A Stochastic Newton Algorithm for Distributed Convex Optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

We analyze a stochastic Newton algorithm for distributed convex optimization. At the heart of our approach is the recent work showing that least-squares objectives can be optimized to high accuracy using a parallel algorithm with only a single round of communication. Our algorithm expresses the Newton update as the solution to a least-squares problem which we optimize using stochastic gradients and stochastic Hessian-vector products for the objective, both of which can typically be computed efficiently. We analyze our method for quasi-self-concordant objectives (e.g. logistic regression), and demonstrate that it can in some instances achieve faster convergence rates than comparable first-order methods while requiring less communication and a similar amount of computation. We also complement our theoretical analysis with promising empirical results that show our method is competitive with related first-order algorithms.

1 Introduction

Stochastic optimization methods that leverage parallelism have proven immensely useful in modern optimization problems. Recent advances in machine learning have highlighted their importance as these techniques now rely on millions of parameters and increasingly large training sets.

While there are many possible ways of parallelizing optimization algorithms, we consider the intermittent communication setting (Zinkevich et al., 2010; Cotter et al., 2011; Dekel et al., 2012; Shamir et al., 2014; Woodworth et al., 2018, 2021), where M parallel machines work together to optimize an objective during R rounds of communication, and there is an oracle that each machine may query K times during each round. An important example of this setting is when this oracle gives independent, unbiased stochastic estimates of the gradient, in which case this setting includes algorithms like Local SGD (Zinkevich et al., 2010; Coppola, 2015; Zhou and Cong, 2018; Stich, 2018; Woodworth et al., 2020), Minibatch SGD (Dekel et al., 2012), Minibatch AC-SA (Ghadimi and Lan, 2012), and many others.

It is natural to use stochastic gradients in this setting because they can typically be computed very efficiently, and myriad recent papers have studied first-order methods. However, as we will discuss, Woodworth et al. (2021) proved strong lower bounds for stochastic first-order methods in several natural settings, which establish significant limitations on what it is possible to guarantee theoretically. Their work therefore motivates efforts to circumvent these lower bounds by introducing additional assumptions about the objective or considering more powerful oracles. In this paper, we do both.

More specifically, we begin with the observation by Woodworth et al. (2020) that for *least-squares* objectives, certain first-order methods like Local SGD can optimize the objective to a very high degree of accuracy using just a single round of communication. This motivates trying to reduce the task of optimizing general convex objectives to a short sequence of least squares problems. Indeed,

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Table 1: Convergence rates for different algorithms in the intermittent communication setting. Parameters H, Q, σ, and ρ are as defined in Section 2. For the sake of clarity, we omit additional constants and, in the case of FEDSN, logarithmic factors.

Algorithm	Convergence Rate	Assump.	Reference
LOCALSGD	$\frac{HB^2}{KR} + \frac{\sigma B}{\sqrt{MKR}} + \frac{(H\sigma^2 B^4)^{1/3}}{(\sqrt{K}R)^{2/3}}$	A1	Woodworth et al. (2020)
FEDAC	$\frac{HB^2}{KR^2} + \frac{\sigma B}{\sqrt{MKR}} + \min\left\{\frac{H^{1/3}\sigma^{2/3}B^{4/3}}{K^{1/3}R}, \frac{H^{1/2}\sigma^{1/2}B^{3/2}}{K^{1/4}R}\right\}$	A1	Yuan and Ma (2020)
LOCALSGD	$\frac{HB^2}{KR} + \frac{\sigma B}{\sqrt{MKR}} + \frac{(Q\sigma^2 B^5)^{1/3}}{(\sqrt{K}R)^{2/3}}$	A3	Yuan and Ma (2020)
FEDAC	$\frac{HB^2}{KR^2} + \frac{\sigma B}{\sqrt{MKR}} + \frac{H^{1/3}\sigma^{2/3}B^{4/3}}{M^{1/3}K^{1/3}R} + \frac{Q^{1/3}\sigma^{2/3}B^{5/3}}{K^{1/3}R^{4/3}}$	A3	Yuan and Ma (2020)
FEDSN	exp. decay + $\frac{HB^2}{KR} + \frac{\sigma B}{\sqrt{MK}} + \frac{\rho B^2}{K^{1/2}R}$	A2	Theorem 4

this is precisely the idea behind many second-order algorithms including Newton's method (Nesterov and Nemirovskii, 1994), trust-region methods (Nocedal and Wright, 2006), and cubic regularization (Nesterov and Polyak, 2006), as well as methods that go beyond second-order information (Nesterov, 2019; Bullins, 2020).

Computing each Newton step requires solving a linear system of the form $\nabla^2 F(x)\Delta x = \nabla F(x)$. This can, of course, be done directly using the full Hessian of the objective. However, this is computationally prohibitive in high dimensions, particularly when compared to cheap stochastic gradients. To avoid this issue, we reformulate the Newton step as the solution to a least squares problem, $\min_{\Delta x} \frac{1}{2}\|\nabla^2 F(x)\Delta x - \nabla F(x)\|^2$, and use Local SGD to solve this using a single round of communication. Conveniently, computing stochastic gradient estimates for this least squares objective does not require computing the full Hessian matrix, and it only requires stochastic gradients and *stochastic Hessian-vector products*. These are much more attractive computationally because the cost of computing stochastic Hessian-vector products is often within a constant factor of that for stochastic gradients. The advantages offered by Hessian-vector products have been extensively studied, particularly in the settings of both deterministic (Agarwal et al., 2017; Carmon et al., 2018) and stochastic (Allen-Zhu, 2018; Arjevani et al., 2020) sequential non-convex optimization.

