t+1}} &\leq -\frac{1}{2L^{(t)}} \left(\frac{\widetilde{\Delta}_t}{\widetilde{\Delta}_{t+1}}\right) \frac{1}{\|z^{(0)} - z_\nu^\star\|^2} \\ &\leq -\frac{1}{2L^{(t)}\|z^{(0)} - z_\nu^\star\|^2} \,. \end{aligned}$$

Telescoping, we get,

$$\frac{1}{\widetilde{\Delta}_t} \geq \frac{1}{\widetilde{\Delta}_t} - \frac{1}{\widetilde{\Delta}_0} \geq \left(\sum_{\tau=0}^{t-1} \frac{1}{L^{(\tau)}}\right) \frac{1}{2\|z^{(0)} - z_\nu^\star\|^2} \,.$$

This proves the first inequality to be proved. The second inequality follows from the definition in Eq. (26) since $\sum_{k=1}^m \alpha_k = 1$.

Convergence on g The proof follows along the same ideas as the previous proof. Define $\Delta_t := g_\nu(z^{(t)}) - g_\nu(z^\star)$. Suppose $\Delta_t > 0$. Then, we proceed as previously for any $\tau < t$ to note by convexity and Cauchy-Schwartz inequality that

$$\Delta_\tau \leq \|\nabla g_\nu(z^{(\tau)})\|\|z^{(\tau)} - z^\star\| \,.$$

Again, plugging this into (30), using that $\Delta_\tau/\Delta_{\tau+1} \geq 1$ and invoking Lemma 20 gives (since $\Delta_\tau > 0$)

$$\frac{1}{\Delta_\tau} - \frac{1}{\Delta_{\tau+1}} \leq -\frac{1}{2L^{(\tau)}\|z^{(0)} - z^\star\|^2} \,.$$

Telescoping and taking the reciprocal gives

$$g_\nu(z^{(t)}) - g_\nu(z^\star) = \Delta_t \le \frac{2\|z^{(0)} - z^\star\|^2}{\sum_{\tau=0}^{t-1} 1/L^{(\tau)}} \,.$$

Using (11) completes the proof for the case that $\Delta_t > 0$. Note that if $\Delta_t \le 0$, it holds that $\Delta_{t'} \le 0$ for all $t' > t$. In this case, $g_\nu(z^{(t)}) - g_\nu(z^\star) \le 0$. Again, (11) implies that $g(z^{(t)}) - g(z^\star) \le \nu/2$, which is trivially upper bounded by the quantity stated in the theorem statement. This completes the proof. □

E.5 Faster Rate of Convergence

We now make an additional assumption:

Assumption 22. The geometric median $z^\star$ does not coincide with any of $w_1, \cdots, w_m$. In other words,

$$\widetilde{\nu} := \min_{k=1,\cdots,m} \|z^\star - w_k\| > 0 \,. \tag{31}$$

Remark 23. *Beck and Sabach [4, Lemma 8.1] show a lower bound on $\widetilde{\nu}$ in terms of $\alpha_1, \cdots, \alpha_m$ and $w_1, \cdots, w_m$.*

Now, we analyze the condition under which the $z^\star = z_\nu^\star$.

Lemma 24. *Under Assumption 22, we have that $z^\star = z_\nu^\star$ for all $\nu \le \widetilde{\nu}$, where $\widetilde{\nu}$ is defined in (31).*

Proof. By the definition of the smooth norm in (10), we observe that $\|z^\star - w_k\|_{(\nu)} = \|z^\star - w_k\|$ for all $\nu \le \widetilde{\nu}$, and hence, $g_\nu(z^\star) = g(z^\star)$. For any $z \in \mathbb{R}^d$, we have,

$$g_\nu(z) \overset{(11)}{\ge} g(z) \ge g(z^\star) = g_\nu(z^\star) \,,$$

or that $z^\star = z_\nu^\star$. □

In this case, we get a better rate on the non-smooth objective g.

Corollary 25. *Consider the setting of Theorem 21 where Assumption 22 holds and $\nu \le \widetilde{\nu}$. Then, the iterate $z^{(t)}$ produced by Algorithm 7 satisfies,*

$$g(z^{(t)}) - g(z^\star) \le \frac{2\|z^{(0)} - z^\star\|^2}{\widehat{\nu} t} \,,$$

where $\widehat{\nu}$ is defined in Eq. (29).

Proof. This follows from Theorem 21's bound on $g_\nu(z^{(t)}) - g_\nu(z_\nu^\star)$ with the observations that $g(z^{(t)}) \overset{(11)}{\le} g_\nu(z^{(t)})$ and $g(z^\star) = g_\nu(z^\star)$ (see the proof of Lemma 24). □

The previous corollary obtains the same rate as Beck and Sabach [4, Theorem 8.2], up to constants upon using the bound on $\widetilde{\nu}$ given by Beck and Sabach [4, Lemma 8.1].

We also get as a corollary a bound on the performance of Weiszfeld's original algorithm without smoothing, although it could be numerically unstable in practice. This bound depends on the actual iterates, so it is not informative about the performance of the algorithm a priori.

Corollary 26. *Consider the setting of Theorem 21. Under Assumption 22, suppose the sequence* $(z^{(t)})$ *produced by Weiszfeld's algorithm in Eq. (13) satisfies* $\|z^{(t)} - w_k\| > 0$ *for all* t *and* k*, then it also satisfies*

$$g(z^{(t)}) - g(z^\star) \leq \frac{2\|z^{(0)} - z^\star\|^2}{\nu^{(t)} t} .$$

where $\nu^{(t)}$ *is given by*

$$\nu^{(t)} = \min\left\{\widetilde{\nu}, \min_{\tau=0,\cdots,t} \min_{k\in[m]} \|z^{(\tau)} - w_k\|\right\} .$$

Proof. Under these conditions, note that the sequence $(z^{(\tau)})_{\tau=0}^t$ produced by the Weiszfeld algorithm without smoothing coincides with the sequence $(z^{(\tau)}_{\nu^{(t)}})_{\tau=0}^t$ produced by the smoothed Weiszfeld algorithm at level $\nu = \nu^{(t)}$. Now apply Cor. 25. □

E.6 Comparison to Previous Work

We compare the results proved in the preceding section to prior work on the subject.

Comparison to Beck and Sabach [4] They present multiple different variants of the Weiszfeld algorithm. For a particular choice of initialization, they can guarantee that a rate of the order of $1/\widetilde{\nu}t$. This choice of initialization cannot be implemented using a secure aggregation oracle since it requires the computation of all pairwise distances $\|w_k - w_{k'}\|$. Moreover, a naive implementation of their algorithm could be numerically unstable since it would involve division by small numbers. Guarding against division by small numbers would lead to the smoothed variant considered here. Note that our algorithmic design choices are driven by the federated learning setting.

Comparison to Beck [3] He studies general alternating minimization algorithms, including the Weiszfeld algorithm as a special case, with a different smoothing than the one considered here. While his algorithm does not suffer from numerical issues arising from division by small numbers, it always suffers a bias from smoothing. On the other hand, the smoothing considered here is more natural in that it reduces to Weiszfeld's original algorithm when $\|z^{(t)} - w_k\| > \nu$, i.e., when we are not at a risk of dividing by small numbers. Furthermore, the bound in Thm. 21 exhibits a better dependence on the initialization $z^{(0)}$.

F Computing the Geometric Median with Mirror-Prox

Here, we review the mirror-prox algorithm, as presented by Juditsky and Nemirovski [28], and then instantiate it for our setting.

F.1 Additional Notation

We first describe the some notation in addition to what has been presented in Appendix D.

In this section $\|\cdot\|$ refers to a general norm, while the Euclidean norm is denoted as $\|\cdot\|_2$. We denote by $\|\cdot\|_*$ the dual norm to $\|\cdot\|$. Next, for $z \in \mathbb{R}^d$ and $u \in \mathbb{R}^{d'}$, we use shorthand $(z, u) \in \mathbb{R}^{d+d'}$ to denote the concatenation of z and u. On the other hand, for $U \in \mathbb{R}^{d\times m}$, $(z, U) \in \mathbb{R}^{d(m+1)}$ denotes $(z, u_1, \cdots, u_m)$, where u_k is the k^{th} column of U.

F.2 The Mirror-Prox Algorithm: Review

For convex, compact sets $\mathcal{Z} \subset \mathbb{R}^d$ and $\mathcal{U} \subset \mathbb{R}^{d'}$, fix a norm $\|\cdot\|$ on $\mathcal{Z} \times \mathcal{U}$. Consider the function $\phi : \mathcal{Z} \times \mathcal{U} \to \mathbb{R}$ satisfying the following assumption.

Assumption 27. The function $\phi(z, u)$ is convex-concave, i.e., $\phi(\cdot, u)$ is convex for all $u \in \mathcal{U}$ and $\phi(z, \cdot)$ is concave for all $z \in \mathcal{Z}$. Furthermore, its gradient $\nabla\phi(z, u) = (\nabla_z\phi(z, u), \nabla_u\phi(z, u))$ is L-Lipschitz with respect to $\|\cdot\|$.

With some abuse of notation, we will interchangeably use $\phi(z, u)$ and $\phi((z, u))$.
We are interested in the saddle point problem

$$\min_{z\in\mathcal{Z}} \max_{u\in\mathcal{U}} \phi(z, u)\,. \tag{32}$$

The algorithm is stated in terms of the monotone operator $\Phi : \mathcal{Z} \times \mathcal{U} \to \mathbb{R}^d \times \mathbb{R}^{d'}$ associated with (32) defined as

$$\Phi(z, u) = (\nabla_z\phi(z, u), -\nabla_u\phi(z, u))\ . \tag{33}$$

Since we assume that $\nabla\phi(z, u)$ is L-Lipschitz w.r.t. $\|\cdot\|$, it follows that $\Phi(z, u)$ is also L-Lipschitz w.r.t. $\|\cdot\|$.

