A Stochastic Newton Algorithm for Distributed Convex Optimization

Brian Bullins
Toyota Technological Institute at Chicago
bbullins@ttic.edu

Kumar Kshitij Patel
Toyota Technological Institute at Chicago
kkpatel@ttic.edu

Ohad Shamir
Weizmann Institute of Science
ohad.shamir@weizmann.ac.il

Nathan Srebro
Toyota Technological Institute at Chicago
nati@ttic.edu

Blake Woodworth
Toyota Technological Institute at Chicago
blake@ttic.edu

Abstract

We propose and analyze a stochastic Newton algorithm for homogeneous distributed stochastic convex optimization, where each machine can calculate stochastic gradients of the same population objective, as well as stochastic *Hessian-vector products* (products of an independent unbiased estimator of the Hessian of the population objective with arbitrary vectors), with many such stochastic computations performed between rounds of communication. We show that our method can reduce the number, and frequency, of required communication rounds compared to existing methods without hurting performance, by proving convergence guarantees for quasi-self-concordant objectives (e.g., logistic regression), alongside empirical evidence.

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 where during each round each machine may perform some basic operation (e.g., access the objective by invoking some oracle) K times, and then communicate with all other machines. An important example of this setting is when this basic operation 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, 2019; Woodworth et al., 2020a), Minibatch SGD (Dekel et al., 2012), Minibatch AC-SA (Ghadimi and Lan, 2012), and many others.

We are motivated by the observation of Woodworth et al. (2020a) that for *quadratic* objectives, first-order methods such as one-shot averaging (Zinkevich et al., 2010; Zhang et al., 2013)—a special case of Local SGD with a single round of communication—can optimize the objective to a very high degree of accuracy. This prompts trying to reduce the task of optimizing general convex objectives to a short sequence of quadratic problems. Indeed, this is precisely the idea behind many second-order algorithms including Newton's method (Nesterov and Nemirovskii, 1994), trust-region methods

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

Table 1: Convergence guarantees for different algorithms in the intermittent communication setting. Notation is as follows: H: smoothness; U: third-order-smoothness; σ: stochastic gradient variance; ρ: stochastic Hessian-vector product variance; $g(x;z)$: stochastic gradient oracle; $h(x,u;z')$: stochastic Hessian-vector product oracle (see Section 2 for complete details). For the sake of clarity, we omit additional constants and logarithmic factors.

Algorithm (Reference)	Convergence Rate	Assumption, Oracle Access
Local SGD (Woodworth et al., 2020a)	$\frac{HB^2}{KR} + \frac{\sigma B}{\sqrt{MKR}} + \frac{H^{1/3}\sigma^{2/3}B^{4/3}}{K^{1/3}R^{2/3}}$	A1 $g(x;z)$
FedAc (Yuan and Ma, 2020)	$\frac{HB^2}{KR^2} + \frac{\sigma B}{\sqrt{MKR}} + \min\left\{\frac{H^{1/3}\sigma^{2/3}B^{4/3}}{K^{1/3}R}, \frac{H^{1/2}\sigma^{1/2}B^{3/2}}{K^{1/4}R}\right\}$	A1 $g(x;z)$
Local SGD (Yuan and Ma, 2020)	$\frac{HB^2}{KR} + \frac{\sigma B}{\sqrt{MKR}} + \frac{U^{1/3}\sigma^{2/3}B^{5/3}}{K^{1/3}R^{2/3}}$	A3 (3^{rd}-order Smooth) $g(x;z)$
FedAc (Yuan and Ma, 2020)	$\frac{HB^2}{KR^2} + \frac{\sigma B}{\sqrt{MKR}} + \frac{H^{1/3}\sigma^{2/3}B^{4/3}}{M^{1/3}K^{1/3}R} + \frac{U^{1/3}\sigma^{2/3}B^{5/3}}{K^{1/3}R^{4/3}}$	A3 (3^{rd}-order Smooth) $g(x;z)$
FedSN **(Theorem 1)**	exp. decay + $\frac{HB^2}{KR} + \frac{\sigma B}{\sqrt{MK}} + \frac{\rho B^2}{\sqrt{KR}}$	A2 (QSC) $g(x;z)$, $h(x,u;z')$