Our main algorithmic contribution is the method FEDERATED-STOCHASTIC-NEWTON (FEDSN), a distributed approximate Newton's method which leverages the benefits of Local SGD for least-squares problems (Woodworth et al., 2020). Under the stronger smoothness condition of quasi-self-concordance (Bach, 2010; Carmon et al., 2020) we show expected convergence rates which, for a relevant regime of parameters, improve upon the rates of previous methods such as Local SGD and FEDAC (Yuan and Ma, 2020). We also show that, under certain conditions, our method can guarantee lower error than the lower bound of Woodworth et al. (2021), meaning that our method can sometimes do better than any first-order algorithm could in the intermittent communication setting.

Stronger smoothness conditions and Newton's method

We will look into notions of smoothness which are standard for second-order methods, particularly that of α-quasi-self-concordance (QSC) (Bach, 2010; Carmon et al., 2020), which is satisfied when, for all $x, v, u \in \mathbb{R}^d$,

$$|\nabla^3 F(x)[v, u, u]| \leq \alpha \|v\| \left(\nabla^2 F(x)[u, u]\right). \tag{1}$$

Though quasi-self-concordance is perhaps a less well-studied condition than self-concordance (Nesterov and Nemirovskii, 1994), several recent works have brought its usefulness to light (Bach, 2010; Karimireddy et al., 2018; Carmon et al., 2020). Notably, for logistic regression, i.e., problems of the

form

$$\min_x F(x) = \frac{1}{N}\sum_{i=1}^{N} \log\Big(1 + e^{-b_i \langle a_i,\, x\rangle}\Big), \tag{2}$$

we observe that α-quasi-self-concordance holds with $\alpha \leq \max_i\{\|b_i a_i\|\}$.

A basic driver of progress common among these approaches for quasi-self-concordant optimization is the (damped) Newton's method for $\eta_t > 0$, i.e.,

$$x_{t+1} = x_t - \eta_t[\nabla^2 F(x_t)]^{-1}\nabla F(x_t). \tag{3}$$

Bach (2010) analyzes (3) under QSC conditions, in a manner analogous to that of standard self-concordance, to establish the behavior of Newton's method particularly in the region of quadratic $(\log\log(1/\epsilon))$ convergence. Carmon et al. (2020) use a Newton-type descent method to approximately implement a certain ℓ_2-ball minimization oracle that, when combined with an acceleration scheme, attains $F(x_T) - F(x^*) \leq \epsilon$ with only $\tilde{O}((D\alpha)^{2/3})$ calls to a least-squares solver, where D is a bound on the initial distance to the optimum.

A downside to these approaches is the potentially high cost of exactly computing each Newton step. Indeed, computing and inverting the full Hessian is generally superlinear in the problem dimension, which is prohibitive for machine learning problems that regularly have millions or even billions of parameters.

Our results

The main contribution of this work is a stochastic Newton-type algorithm for distributed quasi-self-concordant convex optimization. As we discuss further in Section 3, our approach first represents each Newton step as the solution to a least-squares problem, which we may then solve to sufficient accuracy using an efficient parallel algorithm for such problems (Woodworth et al., 2020). These steps are then used as part of an approximate trust-region-type problem, all the while dealing with technical challenges that arise from the stochastic nature of the problem. We then present the convergence rate of our main theoretical method FEDSN in Theorem 4, which can be shown to improve upon previous distributed methods for some parameter regimes in terms of M, K, and R, for different guarantees depending on the smoothness conditions. In Section 4, we provide a more practical version of our method, FEDSN-LITE (Algorithm 4), alongside empirical results that are competitive with related stochastic distributed methods, and even show improvements when subject to more limited communication.

Comparison with previous methods

A method of significant interest in the context of parallel optimization is the local stochastic gradient descent (SGD) algorithm (Mangasarian, 1995; Zinkevich et al., 2010). The idea of this approach is to execute K steps of stochastic gradient descent independently on M machines, after which the results of these parallel runs of SGD are communicated and averaged at the end of each of the R rounds to become the starting point for the following round. Analyzing the runtime of Local SGD has recently been the focus of several works (Stich, 2018; Stich and Karimireddy, 2019; Khaled et al., 2019; Woodworth et al., 2020). An accelerated variant has also been analyzed by Yuan and Ma (2020). Another approach to distributed optimization, particularly in the empirical risk minimization (ERM) setting, is to construct estimates of F by distributing the component functions of the finite-sum problem across machines. Leveraging this structure, in combination with distributed second-order methods, has since been shown to lead to improved communication efficiency (Shamir et al., 2014; Zhang and Xiao, 2015; Reddi et al., 2016; Wang et al., 2018; Crane and Roosta, 2019; Islamov et al., 2021; Gupta et al., 2021).

An overview comparing our method FEDSN (Algorithm 1) with Local SGD (Woodworth et al., 2020) and FEDAC (Yuan and Ma, 2020), under appropriate assumption sets, is presented in Table 1. We would note that, since Assumption 2 includes the quasi-self-concordance condition (in addition to providing stochastic Hessian-vector product access), rather than third-order-smoothness as in Assumption 3, it becomes difficult to directly compare against these previous methods. Nevertheless, to provide some sense in which we may compare FEDSN with FEDAC for functions which are both α-quasi-self-concordant and Q-third-order-smooth, we consider the case where $H = B = \sigma = \rho = O(1)$, M is taken sufficiently large, and where $R = \tilde{O}(B\alpha)$. In this setting, we will see that the

dominant terms for FEDAC and FEDSN are $Q^{1/3}K^{-1/3}\alpha^{-4/3}$ and $K^{-1/2}\alpha^{-1}$, respectively. Thus, FEDSN dominates FEDAC whenever $Q \gtrsim \alpha K^{-1/2}$.