Setup Consider a distance generating function $\omega : \mathcal{Z} \times \mathcal{U} \to \mathbb{R}$ which is (for simplicity) continuously differentiable and 1-strongly convex w.r.t. $\|\cdot\|$. Further, for $v, v' \in \mathcal{Z} \times \mathcal{U}$, define the Bregman divergence

$$V_v(v') = \omega(v') - \omega(v) - \nabla\omega(v)^\top(v' - v)\,.$$

For any $v \in \mathcal{Z} \times \mathcal{U}$, define the prox mapping $\mathrm{prox}_v : \mathbb{R}^{d+d'} \to \mathcal{Z} \times \mathcal{U}$ as

$$\mathrm{prox}_v(\xi) = \operatorname*{arg\,min}_{v'\in\mathcal{Z}\times\mathcal{U}} \left\{\langle\xi, v'\rangle + V_v(v')\right\}\ .$$

The quality of a putative solution $v = (z, u)$ is measured by the gap function

$$\epsilon_{\text{saddle}}((z, u)) = \max_{u'\in\mathcal{U}} \phi(z, u') - \min_{z'\in\mathcal{Z}} \phi(z', u)\,.$$

Lastly, we define Ω, the ω-diameter of $\mathcal{Z} \times \mathcal{U}$ as

$$\Omega = \max_{v\in\mathcal{Z}\times\mathcal{U}} \omega(v) - \min_{v\in\mathcal{Z}\times\mathcal{U}} \omega(v)\,.$$

Algorithm The mirror-prox algorithm is given in Algo. 8. Each iteration has two prox steps - a prediction and a correction step. We simply use a constant learning rate γ here. An averaged iterate $\overline{v}_t$ is used, as highlighted by the following theorem. See Juditsky and Nemirovski [28, Prop. 6.1] for a proof.

Theorem 28 (Juditsky and Nemirovski [28])**.** *Consider the setting of this section, where ϕ satisfies Assumption 27. Then, Algo. 8 with learning rate $\gamma = 1/L$ produces a sequence $(\overline{v}_t)_{t=1}^{\infty}$, which satisfies*

$$\epsilon_{\text{saddle}}(\overline{v}_t) \le \frac{\Omega L}{t}\,.$$

Algorithm 8 The mirror-prox algorithm

Input: ϕ as in (32) and its associated monotone operator Φ, a norm $\|\cdot\|$, a distance generating function ω, learning rate γ.

Initialize $v_1 = \arg\min_{v \in \mathcal{Z} \times \mathcal{U}} \omega(v)$ and set $\overline{v}_1 = v_1$.
for $t = 1, 2, \cdots$ **do**
  $\widehat{v}_t = \mathrm{prox}_{v_t}(\gamma \Phi(v_t))$ ▷ Prediction
  $v_{t+1} = \mathrm{prox}_{v_t}(\gamma \Phi(\widehat{v}_t))$ ▷ Correction
  $\overline{v}_t = \frac{1}{t} \sum_{\tau=1}^{t} \widehat{v}_\tau$ ▷ Averaging

F.3 The Mirror-Prox Algorithm for Robust Aggregation

We come back now to applying the mirror-prox algorithm to find the geometric median. We consider the more general problem of minimizing g_ν for some $\nu \geq 0$, where $g_0 \equiv g$. Note that minimizing a smooth function, such as the smooth function g_ν, by finding the saddle point of an appropriate surrogate using mirror-prox is suboptimal. Indeed, Nesterov's acclerated gradient method acheives a faster rate. In our experiments, we actually apply mirror-prox to the non-smooth case $\nu = 0$, in which case it is optimal.

Surrogate Recall the definition of constraint sets

$$\mathcal{U} = \left\{ U \in \mathbb{R}^{d \times m} \,:\, u_k^\top u_k \leq 1 \text{ for } k = 1, \cdots, m \right\}, \text{ and}$$
$$\mathcal{Z} = \left\{ z \in \mathbb{R}^d \,:\, \|z\|_2 \leq R \right\}.$$

We are interested in the saddle point problem

$$\min_{z \in \mathcal{Z}} \max_{U \in \mathcal{U}} \left[\phi_\nu(z, U) := \sum_{k=1}^{m} \alpha_k u_k^\top (z - w_k) - \frac{\nu}{2} \sum_{k=1}^{m} \|u\|_2^2 \right]. \tag{34}$$

The monotone operator associated with (34) is

$$\Phi_\nu(z, U) = \begin{pmatrix} \sum_{k=1}^{m} \alpha_k u_k \\ -\alpha_1 (z - w_1) + \nu u_1 \\ \vdots \\ -\alpha_m (z - w_m) + \nu u_m \end{pmatrix}. \tag{35}$$

We can easily verify that ϕ_ν is convex-concave, or equivalently that Φ_ν is monotone, for any $\nu \geq 0$.

We now show that this saddle point problem is equivalent to the original minimization problem.

Claim 29. *For any $w_1, \cdots, w_m \in \mathbb{R}^d$, $\alpha_1, \cdots, \alpha_m > 0$, and $\nu \geq 0$, it holds that,*

$$\min_{z \in \mathcal{Z}} \max_{U \in \mathcal{U}} \phi_\nu(z, U) = \min_{z \in \mathbb{R}^d} g_\nu(z).$$

Proof. Since $\mathcal{U}$ is the Cartesian product of $\{u_k \in \mathbb{R}^d \,:\, u_k^\top u_k \leq 1\}$, and the objective is separable, it follows by definition of $\|\cdot\|_{(\nu)}$ that $\max_{U \in \mathcal{U}} \phi_\nu(z, U) = g_\nu(z)$.

To complete the proof, we now show that any minimizer $z^\star$ of g_ν must occur in $\mathrm{conv}\{w_1, \cdots w_m\}$, which is contained in $\mathcal{Z}$. The case $\nu = 0$ is shown by Kuhn [32, Thm. 2.2]. Suppose $\nu > 0$. From first-order optimality conditions of $z^\star$, we deduce that

$$\sum_{k=1}^{m} \beta_k (z^\star - w_k) = 0 \iff z^\star = \frac{\sum_{k=1}^{m} \beta_k w_k}{\sum_{k=1}^{m} \beta_k},$$

where $\beta_k = \alpha_k / \max\{\nu, \|z^\star - w_k\|\} > 0$. □

For the algorithm to be fully specified, we must specify a norm $\|\cdot\|$ as well as a distance generating function ω. To this end, we define a constant

$$\kappa := \frac{m}{R^2}\,. \tag{36}$$

Norm Define the norm $\|\cdot\|$ on $\mathcal{Z} \times \mathcal{U}$ as

$$\|(z, U)\|^2 := \kappa\|z\|_2^2 + \sum_{k=1}^m \|u_k\|_2^2\,. \tag{37}$$

Its dual norm is given by

$$\|(z, U)\|_*^2 := \frac{1}{\kappa}\|z\|_2^2 + \sum_{k=1}^m \|u_k\|_2^2\,.$$

Lipschitz constant We now find the Lipschitz constant L of Φ_ν.

Claim 30. *Fix any $\nu \geq 0$, the operator Φ_ν from (35) is L-Lipschitz with respect to $\|\cdot\|$ from (37) where*

$$L \leq 2\max\{\nu, 1/\sqrt{\kappa}\}\,. \tag{38}$$

Proof. For any $(z, U), (z', U') \in \mathcal{Z} \times \mathcal{U}$, we successively deduce,

$$\begin{aligned}
\|\Phi_\nu(z', U') - \Phi_\nu(z, U)\|_*^2 &= \frac{1}{\kappa}\left\|\sum_{k=1}^m \alpha_k(u_k' - u_k)\right\|_2^2 + \sum_{k=1}^m \|\alpha_k(z_k - z_k') + \nu(u_k' - u_k)\|_2^2 \\
&\overset{(*)}{\leq} \sum_{k=1}^m \frac{\alpha_k}{\kappa}\|u_k' - u_k\|_2^2 + \sum_{k=1}^m 2\alpha_k^2\|z' - z\|_2^2 + \sum_{k=1}^m 2\nu^2\|u_k' - u_k\|_2^2 \\
&= \kappa\|z' - z\|_2^2\left(\frac{2}{\kappa}\sum_{k=1}^m \alpha_k^2\right) + \sum_{k=1}^m \|u_k' - u_k\|_2^2\left(2\nu^2 + \frac{\alpha_k}{\kappa}\right) \\
&\overset{(\#)}{\leq} \max\left\{2\nu^2 + \frac{1}{\kappa}, \frac{2}{\kappa}\right\}\left(\kappa\|z' - z\|_2^2 + + \sum_{k=1}^m \|u_k' - u_k\|_2^2\right) \\
&= \max\left\{2\nu^2 + \frac{1}{\kappa}, \frac{2}{\kappa}\right\}\|(z', U') - (z, U)\|^2\,.
\end{aligned}$$

Here, $(*)$ followed from a combination of the Jensen's inequality to $\|\cdot\|_2^2$ and $\|a + b\|_2^2 \leq 2\|a\|_2^2 + 2\|b\|_2^2$, while $(\#)$ followed from the fact that $0 < \alpha_k < 1$ and

$$\sum_{k=1}^m \alpha_k^2 \leq \left(\sum_{k=1}^m \alpha_k\right)^2 = 1\,.$$

We thus conclude that $L^2 \leq \max\left\{2\nu^2 + 1/\kappa, 2/\kappa\right\}$. When $2\nu^2\kappa < 1$, we get that $L^2 < 2/\kappa < 4/\kappa$. On the other hand, when $2\nu^2\kappa \geq 1$, we get that $1/\kappa \leq 2\nu^2$ and $L^2 \leq 4\nu^2$. Combining the two, we get that $L^2 \leq 4\max\{\nu^2, 1/\kappa\}$. □

Algorithm 9 The Mirror-Prox Algorithm to find the Smoothed Geometric Median

Input: $w_1, \cdots, w_m \in \mathbb{R}^d$, $\alpha_1, \cdots, \alpha_m > 0$ with $\sum_k \alpha_k = 1$, $\nu \geq 0$, learning rate γ, primal scaling factor $\kappa > 0$, iteration budget T

