FedSplit: an algorithmic framework for fast federated optimization

Reese Pathak
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720
pathakr@berkeley.edu

Martin J. Wainwright
Department of Electrical Engineering and Computer Sciences
Department of Statistics
University of California, Berkeley
Berkeley, CA 94720
wainwrig@berkeley.edu

Abstract

Motivated by federated learning, we consider the hub-and-spoke model of distributed optimization in which a central authority coordinates the computation of a solution among many agents while limiting communication. We first study some past procedures for federated optimization, and show that their fixed points need not correspond to stationary points of the original optimization problem, even in simple convex settings with deterministic updates. In order to remedy these issues, we introduce FedSplit, a class of algorithms based on operator splitting procedures for solving distributed convex minimization with additive structure. We prove that these procedures have the correct fixed points, corresponding to optima of the original optimization problem, and we characterize their convergence rates under different settings. Our theory shows that these methods are provably robust to inexact computation of intermediate local quantities. We complement our theory with some experiments that demonstrate the benefits of our methods in practice.

1 Introduction

Federated learning is a rapidly evolving application of distributed optimization for learning problems in large-scale networks of remote clients [13]. These systems present new challenges, as they are characterized by heterogeneity in computational resources, data across a large, multi-agent network, unreliable communication, and privacy constraints due to sensitive client data [15].

Although distributed optimization has a rich history and extensive literature (e.g., see the sources [2, 4, 8, 28, 14, 23] and references therein), renewed interest due to federated learning has led to a flurry of recent work in the area. Notably, McMahan et al. [17] introduced the FedSGD and FedAvg algorithms, by adapting the classical stochastic gradient method to the federated setting, considering the possibility that clients may fail and may only be subsampled on each round of computation. Another recent proposal, FedProx, attempted to mitigate potential device heterogeneity issues by applying averaged proximal updates to solve federated minimization problems. Currently, a general convergence theory of these methods is lacking. Moreover, practitioners have documented failures of convergence in certain settings (*e.g.*, see Figure 3 and related discussion in the work [17]).

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Our contributions: The first contribution of this paper is to analyze some past procedures, and show that even in the favorable setting of deterministic updates (i.e., no stochastic approximation used), these methods typically fail to preserve solutions of the original optimization problem as fixed points. More precisely, even when these methods do converge, the resulting fixed point need *not* correspond to an optimal solution of the desired federated learning problem. Since the stochastic variants implemented in practice are approximate versions of the underlying deterministic procedures, this implies these methods also fail to preserve the correct fixed points in general.

With the motivation of rectifying this undesirable feature, our second contribution is to introduce a family of federated optimization algorithms, which we call `FedSplit`, that do preserve the correct fixed points for distributed optimization problems of the form

$$\text{minimize} \quad F(x) := \sum_{j=1}^{m} f_j(x), \tag{1}$$

where $f_j \colon \mathbf{R}^d \to \mathbf{R}$ are the clients' cost functions for variable $x \in \mathbf{R}^d$. In machine learning applications, the vector $x \in \mathbf{R}^d$ is a parameter of a statistical model. Our procedure and analysis builds on a long line of work relating optimization with monotone operators and operator splitting techniques [4, 26, 7, 1]. In this paper, we focus on the case when f_j are convex functions with Lipschitz continuous gradient [24].

2 Existing algorithms and their fixed points

We focus our discussion on deterministic analogues of two recently proposed procedures—namely, `FedSGD` [17] and `FedProx` [16]. For analysis, it is useful to introduce the equivalent, consensus reformulation [4] of the distributed problem (1):

$$\begin{array}{ll} \text{minimize} & F(x) := \sum_{j=1}^{m} f_j(x_j) \\ \text{subject to} & x_1 = x_2 = \cdots = x_m. \end{array} \tag{2}$$

2.1 Federated gradient algorithms

The recently proposed `FedSGD` method [17] is based on a multi-step projected stochastic gradient method for solving the consensus problem. For our analysis we consider the obvious deterministic version of this algorithm, which replaces the stochastic gradient by the full gradient. Formally, given a stepsize $s > 0$, define the *gradient mappings*

$$G_j(x) := x - s\nabla f_j(x) \qquad \text{for } j = 1, \ldots, m. \tag{3}$$

For a given integer $e \geqslant 1$, we define G_j^e as the e-fold composition of G_j and G_j^0 as the identity operator on $\mathbf{R}^d$. The `FedGD`(s, e) algorithm from initialization $x^{(1)}$ obeys the recursion for $t = 1, 2, \ldots$:

$$x_j^{(t+1/2)} := G_j^e(x_j^{(t)}), \qquad \text{for } j \in [m] := \{1, 2, \ldots, m\}, \text{ and} \tag{4a}$$

$$x_j^{(t+1)} := \overline{x}^{(t+1/2)}, \qquad \text{for } j \in [m]. \tag{4b}$$

Recall that $\overline{x}^{(t+1/2)} = \frac{1}{m}\sum_{j=1}^{m} x_j^{(t+1/2)}$ is the block average. The following result characterizes the fixed points of this procedure.

Proposition 1. *For any $s > 0$ and $e \geqslant 1$, the sequence $\{x^{(t)}\}_{t=1}^{\infty}$ generated by the `FedGD`(s, e) algorithm in equation* (4) *has the following properties: (a) if $x^{(t)}$ is convergent, then the local variables $x_j^{(t)}$ share a common limit $x^\star$ such that $x_j^{(t)} \to x^\star$ as $t \to \infty$ for $j \in [m]$; (b) any such limit $x^\star$ satisfies the fixed point relation*

$$\sum_{i=1}^{e}\sum_{j=1}^{m} \nabla f_j(G_j^{i-1}(x^\star)) = 0. \tag{5}$$

The proof of this claim, as well as all other claims in the paper, are deferred to Appendix A of the supplement.

Unpacking this claim slightly, suppose first that a single update is performed between communications, so $e = 1$. In this case, we have $\sum_{i=1}^{e} \nabla f_j(G_j^{i-1}(x^\star)) = \nabla f_j(x^\star)$, so that if $x^{(t)}$ has a limit x, it satisfies the relations

$$x_1 = x_2 = \cdots = x_m \quad \text{and} \quad \sum_{j=1}^{m} \nabla f_j(x_j) = 0.$$

Consequently, provided that the losses f_j are convex, Proposition 1 implies that the limit of the sequence $x^{(t)}$, when it exists, is a minimizer of the consensus problem (2).

On the other hand, when $e > 1$, a limit of the iterate sequence $x^{(t)}$ must satisfy equation (5), which in general causes the method to have limit points which are not minimizers of the consensus problem. We give a concrete example in Section 2.3.

2.2 Federated proximal algorithms

Another recently proposed algorithm is `FedProx` [16], which can be seen as a distributed method loosely based on the classical proximal point method [24]. For a given stepsize $s > 0$, the *proximal operator* of a function $f \colon \mathbf{R}^d \to \mathbf{R}$ and its associated optimal value, the *Moreau envelope* of f, are given by [19, 24, 25, chap. 1.G]:

$$\mathbf{prox}_{sf}(z) := \operatorname*{arg\,min}_{x \in \mathbf{R}^d} \left\{ f(x) + \frac{1}{2s}\|z - x\|^2 \right\} \quad \text{and} \quad M_{sf}(z) := \inf_{x \in \mathbf{R}^d} \left\{ f(x) + \frac{1}{2s}\|z - x\|^2 \right\}.$$

We remark that when f is convex, the existence of such a (unique) minimizer for the problem implied by the proximal operator is immediate.

With these definitions in place, we can now study the behavior of the `FedProx` method [16]. We again consider a deterministic version of `FedProx`, in which we remove any inaccuracies introduced by stochastic approximation. For a given initialization $x^{(1)}$, for $t = 1, 2, \ldots$:

$$x_j^{(t+1/2)} := \mathbf{prox}_{sf_j}(x_j^{(t)}), \qquad \text{for } j \in [m], \text{ and} \tag{6a}$$

$$x_j^{(t+1)} := \overline{x}^{(t+1/2)}, \qquad \text{for } j \in [m]. \tag{6b}$$

The following result characterizes the fixed points of this method.