2 Preliminaries

We consider the following optimization problem:

$$\min_{x \in \mathbb{R}^d} F(x) \tag{4}$$

for convex F, and we further establish several sets of assumptions under which we might analyze optimization algorithms. We provide these different settings in order to properly compare with previous works, as presented in, e.g., Table 1. We begin with the standard assumptions for smooth, stochastic, distributed optimization.

Assumption 1 (A1)**.**

(a) F is H-smooth, i.e., for all $x, y \in \mathbb{R}^d$, $F(y) \leq F(x) + \nabla F(x)^\top (y - x) + \frac{H}{2}\|y - x\|^2$.

(b) There is a minimizer $x^* \in \arg\min_x F(x)$ such that $\|x^*\| \leq B$.

(c) We are given access to functions $g(x; z)$ such that, for some z from an unknown distribution $\mathcal{D}$,

(i) $g(x; z)$ is an unbiased gradient estimate, i.e., $\mathbb{E}_z g(x; z) = \nabla F(x)$.

(ii) $g(x; z)$ has bounded variance, i.e., $\mathbb{E}_z \|g(x; z) - \nabla F(x)\|^2 \leq \sigma^2$.

Given the basic set above, we now introduce further assumptions both in terms of additional oracle access, as well as other smoothness notions. Crucially, the following details the requirement on the stochastic Hessian-vector products, though we once again stress that the practical cost of such an oracle is often on the order of that for stochastic gradients (Allen-Zhu, 2018).

Assumption 2 (A2)**.** *In addition to Assumption 1, we have:*

(a) F is α-quasi-self-concordant, i.e., $\left|\nabla^3 F(x)[v, u, u]\right| \leq \alpha \|v\| \nabla^2 F(x)[u, u]$ for all $x, v, u \in \mathbb{R}^d$.

(b) We are given access to functions $h(x, v; z)$ such that, for some z from an unknown distribution $\mathcal{D}$,

(i) $h(x, v; z)$ is an unbiased Hessian-vector product estimate, i.e., $\mathbb{E}_z h(x, v; z) = \nabla F^2(x) v$.

(ii) $h(x, v; z)$ has bounded variance of the form $\mathbb{E}_z \|h(x, v; z) - \nabla^2 F(x) v\|^2 \leq \rho^2 \|v\|^2$.

Finally, we consider the following assumptions as considered in Yuan and Ma (2020) which provide control over third-order smoothness and fourth central moment.

Assumption 3 (A3)**.** *In addition to Assumption 1, we have:*

(a) F is Q-third-order-smooth, i.e., for all $x, y \in \mathbb{R}^d$,

$$F(y) \leq F(x) + \nabla F(x)^\top (y - x) + \frac{1}{2} \left\langle \nabla^2 F(x)(y - x),\, y - x \right\rangle + \frac{Q}{6} \|y - x\|^3.$$

(b) $g(x; z)$ has bounded fourth central moment, i.e., $\mathbb{E}_z \|g(x; z) - \nabla F(x)\|^4 \leq \sigma^4$.

3 Main results

In this section we begin by describing our main algorithm, FEDSN (Algorithm 1). Namely, our aim is to solve convex problems $\min_x F(x)$, subject to Assumption 2. Recent efforts have shown how to more efficiently optimize quasi-self-concordant functions, including work by Karimireddy et al. (2018) and Carmon et al. (2020), who break down the overall problem into a sequence of constrained quadratic subproblems. Their work, at a high level, shows that as long as the radius of the constrained problem is not too large, it is possible to approximately solve it with only a logarithmic number of

Algorithm 1 FEDERATED-STOCHASTIC-NEWTON, a.k.a., FEDSN(F, r_0)

for $t = 0, 1, \ldots, T-1$ **do**
  $\Delta\tilde{x}_t = \text{CONSTRAINED-QUADRATIC-SOLVER}\big(d(5r_0) \cdot \nabla^2 F(x_t), \nabla F(x_t), r_0\big)$
  Update: $x_{t+1} = x_t + \Delta\tilde{x}_t$
Return: x_T

Newton-type updates. The trust-region-type problem in turn can be solved by approximately finding some "ideal" regularization parameter λ^* (Nocedal and Wright, 2006; Carmon et al., 2020).

Our method proceeds in a similar fashion: each iteration of Algorithm 1 provides an approximate solution to a constrained quadratic problem. However unlike previous works, a key difficulty comes from the fact that our Newton-type updates can only ensure small error in expectation. This issue of stochasticity is compounded by the requirements that come from the search for a "good enough" regularization parameter λ. It is details such as these in the subproblem solver CONSTRAINED-QUADRATIC-SOLVER (Algorithm 2) which contribute to additional technical considerations. Due to space constraints, we defer the proofs to the appendix.

To begin, we follow Karimireddy et al. (2018) in defining $d(r)$-local (Hessian) stability.

Definition 1. *We say that F is $d(r)$-locally stable if for any $x \neq y$ such that $\|x - y\| \leq r$ and $\|x - y\|_{\nabla^2 F(x)} > 0$,*

$$\|x - y\|_{\nabla^2 F(y)} \leq d(r)\|x - y\|_{\nabla^2 F(x)}.$$

As the next lemma shows, quasi-self-concordance is sufficient to ensure this type of local stability.

Lemma 1 (Theorem I (Karimireddy et al., 2018))**.** *If F is α-quasi-self-concordant and convex, then F is $d(r) = \exp(\alpha r)$-locally stable.*

The advantage of local stability is that it ensures that approximate solutions to locally-defined constrained quadratic problems can guarantee progress on the global objective, and we state this more formally in the following theorem. Note that this theorem is similar to (Theorem IV Karimireddy et al., 2018), though we allow for an additive error in the subproblem solves in addition to the multiplicative error, and its proof can be found in Appendix A.