Compute $\overline{w} = \sum_{k=1}^m \alpha_k w_k$ and set $R = \max_{k=1,\cdots,m} \|\overline{w} - w_k\|$

Set $z_1 = \overline{w}$ and $u_k = 0$ for $k = 1, \cdots, m$

for $t = 1, 2, \cdots, t$ **do**

$\quad \widehat{z}^{(t)} = \mathrm{proj}_{\mathcal{Z}} \left(z^{(t)} - (\gamma/\kappa) \sum_{k=1}^m \alpha_k u_k^{(t)} \right)$

$\quad \widehat{u}_k^{(t)} = \mathrm{proj}_{\mathcal{B}} \left(u_k^{(t)} + \gamma \alpha_k (z^{(t)} - w_k) - \gamma \nu u_k^{(t)} \right)$ for each $k = 1, \cdots, m$

$\quad \overline{z}^{(t)} = \overline{z}^{(t-1)}(t-1)/t + \widehat{z}^{(t)}/t$

$\quad z^{(t+1)} = \mathrm{proj}_{\mathcal{Z}} \left(z^{(t)} - (\gamma/\kappa) \sum_{k=1}^m \alpha_k \widehat{u}_k^{(t)} \right)$

$\quad u_k^{(t+1)} = \mathrm{proj}_{\mathcal{B}} \left(u_k^{(t)} + \gamma \alpha_k (\widehat{z}^{(t)} - w_k) - \gamma \nu \widehat{u}_k^{(t)} \right)$ for each $k = 1, \cdots, m$

return $\overline{z}^{(T)}$

Distance generating function and prox mapping Next, we define the following distance generating function $\omega : \mathcal{Z} \times \mathcal{U} \to \mathbb{R}$:

$$\omega(z, U) = \frac{1}{2}\|(z, U)\|^2 = \frac{\kappa}{2}\|z\|_2^2 + \frac{1}{2}\sum_{k=1}^m \|u_k\|_2^2 \,. \tag{39}$$

It is evidently continuously differentiable and 1-strongly convex w.r.t. $\|\cdot\|$. The ω-diameter of $\mathcal{Z} \times \mathcal{U}$ is then

$$\Omega = \max_{v \in \mathcal{Z} \times \mathcal{U}} \omega(v) - \min_{v \in \mathcal{Z} \times \mathcal{U}} \omega(v) = \frac{1}{2}(\kappa R^2 + m) = m \,. \tag{40}$$

The corresponding Bregman divergence is

$$V_{(z,U)}((z', U')) = \frac{\kappa}{2}\|z' - z\|_2^2 + \frac{1}{2}\sum_{k=1}^m \|u_k' - u_k\|_2^2 \,,$$

and the associated prox-mapping is given by the following claim.

Claim 31. *Let $\mathcal{B} = \{u \in \mathbb{R}^d : u^\top u \leq 1\}$ denote the unit ball in $\mathbb{R}^d$. We have,*

$$\begin{aligned}
\mathrm{prox}_{(z,U)}\left((\xi, \zeta_1, \cdots, \zeta_m)\right) &= \begin{pmatrix} \mathrm{proj}_{\mathcal{Z}}\left(z - \kappa^{-1}\xi\right) \\ \mathrm{proj}_{\mathcal{B}}\left(u_1 - \zeta_1\right) \\ \vdots \\ \mathrm{proj}_{\mathcal{B}}\left(u_m - \zeta_m\right) \end{pmatrix} \\
&= \begin{pmatrix} \overline{w} + R(z - \overline{w} - \kappa^{-1}\xi)/\max\{R, \|z - \overline{w} - \kappa^{-1}\xi\|_2\} \\ (u_1 - \zeta_1)/\max\{1, \|u_1 - \zeta_1\|_2\} \\ \vdots \\ (u_m - \zeta_m)/\max\{1, \|u_m - \zeta_m\|_2\} \end{pmatrix} .
\end{aligned}$$

Proof. We show the derivation for z-component of the prox mapping. The derivation for each u_k is completely

analogous. We have,

$$\begin{aligned}
\mathrm{prox}_z(\xi) &= \arg\min_{z' \in \mathcal{Z}} \left\{ \xi^\top z' + \frac{\kappa}{2}\|z' - z\|_2^2 \right\} \\
&= \arg\min_{z' \in \mathcal{Z}} \left\{ \xi^\top z' + \frac{\kappa}{2}\|z'\|_2^2 - \kappa z^\top z' \right\} \\
&= \arg\min_{z' \in \mathcal{Z}} \left\{ \frac{\kappa}{2}\|z'\|_2^2 - (\kappa z - \xi)^\top z' \right\} \\
&= \arg\min_{z' \in \mathcal{Z}} \left\{ \frac{1}{2}\|z'\|_2^2 - (z - \kappa^{-1}\xi)^\top z' \right\} \\
&= \arg\min_{z' \in \mathcal{Z}} \left\| z' - (z - \kappa^{-1}\xi) \right\|_2^2 \\
&= \mathrm{proj}_{\mathcal{Z}} \left(z - \kappa^{-1}\xi \right) .
\end{aligned}$$

Since $\mathcal{Z}$ is the ball $\{z \in \mathbb{R}^d : \|z - \overline{w}\|_2 \leq R\}$, the second equality follows. □

Algorithm The overall algorithm is shown in Algo. 9.

Performance guarantee

Corollary 32. *Consider the saddle point problem defined in (34) for any $\nu \geq 0$. Then, Algo. 9 (mirror-prox) generates a sequence $((\overline{z}_t, \overline{U}_t))_{t=1}^{\infty}$ which satisfies*

$$g_\nu(\overline{z}_t) - \min_z g_\nu(z) \leq \epsilon_{\text{saddle}}(\overline{z}_t, \overline{U}_t) \leq \frac{2\max\{m\nu, \sqrt{m}R\}}{t} .$$

Proof. Algo. 9 is nothing but Algo. 8 instantiated with the norm from (37) and distance generating function from (39). The second inequality comes from plugging into Theorem 28 the expressions of Ω, L respectively from Claim 30 and Eq. (40). Towards proving the first inequality, we first note that

$$\min_{z \in \mathcal{Z}} \max_{U \in \mathcal{U}} \phi_\nu(z, U) = \max_{U \in \mathcal{U}} \min_{z \in \mathcal{Z}} \phi_\nu(z, U) =: \phi_\nu^\star .$$

This follows, for instance, from Hiriart-Urruty and Lemaréchal [21, Thm. VII.4.3.1]. Next, for any $(z, U) \in \mathcal{Z} \times \mathcal{U}$, we note that $g_\nu(z) = \max_{U' \in \mathcal{U}} \phi_\nu(z, U')$ by definition, and that,

$$\phi_\nu^* = \max_{U' \in \mathcal{U}} \min_{z' \in \mathcal{Z}} \phi_\nu(z', U') \geq \min_{z' \in \mathcal{Z}} \phi_\nu(z', U) .$$

Thus, we get the first inequality. □

F.4 The Mirror-Prox Algorithm for Robust Aggregation in Federated Learning: Details

The application of mirror-prox for robust aggregation in the federated learning setting is given in Algo. 10. Each iteration of the algorithm requires two calls the secure aggregation oracle.

Note that both $\mathcal{Z}$ and $\mathcal{B}$ are balls. Thus the projections can be computed in $\mathcal{O}(d)$ time as follows:

$$\begin{aligned}
\mathrm{proj}_{\mathcal{Z}}(z') &= \overline{w} + R(z' - \overline{w})/\max\{R, \|z' - \overline{w}\|_2\} , \\
\mathrm{proj}_{\mathcal{B}}(u') &= u/\max\{1, \|u\|_2\} .
\end{aligned}$$

Algorithm 10 The Mirror-Prox Algorithm to for Robust Aggregation in FL

Input: $w_1, \cdots, w_m \in \mathbb{R}^d$, $\alpha_1, \cdots, \alpha_m > 0$ with $\sum_k \alpha_k = 1$, iteration budget T, secure average oracle $\mathcal{A}$, learning rate γ, primal scaling factor $\kappa > 0$, iteration budget T

Compute $\overline{w} = \sum_{k=1}^m \alpha_k w_k$ using $\mathcal{A}$

Server broadcasts $\overline{w}$ to devices and computes $R = \max_{k=1,\cdots,m} \|\overline{w} - w_k\|$

Server $z_1 = \overline{w}$ and each device sets $u_k = 0$ for $k = 1, \cdots, m$

for $t = 1, 2, \cdots, t$ **do**

  Server broadcasts $z^{(t)}$ to devices and computes $\overline{u}^{(t)} = \sum_{k=1}^m \alpha_k u_k^{(t)}$ using $\mathcal{A}$

  Server sets $\widehat{z}^{(t)} = \mathrm{proj}_{\mathcal{Z}}\left(z^{(t)} - (\gamma/\kappa)\overline{u}^{(t)}\right)$

  Device k sets $\widehat{u}_k^{(t)} = \mathrm{proj}_{\mathcal{B}}\left(u_k^{(t)} + \gamma\alpha_k(z^{(t)} - w_k)\right)$ for $k = 1, \cdots, m$

  Server sets $\overline{z}^{(t)} = \overline{z}^{(t-1)}(t-1)/t + \widehat{z}^{(t)}/t$

  Server broadcasts $\widehat{z}^{(t)}$ to devices and computes $\widetilde{u}^{(t)} = \sum_{k=1}^m \alpha_k \widehat{u}_k^{(t)}$ using $\mathcal{A}$

  Server sets $z^{(t+1)} = \mathrm{proj}_{\mathcal{Z}}\left(z^{(t)} - (\gamma/\kappa)\widetilde{u}^{(t)}\right)$

  Device k sets $u_k^{(t+1)} = \mathrm{proj}_{\mathcal{B}}\left(u_k^{(t)} + \gamma\alpha_k(\widehat{z}^{(t)} - w_k)\right)$ for each $k = 1, \cdots, m$

return $\overline{z}^{(T)}$

G Experiments: Full Details

The section contains a full description of the experimental setup as well as the results.