Proposition 2. *For any stepsize $s > 0$, the sequence $\{x^{(t)}\}_{t=1}^{\infty}$ generated by the `FedProx` algorithm (see equations* (6a) *and* (6b)*) has the following properties: (a) if $x^{(t)}$ is convergent then, the local variables $x_j^{(t)}$ share a common limit $x^\star$ such that $x_j^{(t)} \to x^\star$ as $t \to \infty$ for each $j \in [m]$; (b) the limit $x^\star$ satisfies the fixed point relation*

$$\sum_{j=1}^{m} \nabla M_{sf_j}(x^\star) = 0. \tag{7}$$

Hence, we see that this algorithm has fixed points that will be a zero of the sum of the gradients of the Moreau envelopes M_{sf_j}, rather than a zero of the sum of the gradients of the functions f_j themselves. When $m > 1$, these fixed point relations are, in general, different.

It is worth noting a very special case in which `FedGD` and `FedProx` will preserve the correct fixed points, even when $e > 1$. In particular, suppose *all* of local cost functions share a common minimizer $x^\star$, so that $\nabla f_j(x^\star) = 0$ for $j \in [m]$. Under this assumption, we have $G_j(x^\star) = x^\star$ all $j \in [m]$, and hence by arguing inductively, we have $G_j^i(x^\star) = x^\star$ for all $i \geqslant 1$. Additionally recall that the minimizers of f_j and M_{sf_j} coincide. Consequently, the fixed point relations (5) and (7) corresponding to `FedGD` and `FedProx` respecively, are both equivalent to the optimality condition for the federated problem. However, we emphasize this condition is *not realistic* in practice: if the optima of f_j are exactly (or even approximately) the same, there would be little point in sharing data between devices by solving the federated learning problem. In contrast, the `FedSplit` algorithm presented in the next section retains correct fixed points for general federated learning problems without making such unrealistic, additional assumptions.

2.3 Example: Incorrectness on a least squares problem

We illustrate these non-convergence results by specializing to least squares and carrying out a simulation study on a synthetic least squares dataset. For $j = 1, \ldots, m$, suppose that we are given a design matrix $A_j \in \mathbf{R}^{n_j \times d}$ and a response vector $b_j \in \mathbf{R}^{n_j}$. The least squares regression problem defined by all the devices takes the form

$$\text{minimize} \quad F(x) := \frac{1}{2} \sum_{j=1}^{m} \|A_j x - b_j\|^2. \tag{8}$$

This problem is a special case of our general problem (1) with $f_j(x) = (1/2)\|A_j x - b_j\|^2$ for all j. When A_j are full rank, the solution to this problem is unique and given by

$$x^\star_{\text{ls}} = \left(\sum_{j=1}^{m} A_j^\mathsf{T} A_j \right)^{-1} \sum_{j=1}^{m} A_j^\mathsf{T} b_j. \tag{9}$$

Following Proposition 2, it is easy to verify that `FedProx` has fixed points of the form

$$x^\star_{\texttt{FedProx}} = \left(\sum_{j=1}^{m} \left\{ I - (I + sA_j^\mathsf{T} A_j)^{-1} \right\} \right)^{-1} \left(\sum_{j=1}^{m} (A_j^\mathsf{T} A_j + (1/s)I)^{-1} A_j^\mathsf{T} b_j \right).$$

Following Proposition 1, it is easy to verify that `FedGD` has fixed points of the form[1]

$$x^\star_{\texttt{FedGD}} = \left(\sum_{j=1}^{m} A_j^\mathsf{T} A_j \left\{ \sum_{k=0}^{e-1} (I - sA_j^\mathsf{T} A_j)^k \right\} \right)^{-1} \left(\sum_{j=1}^{m} \left\{ \sum_{k=0}^{e-1} (I - sA_j^\mathsf{T} A_j)^k \right\} A_j^\mathsf{T} b_j \right). \tag{10}$$

Therefore the previous three displays show that in general, when $m > 1$ and $e > 1$—that is, with more than one client, and more than one local update between communication rounds—we have $x^\star_{\texttt{FedProx}} \neq x^\star_{\text{ls}}$ and $x^\star_{\texttt{FedGD}} \neq x^\star_{\text{ls}}$. Therefore, we see that `FedProx` and `FedGD` do not have the correct fixed points, even with idealized deterministic updates.

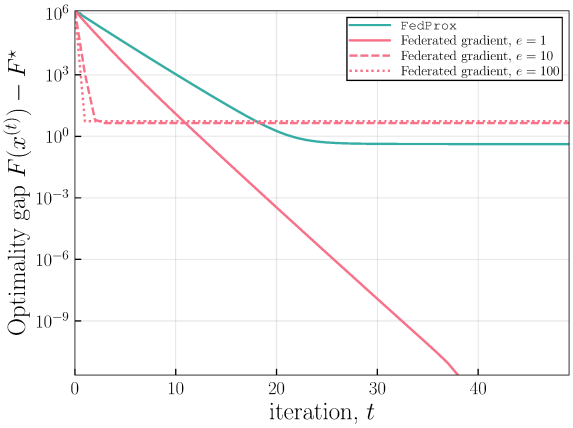


Figure 1. Plots of $F(x^{(t)}) - F^\star$ versus iteration number t for a least-squares problem (8). This measures the difference between the optimal value and the value on round t given by an algorithm.

Figure 1 shows the results of applying the (deterministic) versions of `FedProx` and `FedSGD`, with varying numbers of local epochs $e \in \{1, 10, 100\}$ for the least squares minimization problem (8). As expected, we see that `FedProx` and multi-step, deterministic `FedSGD` fail to converge to the correct fixed point for this problem. Although the presented deterministic variant of `FedSGD` will converge when a single local gradient step is taken between communication rounds (*i.e.*, when $e = 1$), we see that it also does not converge to the optimal solution as soon as $e > 1$. See Appendix B.1 of the supplement for additional details on this simulation study.

[1]Here we assume that $s > 0$ is small enough so that $\|I - sA_j^\mathsf{T} A_j\|_{\text{op}} < 1$, which ensures convergence.

3 FedSplit and convergence guarantees

We now turn to the description of a framework that allows us to provide a clean characterization of the fixed points of iterative algorithms and to propose algorithms with convergence guarantees. Throughout our development, we assume that each function $f_j : \mathbf{R}^d \to \mathbf{R}$ is convex and differentiable.

3.1 An operator-theoretic view

We begin by recalling the consensus formulation (2) of the problem in terms of a block-partitioned vector $x = (x_1, \ldots, x_m) \in (\mathbf{R}^d)^m$, the function $F : (\mathbf{R}^d)^m \to \mathbf{R}$ given by $F(x) := \sum_{j=1}^m f_j(x_j)$, and the constraint set $E := \{x \mid x_1 = x_2 = \cdots = x_m\}$ is the feasible subspace for problem (2). By appealing to the first-order optimality conditions for the problem (2), it is equivalent to find a vector $x \in (\mathbf{R}^d)^m$ such that $\nabla F(x)$ belongs to the normal cone of the constraint set E, or equivalently such that $\nabla F(x) \in E^\perp$. Equivalently, if we define a set-valued operator $\mathcal{N}_E$ as

$$\mathcal{N}_E(x) := \begin{cases} E^\perp, & x_1 = x_2 = \cdots = x_m, \\ \emptyset, & \text{else} \end{cases} \tag{11}$$

then it is equivalent to find a vector $x \in (\mathbf{R}^d)^m$ that satisfies the inclusion condition

$$0 \in \nabla F(x) + \mathcal{N}_E(x). \tag{12}$$

where $\nabla F(x) = (\nabla f_1(x_1), \ldots, \nabla f_m(x_m))$.

When the loss functions $f_j : \mathbf{R}^d \to \mathbf{R}$ are convex, both ∇F and $\mathcal{N}_E$ are monotone operators on $(\mathbf{R}^d)^m$ [1]. Thus, the display (12) is a *monotone inclusion problem*. Methods for solving monotone inclusions have a long history of study within the applied mathematics and optimization literatures [26, 7]. We now use this framework to develop and analyze algorithms for solving the federated problems of interest.

3.2 Splitting procedures for federated optimization

We now describe a method, derived from splitting the inclusion relation, whose fixed points do correspond with global minima of the distributed problem. It is an instantiation of the Peaceman-Rachford splitting [20], which we refer to as the FedSplit algorithm in this distributed setting.