(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, for convex F, a linear system of the form $\nabla^2 F(x)\Delta x = -\nabla F(x)$. Unfortunately, this may be prohibitive in a high dimensional setting, and may not even be feasible if F is only accessible through a stochastic oracle in a streaming fashion, as is the case in the setting we consider. To avoid these issues, we reformulate the Newton step as the solution to a convex quadratic problem, $\min_{\Delta x} \frac{1}{2}\Delta x^\top \nabla^2 F(x)\Delta x + \nabla F(x)^\top \Delta x$, which we then solve using one-shot averaging. Conveniently, computing stochastic gradient estimates for this quadratic objective does not require computing the full Hessian matrix, as it only requires stochastic gradients and *stochastic Hessian-vector products*. This is attractive computationally since, for many problems, the cost of computing stochastic Hessian-vector products is similar to the cost of computing stochastic gradients, and both involve similar operations (Pearlmutter, 1994). Furthermore, highlighting the importance of these estimates, recent works have relied on Hessian-vector products to attain faster rates for reaching approximate stationary points in both deterministic (Agarwal et al., 2017; Carmon et al., 2018) and stochastic (Allen-Zhu, 2018; Arjevani et al., 2020) non-convex optimization.

In the context of distributed optimization, second-order methods have shown promise in the empirical risk minimization (ERM) setting, whereby estimates of F are constructed by distributing the component functions of the finite-sum problem across machines. Such methods which leverage this structure have 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 important difference, however, is that these methods work in a batch setting, meaning they allow for repeated access to the same K examples each round on each machine, giving a total of MK samples. In contrast, we work in the stochastic (one-pass, streaming) setting, and so our model independently samples a fresh set of MK examples per round, for a total of MKR examples (see Appendix G.5 for an empirical comparison).

Our results

Our primary algorithmic contribution, which we present in Section 3 (and include a sketch in Appendix A), is the method Federated-Stochastic-Newton (FedSN), a distributed approximate Newton method which leverages the benefits of one-shot averaging for quadratic problems. We provide in Section 3, under the condition of quasi-self-concordance (Bach, 2010), the main guarantees of our method (Theorem 1). In Section 4 we show how, for some regimes in terms of M, K, and R, our method may improve upon the rates of previous first-order methods, including FedAc (Yuan and Ma, 2020). In Section 5, we compare a more practical version of our method, FedSN-Lite (Algorithm 6) against the other methods, showing we can significantly reduce communication compared to other first-order methods.

2 Preliminaries

We consider the following optimization problem:

$$\min_{x \in \mathbb{R}^d} F(x), \tag{1}$$

and throughout we use F^* to denote the minimum of this problem. We further use $\|\cdot\|$ to denote the standard ℓ_2 norm, we let $\|x\|_{\mathbf{A}} := \sqrt{x^\top \mathbf{A} x}$ for a positive semidefinite matrix $\mathbf{A}$, and we let $\mathbf{I}$ denote the identity matrix of order d.

Next, we establish several sets of assumptions, beginning with those which are standard for smooth, stochastic, distributed convex optimization. We would note that we are working in the *homogeneous* distributed setting (i.e., each machine may access the same distribution), rather than the heterogeneous setting (Khaled et al., 2019; Karimireddy et al., 2019; Koloskova et al., 2020; Woodworth et al., 2020b; Khaled et al., 2020).