Theorem 1. *Let F be $d(r)$-locally stable and convex, let $\|x^*\| \leq B$, and define $Q_t(\Delta x) := \frac{d(5r_0)}{2}\Delta x^\top \nabla^2 F(x_t)\Delta x + \nabla F(x_t)^\top \Delta x$. Suppose that in each iteration of Algorithm 1, $\|\Delta\tilde{x}_t\| \leq 5r_0$ and*

$$\mathbb{E}Q_t(\Delta\tilde{x}_t) - \min_{\Delta x : \|\Delta x\| \leq \frac{1}{2}r_0} Q_t(\Delta x) \leq \theta\left(Q_t(0) - \min_{\Delta x : \|\Delta x\| \leq \frac{1}{2}r_0} Q_t(\Delta x)\right) + \epsilon.$$

Then for each $T \geq 0$, Algorithm 1 guarantees

$$\mathbb{E}F(x_T) - F^* \leq \mathbb{E}[F(x_0) - F^*]\exp\left(-\frac{Tr_0(1-\theta)}{2Bd(\frac{1}{2}r_0)d(5r_0)}\right) + \frac{2Bd(\frac{1}{2}r_0)d(5r_0)\epsilon}{r_0(1-\theta)}.$$

We have now seen how to turn approximate solutions of constrained least squares problems into an approximate minimizer of the overall objective. We next need to ensure that the output of Algorithm 2 meets the conditions of Theorem 1. As previously discussed, Woodworth et al. (2020) showed that first-order methods can very accurately optimize unconstrained quadratic objectives using a single round of communication; but here we need to optimize a quadratic subject to a norm constraint. Our method Algorithm 2 is thus based on the following idea: the minimizer of the *constrained* problem $\min_{x:\|x\|\leq c} Q(x)$ is the same as the minimizer of the *unconstrained* problem $\min_x Q(x) + \frac{\lambda^*}{2}\|x\|^2$ for some problem-dependent regularization parameter λ^*. While the algorithm does not know what λ^* should be a priori, we show that it can be found with sufficient confidence using binary search. The following theorem, stated and proven in full in Appendix B, provides the relevant guarantees.

Theorem 2 (Informal)**.** *Let $\hat{x}$ be the output of Algorithm 2 for appropriate choice of parameters N and C, and let the output $\hat{x}_\lambda$ of One-Shot-SGD$(A + \lambda I, b)$ satisfy for all $\lambda \geq \bar{\lambda}$ that*

$$\mathbb{E}Q_\lambda(\hat{x}) - \min_x Q_\lambda(x) \leq \epsilon(\lambda) \leq \frac{\lambda(r^*(\lambda)^2 + r_0^2)}{800}.$$

Algorithm 2 CONSTRAINED-QUADRATIC-SOLVER(A, b, r_0)

Inputs: A, b
$\Lambda_1 = \left\{ \bar{\lambda}\left(\frac{3}{2}\right)^{n-1} \,:\, n = 1, \ldots, N \right\}$
$i \leftarrow 1$
while $\Lambda_i \neq \varnothing$ **do**
  $\lambda^{(i)} = \text{Median}(\Lambda_i)$
  for $c = 1, \ldots, C$ **do**
    $\hat{x}^{i,c} = \text{ONE-SHOT-SGD}(A + \lambda^{(i)} I, b)$
  if $\sum_{c=1}^{C} \mathbb{1}_{\{\|\hat{x}^{(i,c)}\| \in [\frac{3}{2}\bar{r},\, \frac{7}{2}\bar{r}]\}} > \frac{C}{2}$ **then**
    $\tilde{x} = \text{ONE-SHOT-SGD}(A + \lambda^{(i)} I, b)$
    Return: $\hat{x} = \min\left\{1,\, \frac{5\bar{r}}{\|\tilde{x}\|}\right\}\tilde{x}$
  else if $\sum_{c=1}^{C} \mathbb{1}_{\{\|\hat{x}^{(i,c)}\| \leq \frac{5}{2}\bar{r}\}} > \frac{C}{2}$ **then**
    $\Lambda_{i+1} = \Lambda_i \setminus \left\{ \lambda' \in \Lambda_i \,:\, \lambda' \geq \lambda^{(i)} \right\}$
  else if $\sum_{c=1}^{C} \mathbb{1}_{\{\|\hat{x}^{(i,c)}\| > \frac{5}{2}\bar{r}\}} > \frac{C}{2}$ **then**
    $\Lambda_{i+1} = \Lambda_i \setminus \left\{ \lambda' \in \Lambda_i \,:\, \lambda' \leq \lambda^{(i)} \right\}$
  else
    Return: $\hat{x} = 0$
  $i \leftarrow i + 1$
$\tilde{x} = \text{ONE-SHOT-SGD}(A + \bar{\lambda} I, b)$
Return: $\hat{x} = \min\left\{1,\, \frac{5\bar{r}}{\|\tilde{x}\|}\right\}\tilde{x}$

Algorithm 3 ONE-SHOT-SGD(A, b)

Initialize: $x_0^1, \ldots, x_0^M = 0$ # Intial iterates on each machine
$Q(x) := \frac{1}{2} x^\top A x + b^\top x$, with stochastic gradient $g(x; z)$[1] # $\mathbb{E}_z[g(x; z)] = \nabla Q(x)$
for Each machine $m = 1, \ldots, M$ in parallel **do**
  for $t = 1, \ldots, T$ **do**
    $x_{t+1}^m = x_t^m - \eta_t g(x; z)$
Return: $\hat{x} = \frac{1}{M \sum_{t=0}^{T} w_t} \sum_{m=1}^{M} \sum_{t=0}^{T} w_t x_t^m$