Algorithm 1 [FedSplit] *Splitting scheme for solving federated problems of the form* (1)

Given initialization $x \in \mathbf{R}^d$, proximal solvers $\texttt{prox\_update}_j : \mathbf{R}^d \to \mathbf{R}^d$

Initialize $x^{(1)} = z_1^{(1)} = \cdots = z_m^{(1)} = x$

for $t = 1, 2, \ldots$:

1. **for** $j = 1, \ldots, m$:
   a. *Local prox step:* set $z_j^{(t+1/2)} = \texttt{prox\_update}_j(2x^{(t)} - z_j^{(t)})$
   b. *Local centering step:* set $z_j^{(t+1)} = z_j^{(t)} + 2(z_j^{(t+1/2)} - x^{(t)})$
   end for
2. *Compute global average:* set $x^{(t+1)} = \overline{z}^{(t+1)}$.

end for

Thus, the FedSplit procedure maintains a parameter vector $z_j^{(t)} \in \mathbf{R}^d$ for each device $j \in [m]$. The central server maintains a parameter vector $x^{(t)} \in \mathbf{R}^d$, which collects averages of the parameter estimates at each machine. The local update at device j is defined in terms of a proximal solver $\texttt{prox\_update}_j(\cdot)$, which typically be approximate proximal updates $\texttt{prox\_update}_j(x) \approx \mathbf{prox}_{sf_j}(x)$, uniformly in $x \in \mathbf{R}^d$ for a suitable stepsize $s > 0$. We make the sense of this approximation precise when we state our convergence results in Section 3.3. An advantage to FedSplit is that unlike FedGD and FedProx, it has the correct fixed points for the distributed problem.

Proposition 3. *Suppose for some $s > 0$, $\texttt{prox\_update}_j(\cdot) = \mathbf{prox}_{sf_j}(\cdot)$, for all j. Suppose that $z^\star = (z_1^\star, \ldots, z_m^\star)$ is a fixed point for the* FedSplit *procedure, meaning that*

$$z_j^\star = z_j^\star + 2\left(\mathbf{prox}_{sf_j}(2\overline{z^\star} - z_j^\star) - \overline{z^\star}\right), \qquad \text{for all } j \in [m]. \tag{13}$$

Then the average $x^\star := \frac{1}{m}\sum_{j=1}^m z_j^\star$ is optimal: $\sum_{j=1}^m f_j(x^\star) = \inf_{x \in \mathbf{R}^d} \sum_{j=1}^m f_j(x)$.

3.3 Convergence results

In this section, we give convergence guarantees for the `FedSplit` procedure in Algorithm 1 under exact and inexact proximal operator implementations.

Strongly convex and smooth losses We begin by considering the case when the losses $f_j : \mathbf{R}^d \to \mathbf{R}$ are ℓ_j-strongly convex and L_j-smooth. We define

$$\ell_* := \min_{j=1,\ldots,m} \ell_j, \quad L^* := \max_{j=1,\ldots,m} L_j, \quad \text{and} \quad \kappa := \frac{L^*}{\ell_*}. \tag{14}$$

Note that κ corresponds to the induced condition number of our federated problem (2).

The following result demonstrates that in this setting, our method enjoys geometric convergence to the optimum, even with inexact proximal implementations.

Theorem 1. *Consider the* `FedSplit` *algorithm with possibly inexact proximal implementations,*

$$\|\texttt{prox\_update}_j(z) - \mathbf{prox}_{sf_j}(z)\| \leqslant b \qquad \textit{for all } j \textit{ and all } z \in \mathbf{R}^d, \tag{15}$$

and with stepsize $s = 1/\sqrt{\ell_* L^*}$. Then for any initialization, the iterates satisfy

$$\|x^{(t+1)} - x^\star\| \leqslant \left(1 - \frac{2}{\sqrt{\kappa}+1}\right)^t \frac{\|z^{(1)} - z^\star\|}{\sqrt{m}} + (\sqrt{\kappa}+1)b, \qquad \textit{for all } t = 1, 2, \ldots. \tag{16}$$

We now discuss some aspects of Theorem 1.

Exact proximal evaluations: In the special (albeit unrealistic) case when the proximal evaluations are exact, the uniform bound (15) holds with $b = 0$. Consequently, given some initialization $z^{(1)}$, if we want ε-accuracy, meaning $\|x^{(T)} - x^\star\| \leqslant \varepsilon$, we see that this occurs as soon as T exceeds

$$T(\varepsilon, \kappa) = O(1)\Big\{\sqrt{\kappa}\log\left(\frac{\|z^{(1)} - z^\star\|}{\varepsilon\sqrt{m}}\right)\Big\}$$

iterations of the overall procedure. Here $O(1)$ denotes a universal constant.

Approximate proximal updates by gradient steps: In practice, the `FedSplit` algorithm will be implemented using an approximate prox-solver. Recall that the proximal update at device j at round t takes the form:

$$\mathbf{prox}_{sf_j}(x_j^{(t)}) = \operatorname*{arg\,min}_{u \in \mathbf{R}^d} \big\{ \underbrace{sf_j(u) + \frac{1}{2}\|u - x_j^{(t)}\|_2^2}_{h_j(u)} \big\}.$$

A natural way to compute an approximate minimizer is to run e rounds of gradient descent on the function h_j. Concretely, at round t, we initialize the gradient method with the initial point $u^{(1)} = x_j^{(t)}$, and run gradient descent on h_j with a stepsize α, thereby generating the sequence

$$u^{(t+1)} = u^{(t)} - \alpha\nabla h_j(u^{(t)}) \;=\; u^{(t)} - \alpha s\nabla f_j(u^{(t)}) + \big(u^{(t)} - x_j^{(t)}\big) \tag{17}$$

We define $\texttt{prox\_update}_j(x_j^{(t)})$ to be the output of this procedure after e steps.

Corollary 1 (`FedSplit` convergence with inexact proximal updates)**.** *Consider the* `FedSplit` *procedure run with proximal stepsize $s = \frac{1}{\sqrt{\ell_* L^*}}$, and using approximate proximal updates based on e rounds of gradient descent with stepsize $\alpha = (1 + s\frac{\ell_* + L^*}{2})^{-1}$ initialized (in round t) at the previous iterate $x_j^{(t)}$. Then the the bound* (15) *holds at round t with error at most*

$$b \leqslant \big(1 - \frac{1}{\sqrt{\kappa}+1}\big)^e \, \|x_j^{(t)} - \mathbf{prox}_{sf_j}(x_j^{(t)})\|_2. \tag{18}$$

Given the exponential decay in the number of rounds e exhibited in the bound (18), in practice, it suffices to take a relatively small number of gradient steps. For instance, in our experiments to be reported in Section 4, we find that $e = 10$ suffices to match the exact proximal updates. This inexact proximal update could also be implemented with a gradient method and backtracking line search [5].

Smooth but not strongly convex losses We now consider the case when $f_j : \mathbf{R}^d \to \mathbf{R}$ are L_j-smooth and convex, but not necessarily strongly convex. In this case, the consensus objective $F(z) = \sum_{j=1}^m f_j(z_j)$ is an L^*-smooth function on the product space $(\mathbf{R}^d)^m$.[2]

Our approach to solving such a problem is to apply the `FedSplit` procedure to a suitably regularized version of the original problem. More precisely, given some initial vector $x^{(1)} \in \mathbf{R}^d$ and regularization parameter $\lambda > 0$, let us define the function

$$F_\lambda(z) := \sum_{j=1}^m \Big\{ f_j(z_j) + \frac{\lambda}{2m} \|z_j - x^{(1)}\|^2 \Big\}. \tag{19}$$

We see that $F_\lambda : (\mathbf{R}^d)^m \to \mathbf{R}$ is a λ-strongly convex and $L_\lambda^* = (L^* + \lambda)$-smooth function. The next result shows that for any $\varepsilon > 0$, minimizing the function F_λ up to an error of order ε, using a carefully chosen λ, yields an ε-cost-suboptimal minimizer of the original objective function F.

Theorem 2. *Given some $\lambda \in \left(0, \frac{\varepsilon}{m\|x^{(1)} - x^\star\|^2}\right)$ and any initialization $x^{(1)} \in \mathbf{R}^d$, suppose that we run the* `FedSplit` *procedure (Algorithm 1) on the regularized objective F_λ using exact prox steps with stepsize $s = 1/\sqrt{\lambda L_\lambda^*}$. Then the* `FedSplit` *algorithm outputs a vector $\widehat{x} \in \mathbf{R}^d$ satisfying $F(\widehat{x}) - F^\star \leqslant \varepsilon$ after exceeding $\widetilde{O}\left(\sqrt{\frac{L^*\|x^{(1)} - x^\star\|^2}{\varepsilon}}\right)$ iterations.*[3]

We remark that this faster convergence rate of $\widetilde{O}\left(t^{-2}\right)$ is nearly optimal for first-order algorithms [18], and to our knowledge such results were not known for operator splitting-based procedures prior to this work.