Assumption 1 (A1)**.**

(a) F is convex, differentiable, and 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 a stochastic first-order oracle in the form of an estimator $g : \mathbb{R}^d \times \mathcal{Z} \mapsto \mathbb{R}^d$, and a distribution $\mathcal{D}$ on $\mathcal{Z}$ such that, for any $x \in \mathbb{R}^d$ queried by the algorithm, the oracle draws $z \sim \mathcal{D}$, and the algorithm observes an estimate $g(x; z)$ that satisfies:

(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., $\mathbb{E}_z \|g(x; z) - \nabla F(x)\|^2 \leq \sigma^2$.

In order to provide guarantees for Newton-type methods, we will require additional notions of smoothness. In particular, we consider α-quasi-self-concordance (QSC) (Bach, 2010), which for convex and three-times differentiable F is satisfied for $\alpha \geq 0$ when, for all $x \in \text{dom}(F)$, $v, u \in \mathbb{R}^d$,

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

where we define

$$\nabla^k F(x)[u_1, u_2, \ldots, u_k] := \frac{\partial^k}{\partial u_1, \partial u_2, \ldots, \partial u_k}\Big|_{t_1=0, t_2=0, \ldots, t_k=0} F(x + t_1 u_1 + t_2 u_2 + \cdots + t_k u_k),$$

for $k \geq 1$, i.e., the k^{th} directional derivative of F at x along the directions $u_1, u_2, \ldots, u_k$. Related to this is the condition of α-self-concordance, which has proven useful for classic problems in linear optimization (Nesterov and Nemirovskii, 1994), whereby for all $x \in \text{dom}(F)$, $u \in \mathbb{R}^d$,

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

Though quasi-self-concordance is perhaps not as widely studied as self-concordance, recent work has brought its usefulness to light in the context of machine learning (Bach, 2010; Karimireddy et al., 2018; Carmon et al., 2020). Notably, for logistic regression, i.e., problems of the form

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

we observe that α-quasi-self-concordance holds with $\alpha \leq \max_i\{\|b_i a_i\|\}$. Interestingly, this function is *not* self-concordant, thus highlighting the importance of introducing the notion of QSC for such problems, and indeed, neither of these conditions implies the other in general.[1]

[1]For the other direction, note that $F(x) = -\ln(x)$ is 1-self-concordant but not quasi-self-concordant.

We now introduce further assumptions in terms of both additional oracle access and other smoothness notions. The following outlines the requirements for the stochastic Hessian-vector products, though we again stress that the practical cost of such an oracle is often on the order of that for stochastic gradients (Pearlmutter, 1994; Allen-Zhu, 2018).

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

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

(b) *We are given access to a stochastic Hessian-vector product oracle in the form of an estimator $h : \mathbb{R}^d \times \mathbb{R}^d \times \mathcal{Z} \mapsto \mathbb{R}^d$, and a distribution $\mathcal{D}$ on $\mathcal{Z}$ such that, for any pair $x, u \in \mathbb{R}^d$ queried by the algorithm, the oracle draws $z' \sim \mathcal{D}$, and the algorithm observes an estimate $h(x,u;z')$ that satisfies:*

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

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

Meanwhile, other works (e.g., Yuan and Ma, 2020) require different control over third-order smoothness and fourth central moment. We do not require this assumption in our analysis, and include it here for comparison.

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

(a) *F is twice-differentiable and U-third-order-smooth, i.e., for all $x, y \in \mathbb{R}^d$, $F(y) \leq F(x) + \nabla F(x)^\top (y-x) + \frac{1}{2}\left\langle \nabla^2 F(x)(y-x),\, y-x \right\rangle + \frac{U}{6}\|y-x\|^3$.*

(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

We begin by describing our main algorithm, FEDSN (Algorithm 1). Namely, our aim is to solve convex minimization problems $\min_x F(x)$, subject to Assumption 2.