G.1 Datasets and Task Description

We experiment with two tasks, handwritten-letter recognition and character-level language modeling.

G.1.1 Handwritten-Letter Recognition

The first dataset is the EMNIST dataset [15] for handwritten letter recognition.

Data Each datapoint x is a gray-scale image resized to 28×28. Each output y is categorical variable which takes 62 different values, one per class of letter (0-9, a-z, A-Z).

Formulation The task of handwritten letter recognition is cast as a multi-class classification problem with 62 classes.

Distribution of Data The handwritten characters in the images are annotated by the writer of the character as well. We use a non-i.i.d. split of the data grouped by a writer of a given image. We discard devices with less than 100 total input-output pairs (both train and test), leaving a total of 3461 devices. Of these, we sample 1000 devices to use for our experiments, corresponding to about 30% of the data. This selection held constant throughout our experiments. The number of training examples across these devices summarized in the following statistics: median 160, mean 202, standard deviation 77, maximum 418 and minimum 92. This preprocessing was performed using LEAF [11].

Models We use two different models, a linear model and a convolutional neural network.

- Linear Model: The linear model maintains parameters $w_1, \cdots, w_{62} \in \mathbb{R}^{28\times 28}$. For a given image x, class l is assigned score $\langle w_l, x\rangle$, which is then converted to a probability using a softmax operation as $\exp(\langle w_l, x\rangle)/\sum_{l'}\exp(\langle w_{l'}, x\rangle)$. For a new input image x, the prediction is made as $\arg\max_l \langle w_l, x\rangle$.
- Convolutional Neural Network (ConvNet): The ConvNet [36] we consider contains two convolutional layers with max-pooling, followed by a fully connected hidden layer, and another fully connected (F.C.) layer with 62 outputs. When given an input image x, the output of this network is assigned as the scores of each of the classes. Probabilities are assigned similar to the linear model with a softmax operation on the scores. The schema of network is given below:

$$\begin{array}{c}\text{Input}\\ 28\times 28\end{array} \longrightarrow \begin{array}{c}\text{Conv2D}\\ \text{filters} = 32\\ \text{kernel} = 5\times 5\end{array} \longrightarrow \text{ReLU} \longrightarrow \begin{array}{c}\text{Max Pool}\\ \text{kernel} = 2\times 2\\ \text{stride} = 2\end{array} \longrightarrow \begin{array}{c}\text{Conv2D}\\ \text{filters} = 64\\ \text{kernel} = 5\times 5\end{array} \longrightarrow$$

$$\text{ReLU} \longrightarrow \begin{array}{c}\text{Max Pool}\\ \text{kernel} = 2\times 2\\ \text{stride} = 2\end{array} \longrightarrow \begin{array}{c}\text{F.C.}\\ \text{units} = 2048\end{array} \longrightarrow \text{ReLU} \longrightarrow \begin{array}{c}\text{F.C.}\\ \text{units} = 62\end{array} \longrightarrow \text{score}$$

Loss Function We use the cross-entropy loss $\ell(y, p) = -\log p_y$, for probabilities $p = (p_1, \cdots, p_{62})$ and $y \in \{1, \cdots, 62\}$. In the linear model case, it is equivalent to the classical softmax regression.

Evaluation Metric The model is evaluated based on the classification accuracy on the test set.

G.1.2 Character-Level Language Modeling

The second task is to learn a character-level language model over the Complete Works of Shakespeare [53]. The goal is to read a few characters and predict the next character which appears.

Data The dataset consists of text from the Complete Works of William Shakespeare in a big text file.

Formulation We formulate the task as a multi-class classification problem with 53 classes (a-z, A-Z, other) as follows. At each point, we consider the previous $H = 20$ characters, and build $x \in \{0, 1\}^{H\times 53}$ as a one-hot encoding of these H characters. The goal is then try to predict the next character, which can belong to 53 classes. In this manner, a text with l total characters gives l input-output pairs.

Distribution of Data We use a non-i.i.d. split of the data. Each role in a given play (e.g., Brutus from The Tragedy of Julius Caesar) is assigned as a separate device. All devices with less than 100 total examples are discarded, leaving 628 devices. The training set is assigned a random 90% of the input-output pairs, and the other rest are held out for testing. This distribution of training examples is extremely skewed, with the following statistics: median 1170, mean 3579, standard deviation 6367, maximum 70600 and minimum 90. This preprocessing was performed using LEAF [11].

Models We use a long-short term memory model (LSTM) [22] with 128 hidden units for this purpose. This is followed by a fully connected layer with 53 outputs, the output of which is used as the score for each character. As previously, probabilities are obtained using the softmax operation.

Loss Function We use the cross-entropy loss.

Figure 7: Illustration of the data noise introduced in the Shakespeare dataset. The first line denotes the original text. The second line shows the effective x when predicting the "m" of the word "median". The second line shows the corresponding $\widetilde{x}$ after the introduction of the noise. Note that $\widetilde{x}$ is the string "edian's ro" reversed.

Evaluation Metric The model is evaluated based on the accuracy of next-character prediction on the test set.

G.2 Methods, Hyperparameters and Variants

We first describe the noise model, followed by various methods tested.

G.2.1 Noise Model

Since the goal of this work to test the robustness of federated learning models to high noise settings, we artificially inject noise into the method while controlling the level of noise. We experiment with two different noise models in the local computation Ψ defined in Appendix A: data noise or parameter noise.

Data Noise : The model training procedure is not modified, but the data fed into the model is modified. In particular, for some modification $\widetilde{\mathcal{D}}_k$ of the local dataset $\mathcal{D}$ of client k, we use

$$\Psi(k, w, H, \mathcal{D}_k, \mathcal{D}, \mathcal{S}) = \Phi(w, H, \widetilde{\mathcal{D}}_k)\,.$$

The exact nature of the modification $\widetilde{\mathcal{D}}_k$ depends on the dataset:

- EMNIST: For an image-label pair (x, y), its modification $(\widetilde{x}, \widetilde{y})$ is $\widetilde{x} = 1 - x$, the negative of the image x (assuming the pixels are normalized to lie in $[0, 1]$) and $\widetilde{y} = y$.
- Shakespeare: We reverse the original text. The procedure in which (x, y) examples are constructed from the text remains unchanged. This is illustrated in Fig. 7.

Omniscient Noise The data is not modified here but the parameters of a device are directly modified. In particular, $\Psi(k, w, H, \mathcal{D}_k, \mathcal{D}, \mathcal{S})$ for clients $k \in S_t$ returns

$$\Psi(k, w, H, \mathcal{D}_k, \mathcal{D}, \mathcal{S}) = -\frac{1}{\sum_{k \in S_t \cap \mathcal{C}} (p_t)_k} \left(2 \sum_{k \in S_t \setminus \mathcal{C}} (p_t)_k \Phi(w^{(t)}, H_k, \mathcal{D}_k) + \sum_{k \in S_t \cap \mathcal{C}} (p_t)_k \Phi(w^{(t)}, H_k, \mathcal{D}_k) \right),$$

Algorithm 11 The FedAvg algorithm

Input: Function F as in Eq. (1) distributed over K devices, fraction c, minibatch size b, number of local epochs n_e, learning rate sequence (γ_t), initial iterate $w^{(0)}$, secure average oracle $\mathcal{A}$

Server executes:

for $t = 1, 2, \cdots$ **do**
  $m \leftarrow \max\{cK, 1\}$
  Let S_t denote a random set of m devices
  for each device $k \in S_t$ **in parallel do**
    $w_k^{(t)} \leftarrow \textit{LocalUpdate}(k, w^{(t)})$
  $w^{(t+1)} \leftarrow \sum_{k \in S_t} n_k w_k^{(t)} / \sum_{k \in S_t} n_k$ using $\mathcal{A}$

function *LocalUpdate*(k, w) ▷ Run on device k
  for $i = 1, \cdots, n_k n_e / b$ **do**
    $B_i \leftarrow$ random subset of $\mathcal{D}_k$ of size b
    $w \leftarrow w - \frac{\gamma_t}{b} \sum_{f \in \mathcal{D}_k} \nabla f(w)$
  return w

such that

$$\mathbb{E}_{k \sim p_t} \Phi_k(w^{(t)}, H_k) = -\mathbb{E}_{k \sim p_t} \Phi(w^{(t)}, H_k, \mathcal{D}_k) \,.$$

See Prop. 6. In other words, the p_t-weighted arithmetic mean of the model parameter is set to be the negative of what it would have other been without the noise. This noise model requires full knowledge of the data and server state, and is adversarial in nature. This is special case of the general approach of model replacement [2].

Implementation details Given a noise level δ, the set of devices which return outlier updates are selected as follows:

- Start with $\mathcal{C} = \varnothing$.
- Sample device k uniformly without replacement and add to $\mathcal{C}$. Stop when $\sum_{k \in C} n_k$ just exceeds δn.

G.2.2 Methods

We compare the following algorithms:

- the FedAvg algorithm [43],
- the RFA algorithm proposed here in Algo. 4,
- the stochastic gradient descent (SGD) algorithm. The local computation Φ_k of device k is simply one gradient step based on a minibatch drawn from $\mathcal{D}_k$, while the aggregation is simply the weighted arithmetic mean.

G.2.3 Hyperparameters

The hyperparameters for each of these algorithms are detailed below.

FedAvg The FedAvg algorithm, recalled in Algo. 11 for reference, requires the following hyperparameters.

- Device Fraction c: This determines the fraction of devices used in each iteration. We use the default $c = 0.1$ for EMNIST and $c = 0.05$ for the Shakespeare dataset, which fall under the recommended range of McMahan et al. [43].
- Batch Size b, and Number of Local Epochs n_e: For the EMNIST dataset, we use $b = 50, n_e = 5$, and for the Shakespeare dataset, we use $b = 10, n_e = 1$. These values were also shown to achieve competitive performance by McMahan et al. [43].
- Learning Rate (γ_t): We use a learning a learning rate scheme $\gamma_t = \gamma_0 C^{\lfloor t/t_0 \rfloor}$, where γ_0 and C were tuned using grid search on validation set (20% held out from the training set) for a fixed time horizon on the noiseless data. The values which gave the highest validation accuracy were used *for all settings* - both noisy and noise-free. The time horizon used was 2000 iterations for the EMNIST linear model, 1000 iterations for the EMNIST ConvNet 200 iterations for Shakespeare LSTM.
- Initial Iterate $w^{(0)}$: Each element of $w^{(0)}$ is initialized to a uniform random variable whose range is determined according to TensorFlow's "glorot_uniform_initializer".