4 Experiments

In this section, we present numerical results for `FedSpliton` some convex federated optimization problem instances. We include additional details on these simulations in Section B of the supplement.

Logistic regression We begin with federated binary classification, where we solve,

$$\text{minimize} \quad \sum_{j=1}^m \sum_{i=1}^{n_j} \log(1 + \mathrm{e}^{-b_{ij} a_{ij}^\mathsf{T} x}), \tag{20}$$

with variable $x \in \mathbf{R}^d$. We generate the problem data $\{(a_{ij}, b_{ij})\} \subset \mathbf{R}^d \times \{\pm 1\}$ synthetically; see Section B.2.1 in the supplement for details.

We also use `FedSplit` to solve a multiclass classification problem, with K classes. Here we solve

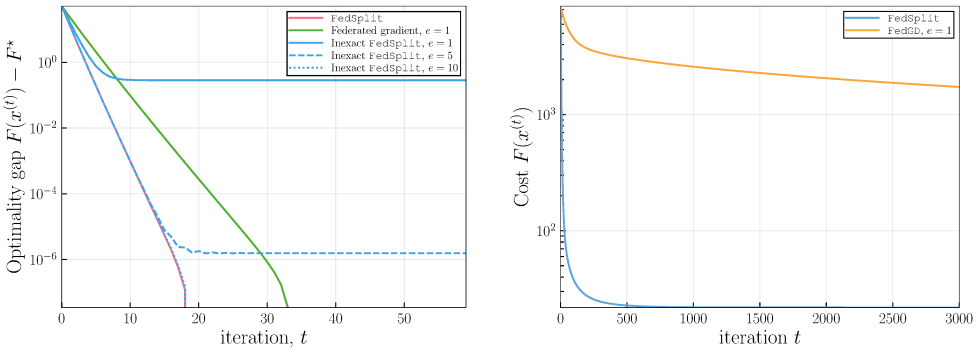
$$\text{minimize} \quad \sum_{j=1}^m \Big\{ \sum_{i=1}^{n_j} \sum_{k=1}^K \log(1 + \mathrm{e}^{-b_{ijk} a_{ij}^\mathsf{T} x_k}) + \frac{\lambda}{2} \sum_{k=1}^K \|x_k\|^2 \Big\} \tag{21}$$

with variables $x_1, x_2, \ldots, x_K \in \mathbf{R}^d$, regularization parameter $\lambda > 0$, and sample size $N = \sum_{j=1}^m n_j$. Here, the problem data $\{(a_{ij}, b_{ij})\} \subset \mathbf{R}^d \times \{\pm 1\}^K$ are images and multiclass labels from the FEMNIST dataset in the LEAF framework [6]. This dataset was proposed as a benchmark for federated optimization; there are $N = 805,263$ images, $m = 3,550$ clients, and $K = 62$ classes. The problem dimension is $d = 6,875$; see Section B.2.2 in the supplement for additional details.

In Figure 2, we present numerical results on problems (20) and (21). We implement `FedSplit` with exact proximal operators and inexact implementations with a constant number of gradient steps $e \in \{1, 5, 10\}$. For comparison, we implemented a federated gradient method as previously described (4). As shown in Figure 2(a), both `FedGD` with $e = 1$ and the `FedSplit` procedure exhibit linear convergence rates. Using inexact proximal updates with the `FedSplit` procedure preserves the linear convergence up to the error floor introduced by the exactness of the updates.In this case, the

[2]To avoid degeneracies, we assume $x \mapsto \sum_{j=1}^m f_j(x)$ is bounded below and attains its minimum.

[3]The $\widetilde{O}(\cdot)$ notation denotes constant and polylogarithmic factors that are not dominant.



(a) (b)

Figure 2. Cost versus iteration for `FedGD` and `FedSplit`. (a) Plot of the optimality gap $F(x^{(t)}) - F^\star$ versus the iteration number t for the logistic regression problem (20). This measures the difference between the optimal cost value and cost of an iterate returned at round t by a given algorithm. (b) Plot of cost $F(x^{(t)})$ versus iteration t for the FEMNIST multiclass logistic problem (21).

inexact proximal updates with $e = 10$—that is, performing 10 local updates per each round of global communication—suffice to track the exact `FedSplit` procedure up to an accuracy below 10^{-6}. In Figure 2(b), we see that `FedSplit` similarly outperforms `FedGD` on actual client data.[4].

Dependence on problem conditioning It is well-known that the convergence rates of first-order methods are affected by problem conditioning. First, let us re-state our theoretical guarantees in terms of *iteration complexity*. We let $T(\varepsilon, \kappa)$ denote the maximum number of iterations required so that, for any problem with condition number at most κ, the iterate $x^{(T)}$ with $T = T(\varepsilon, \kappa)$ satisfies the bound $F(x^{(T)}) - F^\star \leqslant \varepsilon$. For federated objectives with condition number κ as defined in (14), `FedSplit` and `FedGD` have iteration complexities

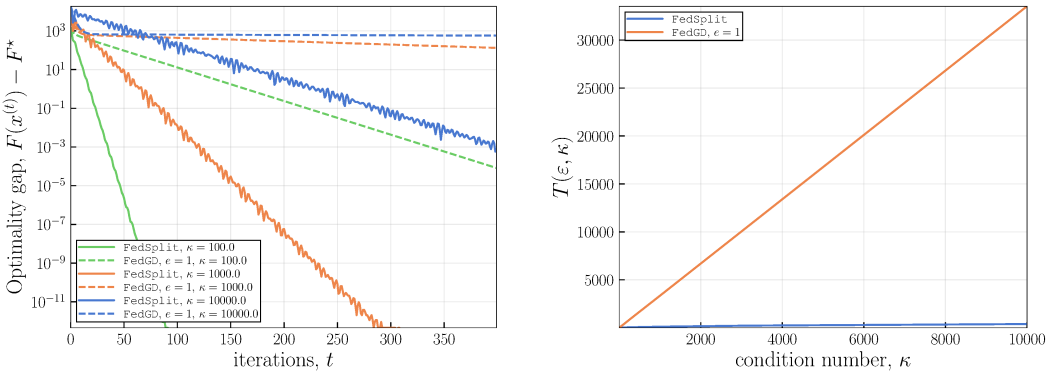
$$T_{\text{FedSplit}}(\varepsilon, \kappa) = O(\sqrt{\kappa} \log(1/\varepsilon)) \quad \text{and} \quad T_{\text{FedGrad}}(\varepsilon, \kappa) = O(\kappa \log(1/\varepsilon)). \tag{22}$$

This follows from Theorem 1 and standard results from convex optimization theory [18]. Hence, whereas `FedSplit` has a more expensive local update, it has much better dependence on the condition number κ. In the context of federated optimization, this iteration complexity should be interpreted as the number of communication rounds between clients and the coordinating entity. Hence, this highlights a concrete tradeoff between local computation and global communication in these methods. Note that while acclereated first-order methods matches the iteration complexity of `FedSplit`, they are sensitive to stepsize misspecification and are not robust to errors incurred in gradient updates [9]. This is in contrast to the inexact convergence guarantees that `FedSplit` enjoys (see Theorem 1).

In Figure 3, we present the results of a simulation study that shows these iteration complexity estimates are accurate in practice. We construct a sequence of least squares problems with varying condition number between 10 and 10000. We then look at the number of iterations required to obtain an ε-cost suboptimal solution with $\varepsilon = 10^{-3}$; see Section B.3 in the supplement for additional simulation details. In this way, we obtain estimates of the functions $\kappa \mapsto T_{\text{FedGrad}}(10^{-3}, \kappa)$ and $\kappa \mapsto T_{\text{FedSplit}}(10^{-3}, \kappa)$, which measure the dependence of the iteration complexity on the condition number. Figure 3 provides plots of these estimated functions.

Consistent with our theory, we see that `FedGD` has an approximately linear dependence on the condition number, whereas the `FedSplit` procedure has much milder dependence on conditioning. Concretely, for an instance with condition number $\kappa = 10000$, the `FedGD` procedure requires on the order of 34000 iterations, whereas the `FedSplit` procedure requires roughly 400 iterations. Therefore, while `FedSplit` involves more expensive intermediate proximal updates, it enjoys a smaller iteration count, which in the context of this federated setting indicates a significantly smaller number of communication rounds between clients and the the centralized server.