Then $\|\hat{x}\| \leq 5r_0$ *and*

$$\mathbb{E}Q(\hat{x}) - \min_{x : \|x\| \leq \frac{1}{2} r_0} Q(x) \leq \frac{3}{4}\left(Q(0) - \min_{x : \|x\| \leq \frac{1}{2} r_0} Q(x) \right) + \epsilon(\lambda_{4r_0}) + \frac{\bar{\lambda} r_0^2}{4} \,.$$

We now show that using Local SGD with a single round of communication—i.e. averaging the results of M independent runs of SGD—suffices to solve each least squares problem. Our analysis is based on similar ideas as the Woodworth et al. (2020) result, however additional care must be taken to account for the fact that the stochastic gradient estimates for the quadratic subproblems have variance that scales with the norm of the iterate. The following theorem, which we prove in Appendix C, shows that ONE-SHOT-SGD (Algorithm 3) supplies CONSTRAINED-QUADRATIC-SOLVER with an output $\hat{x}$ that satisfies the conditions of Theorem 2.

Theorem 3 (Informal)**.** *Let* $Q(x) = \frac{1}{2} x^\top A x + b^\top x$ *with* $\lambda I \preceq A \preceq HI$*, and let a stochastic gradient oracle for* Q *be given such that* $\mathbb{E}_z \|g(x; z) - \nabla Q(x)\|^2 \leq \sigma^2 + \rho^2 \|x\|^2$ *for all* x*. Then when* $T \geq \frac{2}{\lambda} \max\left\{H,\, \frac{\rho^2}{\lambda}\right\}$*, a weighted average of the iterates of* ONE-SHOT-SGD *with appropriate stepsizes guarantees*

$$\mathbb{E}Q(\hat{x}) - Q^* \leq 32\lambda \|x^*\|^2 \exp\left(-\frac{T}{8} \min\left\{ \frac{\lambda}{H},\, \frac{\lambda^2}{\rho^2} \right\} \right) + \frac{32(\sigma^2 + \rho^2 \|x^*\|^2)}{\lambda M T} \,.$$

[1] For $A = \nabla^2 F(y)$ and $b = \nabla F(y)$ the gradient $\nabla Q(x) = \nabla^2 F(y) x + \nabla F(y)$ can be estimated using one stochastic Hessian-vector product for F and one stochastic gradient for F.

Finally, having analyzed Algorithms 1, 2, and 3, we put these all together to prove our main theoretical result, whose proof can be found in Appendix D.

Theorem 4 (Informal). *Let F be convex and satisfy Assumption 2. Then with an appropriate choice of the parameters r_0, $\bar{\lambda}$, N, C, and T, the output of Algorithm 1, using Algorithm 2 and Algorithm 3, satisfies*

$$\mathbb{E}[F(x_T)] - F^* \leq \tilde{O}\left(HB^2\left(\exp\left(-\frac{R}{B\alpha}\right) + \exp(-K)\right) + \frac{\sigma B}{\sqrt{MK}} + \frac{HB^2}{RK} + \frac{\rho B^2}{R\sqrt{K}}\right).$$

Comparison with lower bounds and FEDAC

Recent work by Woodworth et al. (2021) established tight lower bounds for stochastic distributed optimization, and their results effectively resolve the min-max complexity of first-order optimization in the convex and smooth setting. They additionally provide lower bounds under other smoothness conditions, including quasi-self-concordance, which are relevant to the current work. Roughly speaking, they show that under Assumption 2, no first-order intermittent communication algorithm can guarantee suboptimality less than

$$\mathbb{E}F(\hat{x}) - F(x^*) \geq \tilde{\Omega}\left(\frac{HB^2}{K^2R^2} + \frac{\sigma B}{\sqrt{MKR}} + \min\left\{\frac{HB^2}{R^2}, \frac{\alpha\sigma B^2}{\sqrt{K}R^2}, \frac{\sigma B}{\sqrt{KR}}\right\}\right).$$

Comparing this lower bound with our guarantee in Theorem 4, we see that our approximate Newton method can, in some regimes, achieve better error than is possible using any first-order method. For example, with $H = B = \sigma = \alpha = 1$, $M \to \infty$, $R = 4\log K$, and K sufficiently large, the upper and lower bounds reduce to

$$\mathbb{E}F(\hat{x}) - F^* \lesssim \frac{1}{K^4} + \exp(-K) + \frac{1}{K\log K} + \frac{\rho}{\sqrt{K}\log K} \approx \frac{\rho}{\sqrt{K}\log K} \tag{5}$$

$$\mathbb{E}F(\hat{x}) - F(x^*) \gtrsim \frac{1}{K^2\log^2 K} + \min\left\{\frac{1}{\log^2 K}, \frac{1}{\sqrt{K}\log^2 K}, \frac{1}{\sqrt{K\log K}}\right\} \approx \frac{1}{\sqrt{K}\log^2 K}. \tag{6}$$

Therefore, when the Hessian-vector product noise, ρ, is small enough, our upper bound can be less than the lower bound. For this to happen, our result does require that ρ be fairly small, but, this lower bound may not be tight, and we are not currently aware of any first-order methods whose guarantee matches this lower bound. Indeed, the best first-order method under higher-order smoothness assumptions is FEDAC (Yuan and Ma, 2020), and in the same parameter regime as above, our algorithm guarantees error on the order of $\rho/(\sqrt{K}\log K)$ while theirs only ensures error $1/(K^{1/3}\log^{4/3} K)$. Therefore, our method can substantially outperform existing first-order methods even when the Hessian-vector noise is moderately large. Our result demonstrates the potential for developing new algorithms using stochastic Hessian-vector products, which may perform better than any first-order method could!