RFA The RFA algorithm, proposed here in Algo. 4 requires the following hyperparameters which FedAvg also requires.

- Device Fraction c, Batch Size b, Number of Local Epochs n_e, Initial Iterate $w^{(0)}$: We use the same settings as for the FedAvg algorithm. Moreover, Sec. G.4.4. presents effect of hyperparameters on RFA .
- Learning Rate (γ_t): We use the *same* learning rate and decay scheme found for FedAvg without further tuning.

In addition, RFA requires the following additional parameters:

- Algorithm: We use the smoothed Weiszfeld's algorithm as default based on the results of Fig. 9a in Sec. G.4.2 which compares it favorably against mirror-prox.
- Smoothing parameter ν: Based on the interpretation that ν guards against division by small numbers, we simply use $\nu = 10^{-6}$ throughout.
- Robust Aggregation Stopping Criterion: The concerns the stopping criterion used to terminate either the smoothed Weiszfeld algorithm or mirror-prox. We use two criteria: an iteration budget and a relative improvement condition - we terminate if a given iteration budget has been extinguished, or if the relative improvement in objective value $|g_\nu(z^{(t)}) - g_\nu(z^{(t)})|/g_\nu(z^{(t+1)}) \leq 10^{-6}$ is small.

G.3 Evaluation Methodology and Other Details

We specify here the quantities appearing on the x and y axes on the plots, as well as other details.

x Axis As mentioned in Section 2, the goal of federated learning is to learn the model with as few rounds of communication as possible. Therefore, we evaluate various methods against the number of rounds of communication, which we measure via the number of calls to a secure average oracle.

Note that FedAvg and SGD require one call to the secure average oracle per outer iteration, while RFA could require several. Hence, we also evaluate performance against the number of outer iterations.

y Axis We are primarily interested in the test accuracy, which measures the performance on unseen data. We also plot the function value F, which is the quantity our optimization algorithm aims to minimize. We call this the train loss. For completeness, the plots also show the train accuracy as well as the test loss.

Since the size of the training and test sets on each client may differ in size, the test loss is measured as

$$F_{\text{test}}(w) = \frac{1}{n} \sum_{k=1}^{K} n_k \left(\frac{1}{|\widehat{\mathcal{D}}_j|} \sum_{f \in \widehat{\mathcal{D}}_j} f(w) \right) ,$$

where $\widehat{\mathcal{D}}_k$ is the test set on device k, and $n_k = |\mathcal{D}_k|$ is the size of the *training set* on device k, and $n = \sum_{k=1}^{K} n_k$ is the total number of training examples. On the other hand, the accuracy is simply measured as the fraction of correct classifications.

Evaluation with Data Corruption In experiments with data noise, while the training is performed on noisy data, we evaluate train and test progress using the noise-free data.

Software We use the package LEAF [11] to simulate the federated learning setting. The models used are implemented in TensorFlow.

Hardware Each experiment was run in a simulation as a single process. The EMNIST linear model experiments were run on two workstations with 126GB of memory, with one equipped with Intel i9 processor running at 2.80GHz, and the other with Intel Xeon processors running at 2.40GHz. Experiments involving neural networks were run either on a 1080Ti or a Titan Xp GPU.

Random runs Each experiment is repeated 5 times with different random seeds, and the solid lines in the plots here represents the mean over these runs, while the shaded areas show the maximum and minimum values obtained in these runs.

G.4 Experimental Results

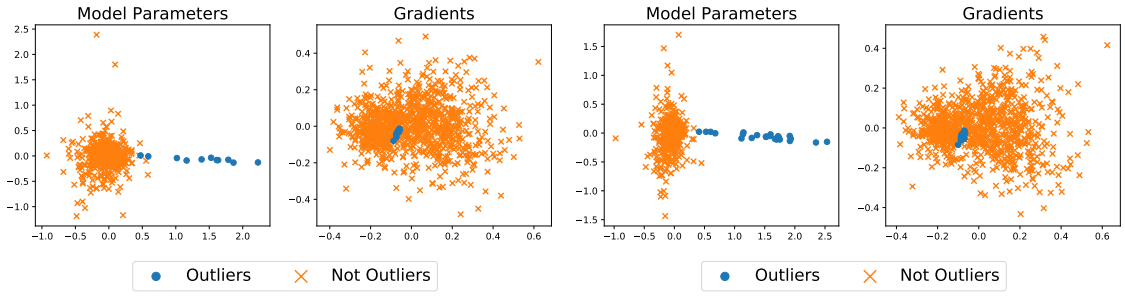
The experimental results are divided into distinct parts: Section G.4.1 visualizes the principal components of parameter vectors and gradients, Section G.4.2 compares different algorithms to compute the geometric median, while Section G.4.3 compares the RFA algorithm to its competitors. Lastly, Section G.4.4 studies the effect of the hyperparameters.

G.4.1 Visualizing Aggregation of Model Parameters vs. Gradients

In this section we visualize the first two principal components of the model parameters and gradients.

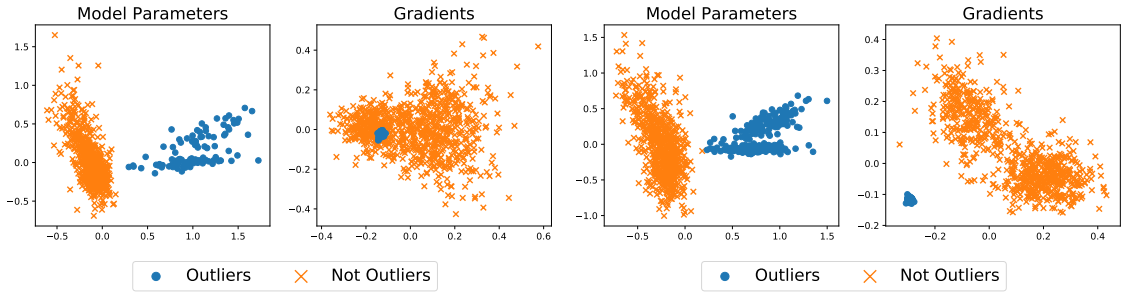
We freeze FedAvg at after 100 iterations on the EMNIST linear model. We run *LocalUpdate* on each of the device and collect their model parameters vectors. We also compute the gradient on 10% of the data on each device and collect all the gradients vectors. We perform principal component analysis (PCA) on each of these collections and visualize the first two principal components in Fig. 8, for different levels of data noise.

This visualization is useful to note the difference between aggregation of model parameters, the approach taken by this paper, versus the aggregation of gradients, which is the approach taken by most previous work on robust aggregation for distributed learning.



(a) $\delta = 0.01$.

(b) $\delta = 0.02$.



(c) $\delta = 0.1$.

(d) $\delta = 0.25$.

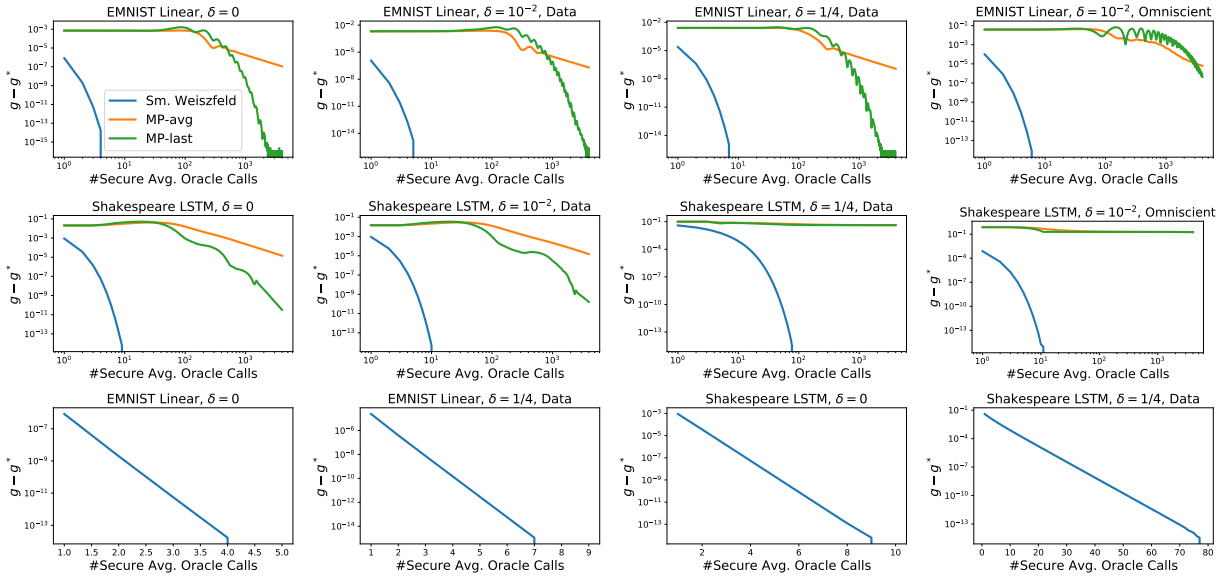
Figure 8: Visualization of the first two principal components of model parameters and gradients for a linear model on the EMNIST dataset under data noise at different noise levels δ. The PCA projection of parameter/gradient vectors of the devices with corrupted data are denoted by blue dots while those of devices with nominal data are denoted by orange 'x'.