[4]Given the large scale nature of this example, we implement an accelerated gradient method for the proximal updates, terminated when the gradient of the proximal objective drops below 10^{-8}.



(a) (b)

Figure 3. Dependence of algorithms on the conditioning. (a) Plot of log cost suboptimality of iterate $x^{(t)}$ versus iteration t for condition number $\kappa \in \{100, 1000, 10000\}$. (b) Plots of the iteration complexity $T(\varepsilon; \kappa)$ versus κ at tolerance level $\varepsilon = 10^{-3}$ for the `FedGD` and `FedSplit` procedures.

5 Discussion

We highlight a few interesting directions for future work on federated learning and `FedSplit` . First, in practice, it is standard to use stochastic optimization algorithms in solving large-scale machine learning problems, and we are currently analyzing stochastic approximation procedures as applied to the device-based proximal updates underlying our method. Our results on the incorrectness of previously proposed methods and the work of Woodworth and colleagues [27] on the suboptimality on multi-step stochastic gradient methods, highlight the need for better understanding of the tradeoff between the accuracy of stochastic and deterministic approximations to intermediate quantities and rates of convergence in federated optimization. We also mention the possibility of employing stochastic approximation with higher-order methods, such as the Newton sketch algorithm [21, 22]. It is also important to consider our procedure under asynchronous updates, perhaps under delays in computation. Finally, an important desideratum in federated learning is suitable privacy guarantees for client the local data [3]. Understanding how noise aggregated through differentially private mechanisms couples with our inexact convergence guarantees is a key direction for future work.

Broader Impact

As mentioned in the introduction, a main application of federated optimization is to large-scale statistical learning, as carried out by application developers and cell phone manufacturers. On the other hand, learning from federated data is also inherent to other settings where data is not stored centrally: consider, for example, collecting clinical trial data across multiple hospitals and running a centralized analysis. Therefore, we envision analysts who are operating in these settings—where data is not available centrally due to communication barriers or privacy constraints—as main benefactors of this work. Our methods enjoy the same trade-offs with respect to biases in data, failures of systems, as other standard first-order algorithms. We believe that having convergent algorithms in this federated setting should help promote good practices with regard to analyzing large-scale, federated, and sensitive datasets.

Acknowledgments and Disclosure of Funding

We thank Bora Nikolic and Cong Ma for their careful reading and comments of an initial draft of this manuscript. RP was partially supported by a Berkeley Fellowship via the ARCS Foundation. MJW was partially supported by Office of Naval Research grant DOD-ONR-N00014-18-1-2640, and NSF grant NSF-DMS-1612948.

References

[1] H. H. Bauschke and P. L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2nd edition, 2017.

[2] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, Boston, MA, 1997.

[3] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers. Protection against reconstruction and its applications in private federated learning. Technical report, 2018. arxiv.org:1812.00984.

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

[5] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, Cambridge, UK, 2004.

[6] S. Caldas, P. Wu, et al. LEAF: A benchmark for federated settings. abs/1812.01097, 2018.

[7] P. L. Combettes. Monotone operator theory in convex optimization. *Math. Program.*, 170(1, Ser. B):177–206, 2018.

[8] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. *J. Mach. Learn. Res.*, 13(1):165—-202, January 2012.

[9] O. Devolder, F. Glineur, and Y. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1–2):37–75, 2014.

[10] S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *J. Mach. Learn. Res.*, 17(83):1–5, 2016.

[11] P. Giselsson and S. Boyd. Linear convergence and metric selection for Douglas-Rachford splitting and ADMM. *IEEE Trans. Automatic Control*, 62(2):532–544, February 2017.

[12] K. Goebel and W. A. Kirk. *Topics in metric fixed point theory*, volume 28 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 1990.

[13] P. Kairouz, H. B. McMahan, et al. Advances and open problems in federated learning. Technical report, December 2019. arXiv.org:1912.04977.

[14] M. Li, D. G. Andersen, A. J. Smola, and K. Yu. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems 27*, pages 19—27. Curran Associates, Inc., 2014.

[15] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: challenges, methods and future directions. Technical Report arxiv.org/abs/1908.07873, August 2019.

[16] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. Technical Report arxiv.org/abs/1812.06127, December 2018.

[17] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas. Communication-efficient learning of deep networks from decentralized data. Technical Report arxiv.org/abs/1602.05629, February 2016.

[18] Y. Nesterov. *Introductory Lectures on Convex Optimization*. Kluwer Academic Publisher, New York, 2004.

[19] N. Parikh, S. Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[20] D. W. Peaceman and Jr. H. H. Rachford. The numerical solution of parabolic and elliptic differential equations. *Journal of the SIAM*, 3(1):28–41, March 1955.

[21] M. Pilanci and M. J. Wainwright. Iterative Hessian Sketch: Fast and accurate solution approximation for constrained least-squares. *Journal of Machine Learning Research*, 17(53):1–38, April 2016.

[22] M. Pilanci and M. J. Wainwright. Newton sketch: A linear-time optimization algorithm with linear-quadratic convergence. *SIAM Jour. Opt.*, 27(1):205–245, March 2017.

[23] P. Richtárik and M. Takáč. Distributed coordinate descent method for learning with big data. *J. Mach. Learn. Res.*, 17(1):2657—2681, January 2016.

[24] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM J. Control Optim.*, 14(5):877–898, 1976.

[25] R.T. Rockafellar and R. J-B Wets. *Variational Analysis*, volume 317. Springer Science & Business Media, 2009.

[26] E. K. Ryu and S.P. Boyd. Primer on monotone operator methods. *Applied Computational Mathematics: an Interational Journal*, 15(1):3–43, 2016.

[27] B. E. Woodworth, K. K. Patel, S. U. Stich, Z. Dai, B. Bullins, H. B. McMahan, O. Shamir, and N. Srebro. Is local SGD better than minibatch SGD? Technical report, 2020. arxiv.org:2002.07839.

[28] Y. Zhang, J. C. Duchi, and M. J. Wainwright. Communication-efficient algorithms for statistical optimization. *J. Mach. Learn. Res.*, 14(68):3321—3363, 2013.

Supplement to "FedSplit: an algorithmic framework for fast federated optimization"

A Proofs

We now turn to the proofs of our main results. Prior to diving into these arguments, we first introduce two operators that play a critical role in our analysis. Given a convex function $\varphi: \mathbf{R}^d \to \mathbf{R}$, we define

$$\mathbf{prox}_{\varphi}(z) := \arg\min_{x \in \mathbf{R}^d} \left\{ \varphi(x) + \frac{1}{2}\|z - x\|^2 \right\} \quad \text{and} \tag{23a}$$

$$\mathbf{refl}_{\varphi}(z) := 2\,\mathbf{prox}_{\varphi}(z) - z. \tag{23b}$$

These are called the proximal and reflected resolvent operators associated with the function φ. The first operator is also known as the resolvent; the second operator above is also known as the Cayley operator of φ. Moreover, our analysis makes use of the (semi)norm on Lipschitz continuous functions $f: \mathbf{R}^d \to \mathbf{R}$ given by

$$\mathrm{Lip}(f) := \sup_{x \neq y} \frac{|f(x) - f(y)|}{\|x - y\|}. \tag{24}$$

For short, we say that that f is $\mathrm{Lip}(f)$-Lipschitz continuous when it satisfies this condition.

A.1 Proofs of guarantees for FedSplit

We begin by proving our guarantees for the FedSplit procedure, including the correctness of its fixed points (Proposition 3); the general convergence guarantee in the strongly convex case (Theorem 1); the general convergence guarantee in the weakly convex case (Theorem 2), and Corollary 1 on its convergence with approximate proximal updates.

A.2 Proof of Proposition 3

By the fixed point assumption, the block average $x^\star := \overline{z^\star}$ satisfies the relation

$$\mathbf{prox}_{s f_j}(2x^\star - z_j^\star) = x^\star \qquad \text{for } j = 1, 2, \ldots, m.$$

Since each f_j is convex and differentiable, by the first-order stationary conditions implied by the definition of the prox operator (23a), we must have

$$\nabla f_j(x^\star) + \tfrac{1}{s}\big\{x^\star - \big(2x^\star - z_j^\star\big)\big\} \;=\; \nabla f_j(x^\star) + \tfrac{1}{s}\big\{z_j^\star - x^\star\big\} \;=\; 0 \quad \text{for } j = 1, \ldots, m.$$

Summing these equality relations over $j = 1, \ldots, m$ and using the fact that $x^\star = \frac{1}{m}\sum_{j=1}^m z_j^\star$ yields the zero gradient condition

$$\sum_{j=1}^m \nabla f_j(x^\star) \;=\; 0.$$

Since the function $x \mapsto \sum_{j=1}^m f_j(x)$ is convex, this zero-gradient condition implies that $x^\star \in \mathbf{R}^d$ is a minimizer of the distributed problem as claimed.