Interestingly, this potential advantage for Hessian-vector product algorithms over first-order methods is specific to distributed optimization. In particular, a Hessian-vector product oracle does *not* provide any advantage for sequential (non-distributed) stochastic smooth convex optimization. This is because the worst-case instances used to establish lower bounds for smooth first-order optimization are quadratic functions (Ghadimi and Lan, 2013), for which Hessian-vector product information is essentially equivalent to stochastic gradient information.

4 Experiments

We begin by presenting a more practical variant of FEDSN called FEDSN-LITE in Algorithm 4. It differs from FEDSN in two major ways: first, it uses ONE-SHOT-SGD (Algorithm 3) as its approximate quadratic solver so that it does not require the search on the regularization strength like Algorithm 2; and in addition, it scales the Newton update before applying it with a stochastic approximation of Newton's decrement (see Boyd et al. (2004); Nesterov (1998)) and a constant step-size α (we use $\alpha = 1.25$ in our experiments). Because of these changes, the newton update is pretty robust to the choice of α and we only need to tune the learning rate of ONE-SHOT-SGD, making it as usable as any other first order method. We also add Polyak momentum to ONE-SHOT-SGD. Note that each step of FEDSN-LITE corresponds to a single round of communication.

Algorithm 4 FEDSN-LITE(F, r_0, α)

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

$\Delta\tilde{x}_t = \text{ONE-SHOT-SGD}(\nabla^2 F(x_t), \nabla F(x_t))$

$\alpha_t := \alpha \left(1 + (\Delta\tilde{x}_t^T \nabla^2 \widetilde{F}(x_t) \Delta\tilde{x}_t)^{1/2}\right)^{-1}$ $\quad$ # $\nabla^2 \widetilde{F}(x_t)$ is s.t. $\mathbb{E}[\nabla^2 \widetilde{F}(x_t)] = \nabla^2 F(x_t)$

Update: $x_{t+1} = x_t + \alpha_t \Delta\tilde{x}_t$

Return: x_T

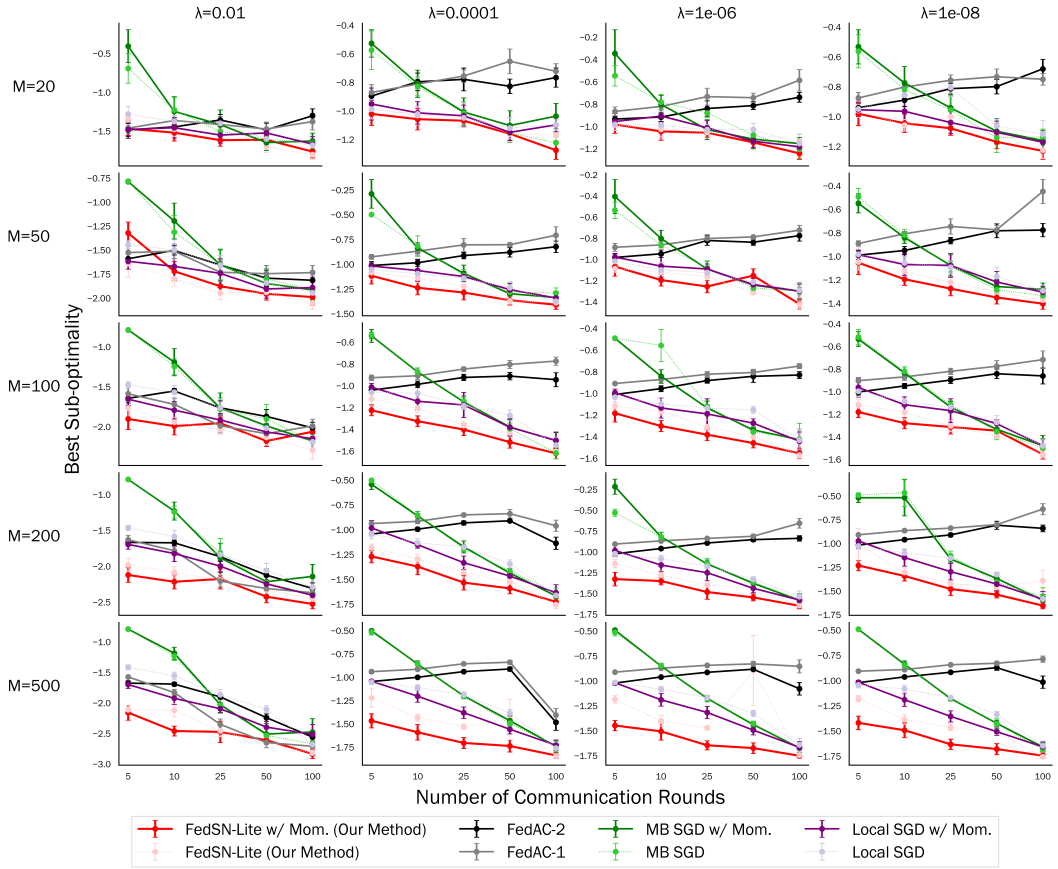

Figure 1: Empirical comparison of FEDSN-LITE (see Section 4) to other methods on the LIBSVM a9a (Chang and Lin, 2011; Dua and Graff, 2017) data-set for λ-regularized logistic regression, as we vary the number of machines used (M), regularization parameter (λ) and the frequency of communication (horizontal axis of each plot), while keeping the total number of steps on each machine (theoretical parallel runtime) fixed at $T = 100$. We compare FEDSN-LITE (with $\alpha = 1.25$) against both the variants of FEDAC proposed in Yuan and Ma (2020), MINI-BATCH SGD (Dekel et al. (2012)), and LOCALSGD (Zinkevich et al. (2010)), the latter two with and without Polyak's momentum. Each algorithm solves a λ-Regularized ERM (RERM) problem, and reports the best relative sub-optimality w.r.t. the optimal minimizer (obtained through a few dozen iterations of exact Newton's method). We search for the best learning rate for each algorithm in $\{0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20\}$ and then run the algorithm 30 times, reporting the mean of these runs, with standard deviation on the error bars. We can see the benefit of using FEDSN-LITE when the number of communication rounds is small (left side of each sub-plot), especially when increasing M and decreasing λ.