G.4.2 Algorithms to Compute the Geometric Median

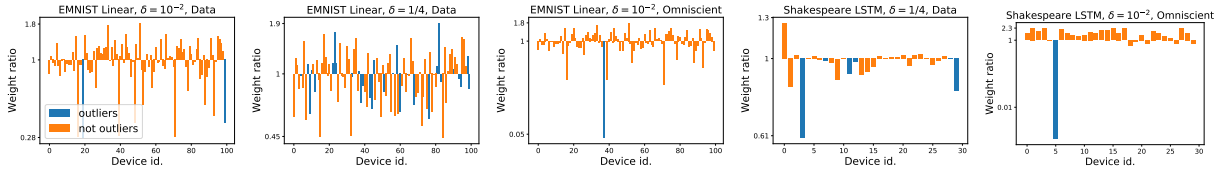
For each of these models, we freeze FedAvg at a certain iteration and try different robust aggregation algorithms. The omniscient noise was applied manually to to parameters obtained from a run with no noise as FedAvg diverged in this case (cf. Fig. 11-13).

First, we compare the performance of the smoothed Weiszfeld algorithm and mirror-prox in Fig. 9a. We plot both the progress of the averaged iterate $\overline{z}^{(t)}$ (MP-avg) as well as $\widehat{z}^{(t)}$ (MP-last). The smoothing Weiszfeld algorithm uses $\nu = 10^{-6}$ but identical performance was obtained for any $\nu < 10^{-2}$ (cf. Cor. 25). We find that the smoothed Weiszfeld algorithm is extremely fast, converging exactly to the smoothed geometric median in a few passes. In fact, the smoothed Weiszfeld algorithm displays (local) linear convergence, as evidenced by the straight line in log scale (third row). Meanwhile, the straight line in the log-log plots (first two rows) highlights the sublinear convergence of mirror-prox. In light of this observation, we only use the smoothed Weiszfeld algorithm going forward. Further, we also maintain a strict iteration budget of 3 iterations. This choice is also justified in hindsight by the results of Fig. 15.

Next, we visualize the weights assigned by the geometric median to the outliers. Note that the smoothed geometric median $w_1, \cdots, w_m$ is some convex combination $\sum_{k=1}^{m} \beta_k w_k$. This weight β_k of w_k is a measure of the influence of w_k on the aggregate. We plot in Fig. 9b the ratio β_k / α_k for each device k, where α_k is its weight in the arithmetic mean and β_k is obtained by running the smoothed Weiszfeld's algorithm to convergence. We expect this ratio to be smaller for worse outliers and ideally zero for obvious outliers. We find that the smoothed geometric median does indeed assign lower weights to the outliers, while only accessing the points via a secure average oracle.



(a) Comparison of convergence of the smoothed Weiszfeld algorithm and mirror-prox for robust aggregation.



(b) Visualization of the re-weighting of points in the robust aggregate.

Figure 9: Performance of robust aggregation algorithms.

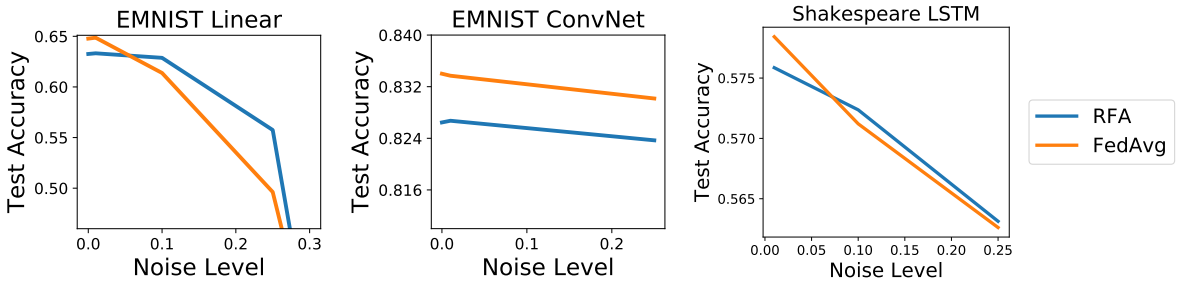


Figure 10: Comparison of robustness of RFA and FedAvg under data noise.

G.4.3 Comparison of RFA with other methods

Next, we study the effect of the level of noise on the robustness of RFA and FedAvg . We study two aspects of robustness: the maximum test performance and the performance over the learning process.

We start with the former. Fig. 10 plots the maximum test accuracy of a single run, averaged over five different runs. Each run was allowed to run to convergence, unless a communication budget was exceeded. Only RFA on (EMNIST, ConvNet) failed to converge within this budget. We observe that RFA gives us improved robustness in the case of the linear model, while simple FedAvg is better for the ConvNet model [cf. 60]. On the LSTM model, both $FedAvg$ and RFA give identical curves.

Next, we plot the performance of competing methods versus the number of calls to the secure average oracle and versus the number of outer iterations (the counter t in Algo. 4 and Algo. 11) in Figures 11 to 13.

Since FedAvg converged by round 2000 in Fig. 12, we extend the curve up to end of the time horizon under consideration.

We note that FedAvg is better in the high signal-to-noise case of $\delta = 0$ or $\delta = 10^{-2}$ under data noise when measured in the number of calls to the secure average oracle. However, it is identical in performance to RFA when measured in terms of the number of outer iterations. Further, both FedAvg and SGD diverge under omniscient noise, as pointed out by Prop. 6. Fig. 14 shows that RFA converges even under omniscient noise.

Furthermore, we observe that RFA is qualitatively more stable than FedAvg in its behavior over time in the high data noise setting of $\delta = 1/4$.

G.4.4 Hyperparameter Study

Here, we study the effect of the various hyperparameters on RFA .

Effect of Iteration Budget of the Smoothed Weiszfeld Algorithm We study the effect of the iteration budget of the smoothed Weiszfeld algorithm in RFA . in Fig. 15. We observe that a low communication budget is faster in the regime of high signal-to-noise ratio, while more iterations work better in the low signal-to-noise ratio regime. We used a budget of 3 calls to the secure average oracle to trade-off between the two scenarios in Sec. G.4.3.

Effect of Number of Devices per Iteration Figure 16 plots the performance of RFA against the number K of devices chosen per round. We observe the following: in the regime of high signal-to-noise ratio, good performance is achieved by selecting 50 devices per round (5%), where as 10 devices per round (1%) is not enough. On the other hand, in high noise regimes, we see the benefit of choosing more devices per round, as a few runs with 10 or 50 devices per round with omniscient noise at 25% diverged.

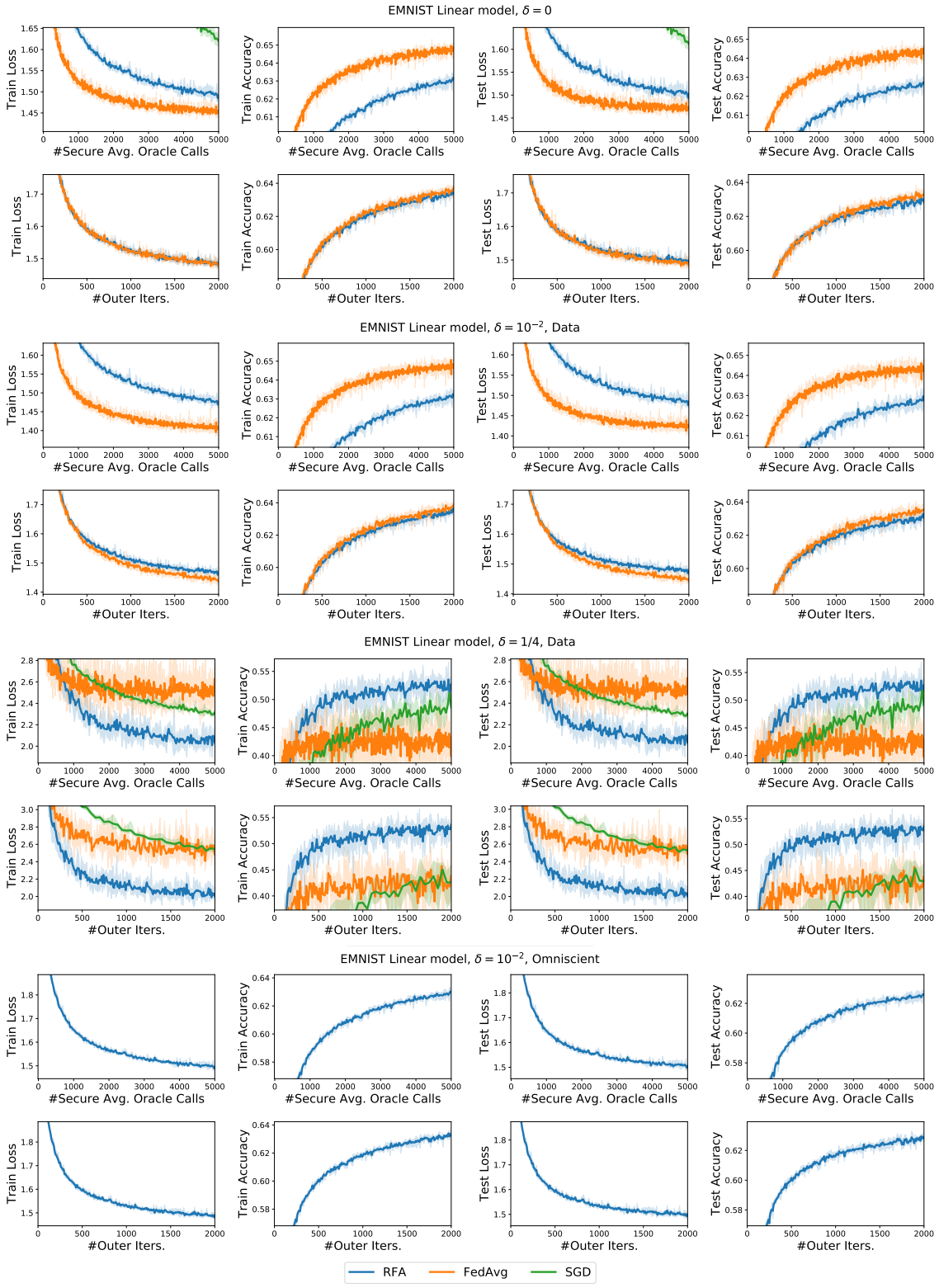


Figure 11: Comparison of methods plotted against number of calls to the secure average oracle or outer iterations on the EMNIST dataset with a linear model.