A.2.1 Proof of Theorem 1

We now turn to the proof of Theorem 1. Our strategy is to prove it as a consequence of a somewhat more general result, which we begin by stating here. In order to lighten notation, we use the fact that the proximal operator for the function $F(z_1, \ldots, z_m) = \sum_{j=1}^m f_j(z_j)$ is block-separable, so that in terms of the block-partitioned vector $z = (z_1, \ldots, z_m)$, we can write

$$\mathbf{prox}_{sF}(z) = \big(\mathbf{prox}_{s f_1}(z_1), \ldots, \mathbf{prox}_{s f_m}(z_m)\big), \quad \text{for all } z = (z_1, \ldots, z_m) \in (\mathbf{R}^d)^m.$$

We also recall the the approximate proximal operator used in the FedSplit procedure, namely

$$\widetilde{\mathbf{prox}}(z) := (\texttt{prox\_update}_1(z_1), \ldots, \texttt{prox\_update}_m(z_m)), \quad \text{for all } z_1, \ldots, z_m \in \mathbf{R}^d.$$

Theorem 3 (Convergence with general residuals)**.** *Suppose that the functions $f_j : \mathbf{R}^d \to \mathbf{R}$ are ℓ_j-strongly convex and L_j-smooth for $j = 1, \ldots, m$, and for $t = 1, 2, \ldots$, define the residuals*

$$r^{(t)} := \widetilde{\mathbf{prox}}(2\overline{z^{(t)}} - z^{(t)}) - \mathbf{prox}_{sF}(2\overline{z^{(t)}} - z^{(t)}). \tag{25}$$

Then with stepsize $s = 1/\sqrt{\ell_* L^*}$, the `FedSplit` *procedure (Algorithm 1) has a unique fixed point $z^\star$, and the iterates satisfy*

$$\|z^{(t+1)} - z^\star\| \leqslant \rho^t \|z^{(1)} - z^\star\| + 2\sum_{j=1}^{t} \rho^{t-j} \|r^{(j)}\| \qquad \text{for } t = 1, 2, \ldots, \tag{26}$$

where $\rho := 1 - 2/(\sqrt{\kappa} + 1)$ is the contraction coefficient.

Let us use Theorem 3 to derive the claim stated in Theorem 1. Note that by Proposition 3, the fixed points of Algorithm 1 are minimizers of F, hence unique under the strong convexity assumption. Consequently, we have

$$\|x^{(t+1)} - x^\star\| \leqslant \frac{1}{\sqrt{m}} \|z^{(t+1)} - z^\star\|, \qquad \text{for all } t = 1, 2, \ldots.$$

Using Theorem 3 and the error bound, we then conclude that

$$\|x^{(t+1)} - x^\star\| \leqslant \frac{1}{\sqrt{m}} \left(1 - \frac{2}{\sqrt{\kappa} + 1}\right)^t \|z^{(1)} - z^\star\| + (\sqrt{\kappa} + 1)b,$$

as claimed.

A.2.2 Proof of Theorem 3

We now turn to the proof of the more general claim. Given additive decomposition $F(z) = \sum_{j=1}^m f_j(z_j)$, the reflected resolvent induced by F is block-separable, taking the form

$$\mathbf{refl}_{sF}(z) = \left(\mathbf{refl}_{sf_1}(z_1), \ldots, \mathbf{refl}_{sf_m}(z_m)\right), \quad \text{for all } z = (z_1, \ldots, z_m) \in (\mathbf{R}^d)^m.$$

Similarly, consider the approximate reflected resolvent defined by the algorithm, namely

$$\widetilde{\mathbf{refl}}(z) := 2\widetilde{\mathbf{prox}}(z) - z, \quad \text{for all } z = (z_1, \ldots, z_m) \in (\mathbf{R}^d)^m.$$

It also has the same block-separable form.

Using these two block-separable operators, we can now define two abstract operators, each acting on the product space $(\mathbf{R}^d)^m$, that allow us to analyze the algorithm. The first operator $\mathcal{T}$ underlies the idealized algorithm, in which the proximal updates are exact, and the second operator $\widehat{\mathcal{T}}$ underlies the practical algorithm, which is based on approximate proximal updates. The idealized algorithm is based on iterating the operator

$$\mathcal{T}(z) := \mathbf{refl}_{sF}\left(\mathbf{refl}_{I_E}(z)\right). \tag{27}$$

In this definition, we use I_E to denote the indicator function for membership in the equality subspace E, so that $\mathbf{refl}_{I_E}$ is the reflected proximal operator for this function.

On the other hand, the practical algorithm generates the sequence $\{z^{(t)}\}_{t=1}^\infty$ via the updates $z^{(t+1)} = \widehat{\mathcal{T}}(z^{(t)})$, where $\widehat{\mathcal{T}} : (\mathbf{R}^d)^m \to (\mathbf{R}^d)^m$ is the *perturbed operator*

$$\widehat{\mathcal{T}}(z) = \widetilde{\mathbf{refl}}\big(\mathbf{refl}_{I_E}(z)\big). \tag{28}$$

Note that the idealized operator $\mathcal{T}$ and perturbed operator $\widehat{\mathcal{T}}$ satisfy the relation

$$\widehat{\mathcal{T}} - \mathcal{T} = \left(\widetilde{\mathbf{refl}} \circ \mathbf{refl}_{I_E} - \mathbf{refl}_{sF} \circ \mathbf{refl}_{I_E}\right). \tag{29}$$

Our proof involves verifying that with the stepsize choice $s = 1/\sqrt{\ell_* L^*}$, the mapping $\mathcal{T}$ is a contraction, with Lipschitz coefficient

$$\mathrm{Lip}(\mathcal{T}) \leqslant \underbrace{1 - \frac{2}{\sqrt{\kappa} + 1}}_{=:\,\rho} < 1. \tag{30}$$

Taking this claim as given for the moment, the contractivity implies that $\mathcal{T}$ has has a unique fixed point [12]—call it $z^\star \in (\mathbf{R}^d)^m$. Comparing with Proposition 3, we see that the definition of fixed points given there agrees with the fixed point $z^\star$ of the operator $\mathcal{T}$, since we have the relation $\mathbf{refl}_{I_E}(z) = 2\overline{z} - z$.

Using this contractivity condition, the distance between this fixed point $z^\star$ and the iterates $z^{(t)}$ of the `FedSplit` procedure can be bounded as

$$\begin{aligned}
\|z^{(t+1)} - z^\star\| &= \|\widehat{\mathcal{T}} z^{(t)} - \mathcal{T} z^\star\| \\
&\overset{\text{(i)}}{\leqslant} \|\mathcal{T} z^{(t)} - \mathcal{T} z^\star\| + 2\|\widetilde{\mathbf{prox}}\,\mathbf{refl}_{I_E}\, z^{(t)} - \mathbf{prox}_{sF}\,\mathbf{refl}_{I_E}\, z^{(t)}\| \\
&\overset{\text{(ii)}}{\leqslant} \mathrm{Lip}(\mathcal{T})\|z^{(t)} - z^\star\| + 2\|r^{(t)}\| \\
&\overset{\text{(iii)}}{\leqslant} \rho\|z^{(t)} - z^\star\| + 2\|r^{(t)}\|, \qquad (31)
\end{aligned}$$

where inequality (i) applies the triangle inequality to the relation (29) between the perturbed and idealized operators; step (ii) follows by definition of the residual $r^{(t)}$ at round t; and step (iii) follows from the bound (30) on the Lipschitz coefficient of $\mathcal{T}$. Performing induction on this bound yields the stated claim.