We compare FEDSN-LITE (see Figure 1) against the two variants of FEDAC presented in Yuan and Ma (2020), MINI-BATCH SGD, and LOCALSGD, closely following the setup in Yuan and Ma (2020) (c.f., their Fig. 3 and 5) on a λ-regularized logistic regression problem. We observe that upon decreasing regularization and increasing the number of machines, FEDSN-LITE has an edge over all the other algorithms, especially when *communication is sparse*. Note that FEDAC requires the

strong convexity constant λ to determine its hyper-parameters. This may be difficult to approximate in practice if the problem is unregularized, and is a practical limitation of FEDAC, so we experiment over a range of λ-Regularized ERM problems to facilitate a fair comparison.

In Figure 2 we present the same experiment as in Section 4 but with a larger theoretical parallel runtime $T = 1000$. The relative performance of the algorithms is qualitatively similar to the $T = 100$ case.

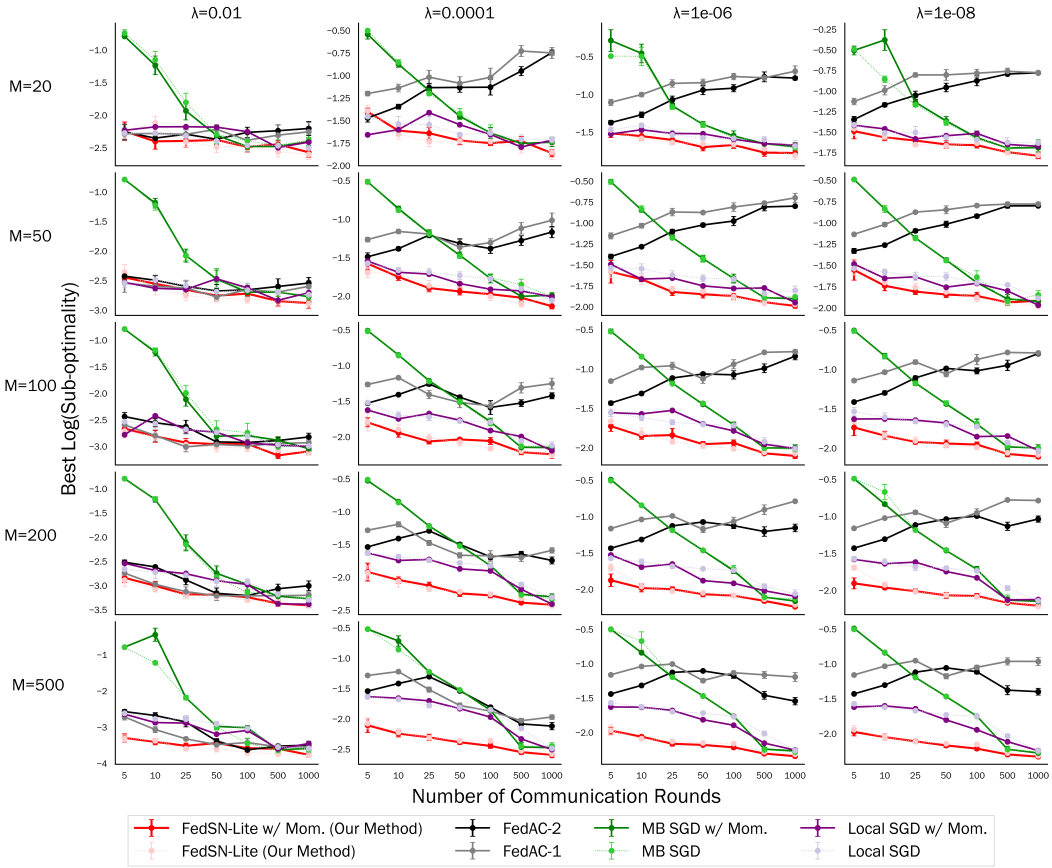

Figure 2: Empirical comparison of FEDSN-LITE (see Section 4) to other methods under similar settings as in fig. 1, though for $T = 1000$. We run each algorithm 10 times with the optimal learning rate, reporting the mean of these runs, with standard deviation on the error bars. Notice again the benefit of using FEDSN-LITE when the number of communication rounds is small (left side of each sub-plot), especially when increasing M and decreasing λ.

5 Future work

In this work, we have shown how to more efficiently optimize convex quasi-self-concordant objectives by leveraging parallel methods for least-squares problems. While our results highlight the gains in this setting by assuming access to a stochastic Hessian-vector product oracle, it would be interesting to explore other instances in which such access could provide additional benefits, possibly even for non-convex settings. Furthermore, just as is it known in the case of stochastic gradient oracle access for general convex objectives (Woodworth et al., 2021), a natural and complementary direction would be to understand the limitations of these oracles for distributed optimization.

References

Naman Agarwal, Zeyuan Allen-Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma. Finding approximate local minima faster than gradient descent. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1195–1199, 2017.