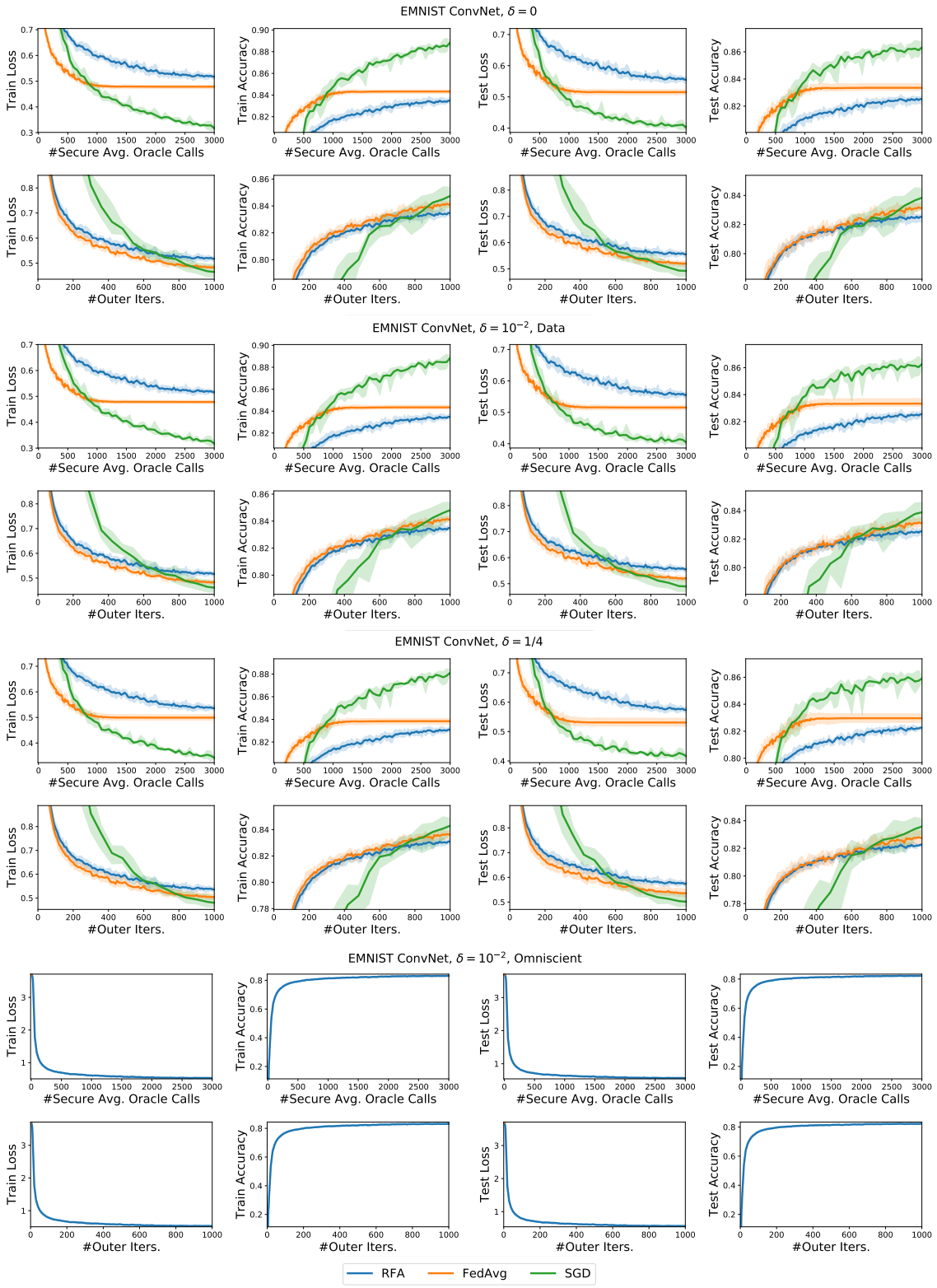


Figure 12: Comparison of methods plotted against number of calls to the secure average oracle or outer iterations on the EMNIST dataset with a convolutional neural network.

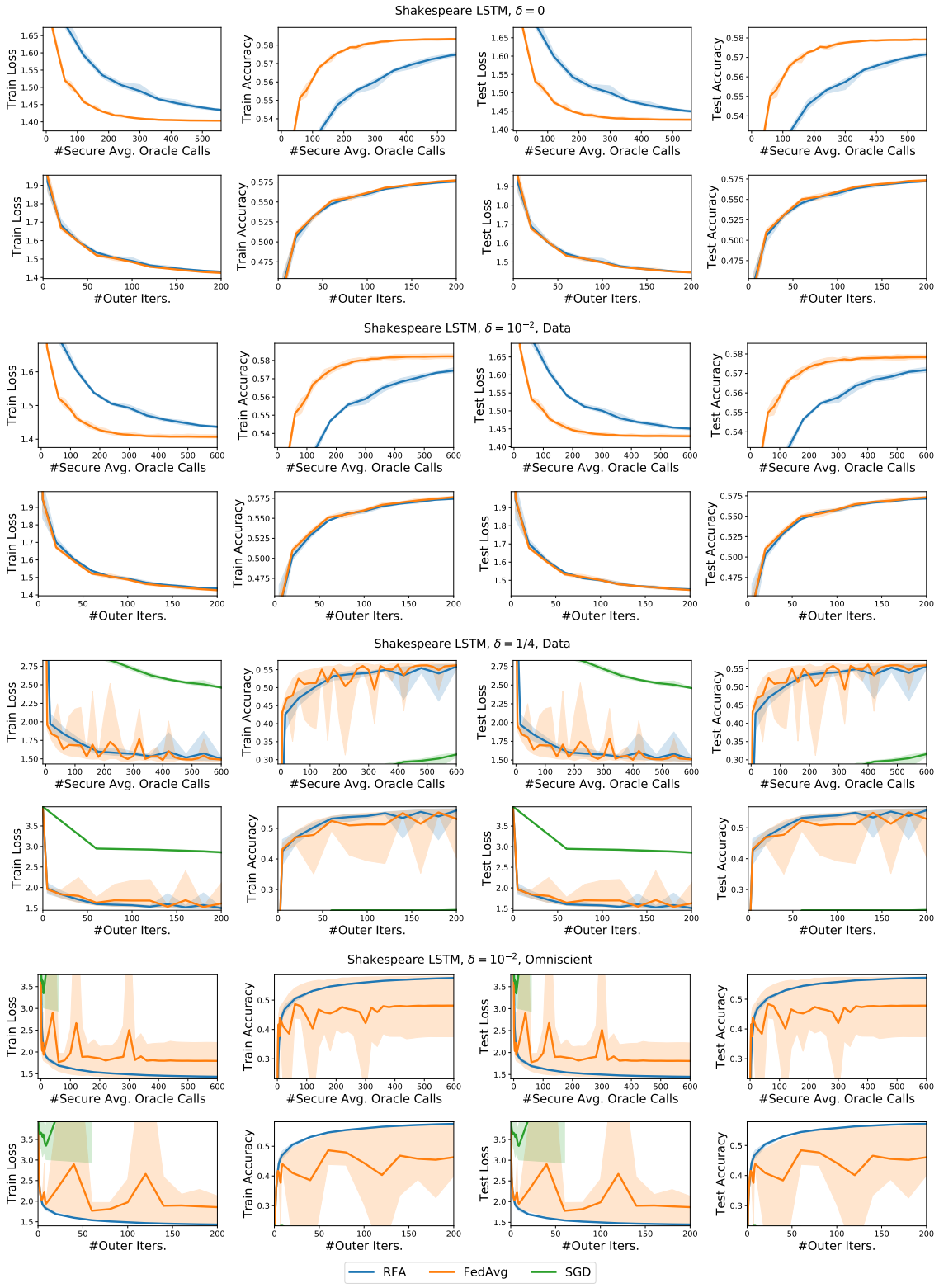


Figure 13: Comparison of methods plotted against number of calls to the secure average oracle or outer iterations on the Shakespeare dataset with an LSTM model.