Proof of the bound (30)**:** It remains to bound the Lipschitz coefficient of the idealized operator $\mathcal{T}$. Since the composite function $F(z) := \sum_{j=1}^m f_j(z_j)$ is ℓ_*-strongly convex and L^*-smooth, known results on reflected proximal operators [11, Theorems 1 and 2] imply that with the stepsize choice $s = 1/\sqrt{\ell_* L^*}$, the operator $\mathbf{refl}_{sF}$ satisfies the bound

$$\|\,\mathbf{refl}_{sF}(z) - \mathbf{refl}_{sF}(z')\|_2 \leqslant \Big(1 - \frac{2}{\sqrt{\kappa}+1}\Big)\|z - z'\|_2 \qquad \text{for all } z, z' \in (\mathbf{R}^d)^m. \qquad (32)$$

On the other hand, the reflected proximal operator $\mathbf{refl}_{I_E}$ for the indicator function $\mathbf{refl}_{I_E}$ is non-expansive, so that

$$\|\,\mathbf{refl}_{I_E}(z) - \mathbf{refl}_{I_E}(z)\|_2 \leqslant \|z - z'\|_2 \qquad \text{for all } z, z' \in (\mathbf{R}^d)^m. \qquad (33)$$

Applying the triangle inequality and using the definition (27) of the idealized operator $\mathcal{T}$, we find that

$$\begin{aligned}
\|\mathcal{T}(z) - \mathcal{T}(z')\|_2 &\leqslant \|\,\mathbf{refl}_{sF}\left(\mathbf{refl}_{I_E}(z)\right) - \mathbf{refl}_{sF}\left(\mathbf{refl}_{I_E}(z')\right)\|_2 \\
&\overset{\text{(iv)}}{\leqslant} \Big(1 - \frac{2}{\sqrt{\kappa}+1}\Big)\,\|\,\mathbf{refl}_{I_E}(z) - \mathbf{refl}_{I_E}(z')\|_2 \\
&\overset{\text{(v)}}{\leqslant} \Big(1 - \frac{2}{\sqrt{\kappa}+1}\Big)\,\|z - z'\|_2,
\end{aligned}$$

where step (iv) uses the contractivity (32) of the operator $\mathbf{refl}_{sF}$, and step (v) uses the non-expansiveness (33) of the operator $\mathbf{refl}_{I_E}$. This completes the proof of the bound (30).

A.2.3 Proof of Corollary 1

By construction, the function h_j is smooth with parameter $M := sL^* + 1$ and strongly convex with parameter $m := s\ell_* + 1$. Consequently, if we define the operator $H_j(u) := u - \alpha\nabla h_j(u)$, then by standard results on gradient methods for smooth-convex functions, the stepsize choice $\alpha = \frac{2}{M+m}$ ensures that the operator H_j is contractive with parameter at least $\rho = 1 - \frac{m}{M}$. Thus, we have the bound

$$\|u^{(e+1)} - u^*\|_2 \leqslant \rho^e\|u^{(1)} - u^*\|_2,$$

where $u^* = \mathbf{prox}_{sf_j}(x_j^{(t)})$ is the optimum of the proximal subproblem. Unpacking the definitions of (m, M) and recalling that $s = 1/\sqrt{\ell_* L^*}$, we have

$$\frac{M}{m} = \frac{sL^* + 1}{s\ell_* + 1} \;=\; \frac{\sqrt{\frac{L^*}{\ell_*}} + 1}{\sqrt{\frac{\ell_*}{L^*}} + 1} \leqslant\; \sqrt{\kappa} + 1,$$

and hence $\rho \leqslant 1 - \frac{1}{\sqrt{\kappa}+1}$, which establishes the claim.

A.2.4 Proof of Theorem 2

Recalling the definition (19) of the regularized objective F_λ, note that it is related to the unregularized objective F via the relation $F_\lambda(x) = F(x) + \frac{m\lambda}{2}\|x - x^{(1)}\|^2$, where $x^{(1)}$ is the given initialization. The proposed procedure is to compute an approximation to the quantity

$$x_\lambda^\star := \arg\min_{x \in \mathbf{R}^d} \Bigg(\underbrace{\sum_{j=1}^m \Big\{ f_j(x) + \frac{\lambda}{2}\|x - x^{(1)}\|^2 \Big\}}_{=: F_\lambda(x)} \Bigg).$$

Now suppose that we have computed a vector $\widehat{x} \in \mathbf{R}^d$ satisfies $F_\lambda(\widehat{x}) - F_\lambda(x_\lambda^\star) \leqslant \varepsilon/2$. Letting $F^\star = F(x^\star)$ denote the optimal value of the original (unregularized) optimization problem, we have

$$F(\widehat{x}) - F^\star = \Big\{ F(\widehat{x}) - F_\lambda(x_\lambda^\star) \Big\} + \Big\{ F_\lambda(x_\lambda^\star) - F(x^\star) \Big\}. \tag{34}$$

By definition of F_λ, we have $F(\widehat{x}) \leqslant F_\lambda(\widehat{x})$. Moreover, again using the definition of F_λ, we have

$$\begin{aligned} F_\lambda(x_\lambda^\star) - F(x^\star) &= F_\lambda(x_\lambda^\star) - F_\lambda(x^\star) + \frac{m\lambda}{2}\|x^\star - x^{(1)}\|^2 \\ &\leqslant \frac{m\lambda}{2}\|x^\star - x^{(1)}\|^2, \end{aligned}$$

where the inequality follows since $x_\lambda^\star$ minimizes F_λ by definition. Substituting these bounds into the initial decomposition (34), we find that

$$\begin{aligned} F(\widehat{x}) - F^\star &\leqslant \Big\{ F_\lambda(\widehat{x}) - F_\lambda(x_\lambda^\star) \Big\} + \frac{m\lambda}{2}\|x^\star - x^{(1)}\|^2 \\ &\leqslant \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon. \end{aligned} \tag{35}$$

where the inequality follows since since $\widehat{x}$ is $(\varepsilon/2)$-cost-suboptimal for F_λ, and by our selection of λ. Thus to finish the proof, we simply need to check how many iterations it takes to compute an $(\varepsilon/2)$-cost-suboptimal point for F_λ.

Let us define the shorthand notation $\overline{L} := \sum_{j=1}^m L_j$ and $\kappa_\lambda := \frac{L^* + \lambda}{\lambda}$. Since F_λ is a sum of functions that are λ-strongly convex and $(L_j + \lambda)$-smooth, it follows that from initialization $x^{(1)}$, the `FedSplit` algorithm outputs iterates $x^{(t)}$ satisfying the bound

$$\begin{aligned} F_\lambda(x^{(t+1)}) - F_\lambda(x_\lambda^\star) &\overset{\text{(i)}}{\leqslant} \frac{\overline{L} + m\lambda}{2}\|x^{(t+1)} - x_\lambda^\star\|^2 \\ &\overset{\text{(ii)}}{\leqslant} \frac{\overline{L} + m\lambda}{2}\left(1 - \frac{2}{\sqrt{\kappa_\lambda} + 1}\right)^{2t} \frac{\|x^{(1)} - z_\lambda^\star\|^2}{m}. \end{aligned} \tag{36}$$

In the above reasoning, inequality (i) is a consequence of the smoothness of the losses f_j when regularized by λ, along with the first-order optimality condition for $x_\lambda^\star$; and bound (ii) then follows by squaring the guarantee of Theorem 1 with $b = 0$. By inverting the bound (36), we see that in order to achieve an $\varepsilon/2$-optimal solution, it suffices to take the number of iterations t to be lower bounded as

$$t \geqslant \left\lceil \frac{\sqrt{\kappa_\lambda} + 1}{4} \log\left\{ \frac{(\overline{L} + \lambda m)\|x^{(1)} - z_\lambda^\star\|^2}{m} \right\} \right\rceil.$$

Evaluating this bound with the choice $\kappa_\lambda = 1 + L^*/\lambda$ and recalling the bound (35) yields the claim of the theorem.

A.3 Characterization of fixed points

In this section we give the two fixed point results for `FedSGD` and `FedProx` as stated in Section 3.1.

A.3.1 Proof of Proposition 1

We begin by characterizing the fixed points of the FedSGD algorithm. By definition, any limit point $(x_1^\star, \ldots, x_m^\star) \in (\mathbf{R}^d)^m$ must satisfy the fixed point relation

$$x_j^\star = \frac{1}{m} \sum_{j=1}^m G_j^e(x_j^\star), \qquad j = 1, 2, \ldots, m.$$

Thus, the limits $x_j^\star$ are common, and this gives part (a) of the claim. Expanding the iterated operator G_j^e gives part (b).