Zeyuan Allen-Zhu. Natasha 2: faster non-convex optimization than sgd. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 2680–2691, 2018.

Yossi Arjevani, Yair Carmon, John C Duchi, Dylan J Foster, Ayush Sekhari, and Karthik Sridharan. Second-order information in non-convex stochastic optimization: Power and limitations. In *Conference on Learning Theory*, pages 242–299. PMLR, 2020.

Francis Bach. Self-concordant analysis for logistic regression. *Electronic Journal of Statistics*, 4: 384–414, 2010.

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Brian Bullins. Highly smooth minimization of non-smooth problems. In *Conference on Learning Theory*, pages 988–1030. PMLR, 2020.

Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford. Accelerated methods for nonconvex optimization. *SIAM Journal on Optimization*, 28(2):1751–1772, 2018.

Yair Carmon, Arun Jambulapati, Qijia Jiang, Yujia Jin, Yin Tat Lee, Aaron Sidford, and Kevin Tian. Acceleration with a ball optimization oracle. *Advances in Neural Information Processing Systems*, 33, 2020.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Greg Coppola. *Iterative parameter mixing for distributed large-margin training of structured predictors for natural language processing*. PhD thesis, The University of Edinburgh, 2015.

Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. Better mini-batch algorithms via accelerated gradient methods. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1647–1655. Curran Associates, Inc., 2011. URL http://papers.nips.cc/paper/4432-better-mini-batch-algorithms-via-accelerated-gradient-methods.pdf.

Rixon Crane and Fred Roosta. Dingo: Distributed newton-type method for gradient-norm optimization. *arXiv preprint arXiv:1901.05134*, 2019.

Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(Jan):165–202, 2012.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Saeed Ghadimi and Guanghui Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization i: A generic algorithmic framework. *SIAM Journal on Optimization*, 22(4):1469–1492, 2012.

Saeed Ghadimi and Guanghui Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization, ii: shrinking procedures and optimal algorithms. *SIAM Journal on Optimization*, 23(4):2061–2089, 2013.

Vipul Gupta, Avishek Ghosh, Michal Derezinski, Rajiv Khanna, Kannan Ramchandran, and Michael Mahoney. Localnewton: Reducing communication bottleneck for distributed learning. *arXiv preprint arXiv:2105.07320*, 2021.

Rustem Islamov, Xun Qian, and Peter Richtárik. Distributed second order methods with fast rates and compressed communication. *arXiv preprint arXiv:2102.07158*, 2021.

Sai Praneeth Karimireddy, Sebastian U Stich, and Martin Jaggi. Global linear convergence of newton's method without strong-convexity or lipschitz gradients. *arXiv preprint arXiv:1806.00413*, 2018.

Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Better communication complexity for local sgd. *arXiv preprint arXiv:1909.04746*, 2019.

LO Mangasarian. Parallel gradient distribution in unconstrained optimization. *SIAM Journal on Control and Optimization*, 33(6):1916–1925, 1995.

Yurii Nesterov. Introductory lectures on convex programming volume i: Basic course. *Lecture notes*, 3(4):5, 1998.

Yurii Nesterov. Implementable tensor methods in unconstrained convex optimization. *Mathematical Programming*, pages 1–27, 2019.

Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

Yurii Nesterov and Boris T Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

Sashank J Reddi, Jakub Konečnỳ, Peter Richtárik, Barnabás Póczós, and Alex Smola. Aide: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879*, 2016.

Ohad Shamir, Nati Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In *International Conference on Machine Learning*, pages 1000–1008. PMLR, 2014.

Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018. URL https://arxiv.org/abs/1805.09767.

Sebastian U Stich and Sai Praneeth Karimireddy. The error-feedback framework: Better rates for sgd with delayed gradients and compressed communication. *arXiv preprint arXiv:1909.05350*, 2019.

Shusen Wang, Fred Roosta, Peng Xu, and Michael W Mahoney. Giant: Globally improved approximate newton method for distributed optimization. *Advances in Neural Information Processing Systems*, 31:2332–2342, 2018.

Blake Woodworth, Jialei Wang, Brendan McMahan, and Nathan Srebro. Graph oracle models, lower bounds, and gaps for parallel stochastic optimization. *arXiv preprint arXiv:1805.10222*, 2018. URL https://arxiv.org/abs/1805.10222.

Blake Woodworth, Kumar Kshitij Patel, Sebastian Stich, Zhen Dai, Brian Bullins, Brendan Mcmahan, Ohad Shamir, and Nathan Srebro. Is local sgd better than minibatch sgd? In *International Conference on Machine Learning*, pages 10334–10343. PMLR, 2020.

Blake Woodworth, Brian Bullins, Ohad Shamir, and Nathan Srebro. The min-max complexity of distributed stochastic convex optimization with intermittent communication. *arXiv preprint arXiv:2102.01583*, 2021.

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

Yuchen Zhang and Lin Xiao. Disco: Distributed optimization for self-concordant empirical loss. In *International Conference on Machine Learning*, pages 362–370. PMLR, 2015.

Fan Zhou and Guojing Cong. On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3219–3227. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/447. URL https://doi.org/10.24963/ijcai.2018/447.

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

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Included as a compressed file in the Supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See the experiment section in the Appendix and figure 1
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See section E in the Appendix and figure 1.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See section E in the Appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] The asset is a freely availabe online ML data-set on LIBSVM/UCI repositories.
   (b) Did you mention the license of the assets? [No] The curators of the data-set didn't specify license information specific to the data-set in their citation policy. Though, they had citation instructions which we've abided by.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our code for reproducing the experiments, and the data from running the experiments which is used to generate the plots in this paper.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] The data-set is free to use after citation.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]