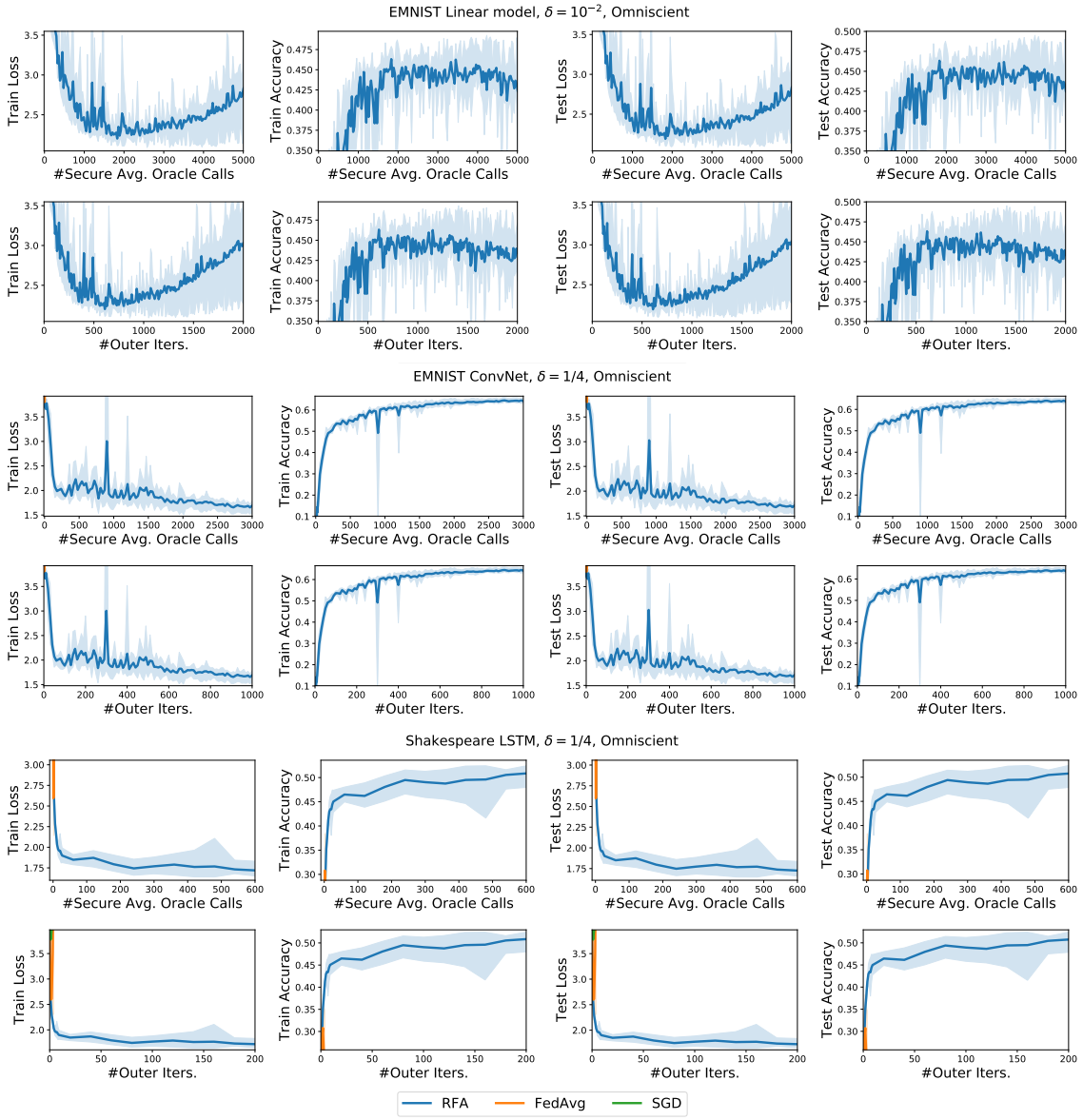


Figure 14: Comparison of methods plotted against number of calls to the secure average oracle or outer iterations for omniscient noise with $\delta = 1/4$.

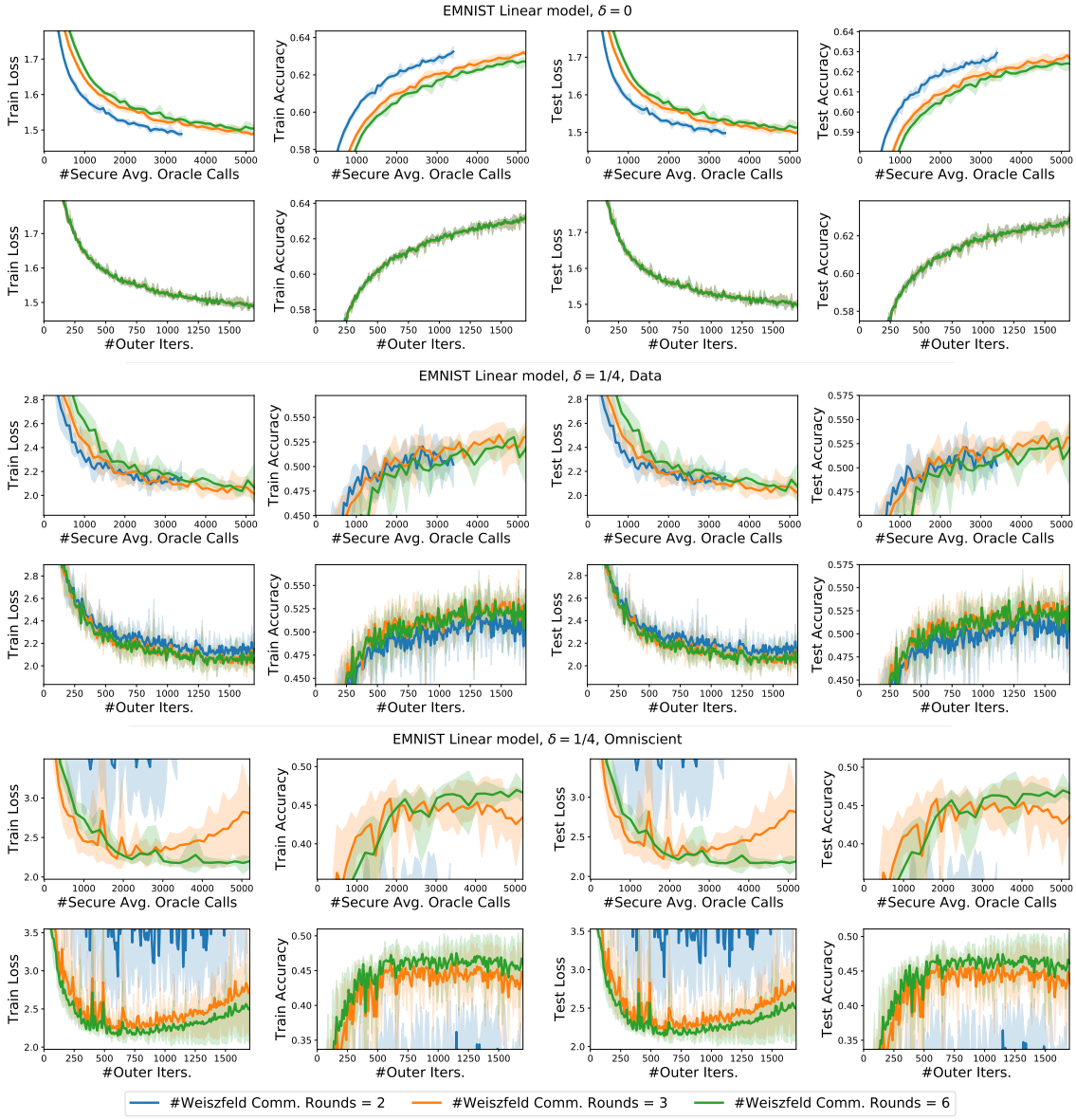


Figure 15: Hyperparameter study: effect of the maximum number of the communication budget on the smoothed Weiszfeld algorithm in RFA on the EMNIST dataset with a linear model.

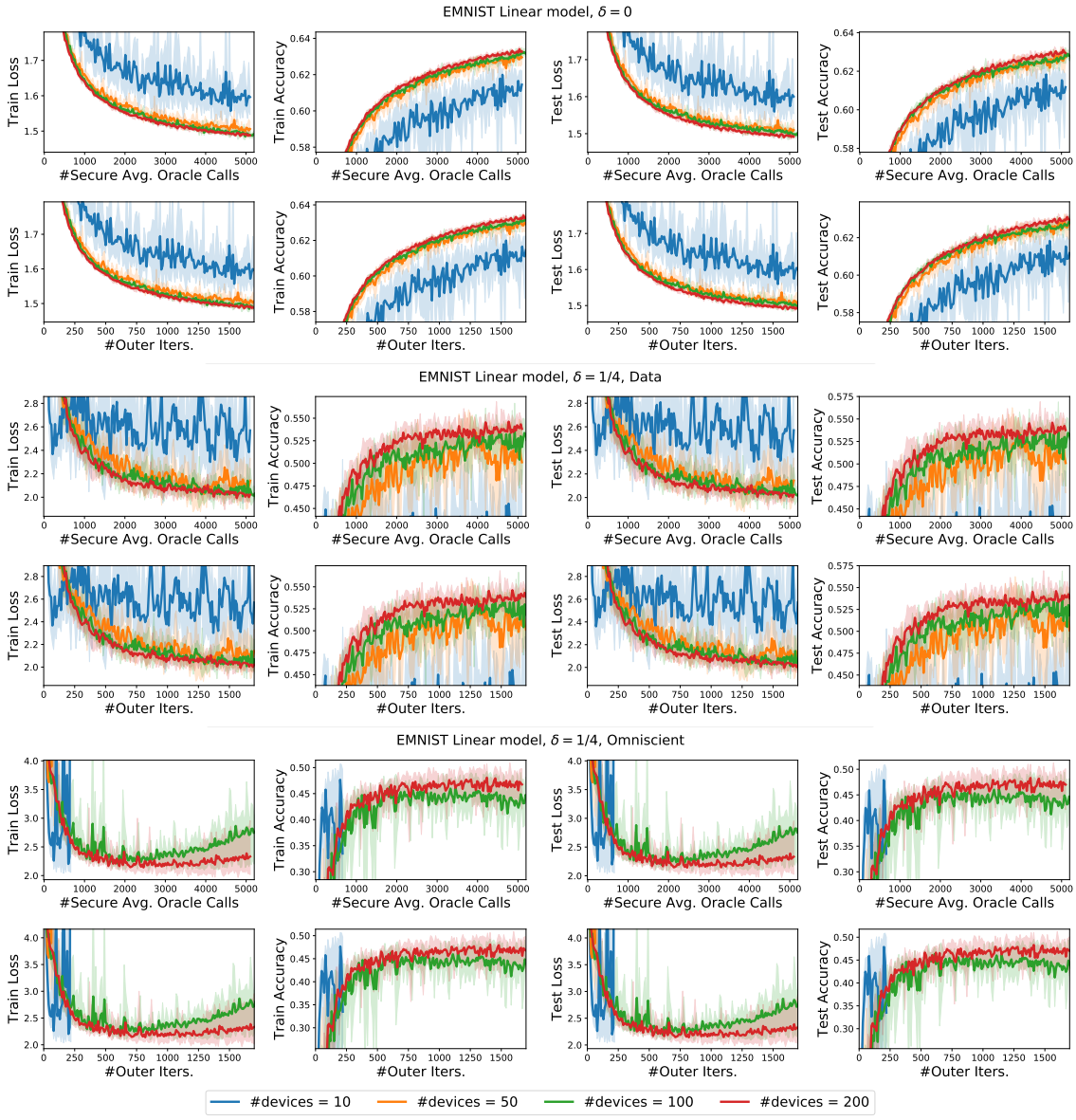


Figure 16: Hyperparameter study: effect of the number of selected client devices per round in RFA on the EMNIST dataset with a linear model.