A.3.2 Proof of Proposition 2

We now characterize the fixed points of the FedProx algorithm. By definition, any limit point $(x_1^\star, \ldots, x_m^\star)$ satisfies

$$x_j^\star = \frac{1}{m} \sum_{j=1}^m \mathbf{prox}_{sf_j}(x_j^\star), \qquad j = 1, 2, \ldots, m. \tag{37}$$

Thus, the limits $x_j^\star$ are common, and this gives part (a) of the claim.

For any convex function, $f \colon \mathbf{R}^d \to \mathbf{R}$, the proximal operator satisfies

$$\mathbf{prox}_{sf}(v) = v - s\nabla M_{sf}(v), \quad \text{for all } s > 0 \text{ and } v \in \mathbf{R}^d.$$

Using this identity in display (37) yields part (b) of the claim.

B Details for simulation studies

All of the experiments were conducted on a 2.6 GHz Intel Core i7 processor, in Python 3.7.3. Our logistic regression experiments used CVXPY, convex programming [10] software that we used to implement the exact proximal operators.

B.1 Results presented in Figure 1

For the simulation, we construct a least squares problem where for $j \in [m]$, the response vector $b_j \in \mathbf{R}^{n_j}$ obeys the linear model $b_j = A_j x_0 + v_j$, where $x_0 \in \mathbf{R}^d$ is the unknown parameter vector to be estimated, and the noise vectors v_j are independently distributed as $v_j \overset{\text{ind.}}{\sim} \mathsf{N}\left(0, \sigma^2 I_{n_j}\right)$ for some $\sigma > 0$. For our experiments reported here, we constructed a random instance of such a problem with $m = 25, \quad d = 100$, $n_j \equiv 500$ and $\sigma^2 = 0.25$. We generated the design matrices with i.i.d.entries of the form $(A_j)_{kl} \overset{\text{i.i.d.}}{\sim} \mathsf{N}(0, 1)$, for $k = 1, \ldots, n_j$ and $l = 1, \ldots, d$. The aspect ratios of A_j satisfy $n_j > d$ for all j, thus by construction the matrices A_j are full rank with probability 1.

B.2 Results presented in Figure 2

B.2.1 Synthetic dataset

Here, we have design matrices $A_j \in \mathbf{R}^{n_j \times d}$ and label vectors $b_j \in \{1, -1\}^{n_j}$. We denote the rows of A_j by $a_{ij} \in \mathbf{R}^d$ for $i = 1, \ldots, n_j$. The conditional probability of positive class label $b_{ij} = 1$ under unknown parameter vector x_0 is then

$$\mathbf{P}\{b_{ij} = 1\} = \frac{\mathrm{e}^{a_{ij}^T x_0}}{1 + \mathrm{e}^{a_{ij}^T x_0}}, \quad \text{for } i \ = 1, \ldots, n_j. \tag{38}$$

Given observations of this form, we solve the *logistic regression* problem, This problem is smooth and convex,and clearly a special case of the more general class of federated problems (1).

We construct random instances of logistic regression problems with the settings $d = 100, n_j \equiv 1000$ and $m = 10$. Hence, we have a total sample size of $n = 10000$. We draw $a_{ij} \overset{\text{i.i.d.}}{\sim} \mathsf{N}(0, I_d)$ for all i, j and $x_0 \overset{\text{i.i.d.}}{\sim} \mathsf{N}(0, I_d)$. The binary labels then are constructed to follow the Bernoulli model (38).

B.2.2 FEMNIST datset

For this experiment only, we used Amazon EC2 to carry out these experiments (on `c5.metal` instances). The original dataset is comprised of 28×28 images, which we vectorize in row major order to obtain data points in $u_{ij} \in \mathbf{R}^{784}$. We further preprocessed these datapoints by adding a constant feature, and adding $(Ru)_+$ and $(Gu)_+$, where $R \in \{\pm 1\}^{3000 \times 784}$ and $G \in \mathbf{R}^{3000 \times 784}$ are filled with i.i.d. Rademacher and standard Normal entries. Here, $(\cdot)_+$ denotes the entrywise positive part of a vector. Therefore our final datapoints are

$$a_{ij} = (1, u_{ij}, (Ru_{ij})_+, (Gu_{ij})_+) \in \mathbf{R}^{6785}.$$

There were $K = 62$ classes in the dataset; we encode the labels as vectors $b_{ij} \in \{\pm 1\}^K$. Formally, if a_{ij} belongs to class $k \in [K]$, we set $b_{ij} = 2e_k - \mathbf{1}$, where e_k denotes the kth standard basis vector in $\mathbf{R}^K$.

We added the additional random features given above to improve the performance of our model on held out data. We set $\lambda = 0.01$ by cross-validation on a smaller subsample of the FEMNIST dataset. Formally, for each client, we select a random, 20% fraction of the data to reserve as a heldout set, not used for training our classifier. We train the one-versus-all multiclass classifier, according to the objective given in (21) by `FedSplit`until approximately satisfying the optimality condition of the distributed problem. We then compute the accuracy of our multiclass classifier on the held out data and repeated this for choices of $\lambda \in [10^{-3}, 10^3]$; $\lambda = 0.01$ worked best on the held out data, giving an accuracy of 73%. As mentioned in the paper, the proximal solves for `FedSplit`were carried out using accelerated gradient descent.

B.3 Results presented in Figure 3

We now describe the results of a simulation study that demonstrates the accuracy of these predicted iteration complexities. At a high level, our strategy is to construct a sequence of problems, indexed by an increasing sequence of condition numbers κ, and to estimate the number of iterations required to achieve a given tolerance $\varepsilon > 0$ as a function of κ. In order to do, it suffices to consider ensembles of least squares problems (8), but with a carefully constructed collection of design matrices, which we now describe.

For a given integer $\ell \geqslant 2$, let $\mathrm{O}(\ell)$ denote the set of $\ell \times \ell$ orthogonal matrices over the reals, and let $\mathsf{Unif}(\mathrm{O}(\ell))$ denote the uniform (Haar) measure on this compact group. With this notation, we begin by sampling i.i.d.random matrices

$$U_j^{(\kappa)} \sim \mathsf{Unif}(\mathrm{O}(n_j)) \quad \text{and} \quad V_j^{(\kappa)} \sim \mathsf{Unif}(\mathrm{O}(d)), \qquad \text{for } j = 1, \ldots, m. \tag{39}$$

For a given condition number $\kappa \geqslant 1$, we define a padded diagonal matrix—that is

$$\Lambda_j^{(\kappa)} = \left[\mathbf{diag}(\lambda_j^{(\kappa)}) \quad 0_{d,(n-d)}\right] \quad \text{where} \quad \lambda_j^{(\kappa)} = (\sqrt{\kappa}, 1, \ldots, 1) \in \mathbf{R}^d.$$

Above, the matrix $0_{d,(n_j-d)} \in \mathbf{R}^{d \times (n_j - d)}$ has all entries equal to zero. Given the random orthogonal matrices and the matrix $\Lambda_j^{(\kappa)} \in \mathbf{R}^{n_j \times d}$, we then construct the design matrices $A_j^{(\kappa)} \in \mathbf{R}^{n_j \times d}$ by setting

$$A_j^{(\kappa)} := U_j^{(\kappa)} \Lambda_j^{(\kappa)} V_j^{(\kappa)}, \quad \text{for all } \; j = 1, \ldots, m.$$

These choices ensure that the federated least squares objective (8) has condition number κ.

As before, the response vectors $b_j^{(\kappa)}$ obey a Gaussian linear measurement model,

$$b_j^{(\kappa)} = A_j^{(\kappa)} x_0 + v_j^{(\kappa)}, \quad \text{for } j = 1, \ldots, m, \quad \text{and for all } \kappa \in K.$$

We again take $v_j^{(\kappa)} \overset{\text{ind.}}{\sim} \mathsf{N}\left(0, \sigma^2 I_{n_j}\right)$. In our experiments, we draw the parameter $x_0 \sim \mathsf{N}\left(0, I_d\right)$, and use the parameter settings

$$m = 10, \quad d = 100, \quad n_j \equiv 400, \quad \text{and} \quad \sigma^2 = 1.$$

With these settings, we iterated over a collection of condition numbers $\kappa \in \{10^0, 10^{0.5}, \ldots, 10^{3.5}, 10^4\}$. For each choice of κ, after generating a random instance as described above, we measured the number of iterations required for `FedGD` and the `FedSplit` procedures, respectively, to reach a target accuracy $\varepsilon = 10^{-3}$, which is modest at best.