Algorithm 1 FEDERATED-STOCHASTIC-NEWTON, a.k.a., FEDSN(x_0)

```
(Operating on objective $F(\cdot)$ with stochastic gradient $g(\cdot;\cdot)$ and Hessian-vector product $h(\cdot;\cdot,\cdot)$
    oracles.)
Input: $x_0 \in \mathbb{R}^d$.
Hyperparameters: $T$: main iterations; and $\bar{\xi}$: local stability (see Table 4).
Output: Approximate solution to $\min_x F(x)$                                   ▷ See Theorem 1
    for $t = 0, 1, \ldots, T-1$ do
        $\Delta\tilde{x}_t$ = CONSTRAINED-QUADRATIC-SOLVER($x_t$)    ▷ Approx. $\min_{u:\|u\|\leq\frac{1}{2}\bar{r}} \frac{\bar{\xi}}{2}u^\top\nabla^2 F(x_t)u + \nabla F(x_t)^\top u$
        Update: $x_{t+1} = x_t + \Delta\tilde{x}_t$
    Return: $x_T$
```

We will rely throughout the paper on several hyperparameter settings and parameter functions, which we collect in Tables 3 and 4. Recall that M is the amount of parallel workers, R is the number of rounds of communication, K is the number of basic operations performed between rounds of communication, and H, B, σ, α, and ρ are as defined in Assumptions 1 and 2.

Among our assumptions, we note in particular the condition of quasi-self-concordance, under which several works have provided efficient optimization methods. For example, Bach (2010) analyzes Newton's method under QSC conditions, in a manner analogous to that of standard self-concordance analyses, to establish its behavior in the region of quadratic $(\log\log(1/\epsilon))$ convergence. More recently, both Karimireddy et al. (2018) and Carmon et al. (2020) have presented methods which rely instead on a trust-region approach, whereby, for a given iterate x_t, each iteration amounts to approximately solving a *constrained* subproblem of the form $\min_{\Delta x:\|\Delta x\|\leq c} \frac{\xi}{2}\Delta x^\top \nabla^2 F(x_t)\Delta x + \nabla F(x_t)^\top \Delta x$, for some $\xi \geq 1$ and problem-dependent radius $c > 0$. This stands in constrast to the

Hyperparameter Setting	Description
$T := \left\lfloor \frac{R}{4\zeta} \log^2\left(\left(\frac{R}{\zeta}\right)\right) \right\rfloor$ (for $\zeta = 4096 + 4(80 + 32\log K + 24\log(1+2\alpha B))^2$)	Main iterations
$\beta := 0$	Momentum
$\bar{r} := \min\left\{\frac{32B}{T}\log(TK),\ \frac{1}{5\alpha}\right\}$	Trust-region radius
$\bar{\xi} := \exp(\alpha\bar{r})$	Local stability
$\bar{\lambda} := \max\left\{\frac{2eH}{K-2},\ \frac{2\rho}{\sqrt{K}},\ \frac{32eH\log(51200)}{K},\ \frac{4\rho\sqrt{2\log(51200)}}{\sqrt{K}},\ \frac{320\sqrt{2}\rho}{\sqrt{MK}},\ \frac{320\sigma}{\bar{r}\sqrt{MK}},\ \frac{8eH}{K-16}\right\}$	Regularization bound
$N := \left\lceil 1 + \frac{5}{2}\log\frac{H(B+5T\bar{r})}{3\bar{\lambda}\bar{r}} \right\rceil$	Binary search iterations
$C := \left\lceil 8\log\left(\lceil \log_2 N\rceil \left(4 + \frac{eH}{\bar{\lambda}} + \frac{80H(B+5T\bar{r})}{\bar{\lambda}\bar{r}}\right)\right)\right\rceil$	Reg. quadratic repetitions

Table 2: Hyperparameters T, β, $\bar{r}$, $\bar{\xi}$, $\bar{\lambda}$, N, and C, as used by FEDSN and its subroutines.

Parameter Function	Description
$\eta_k(\lambda) := \begin{cases} \eta_\lambda & K \le \frac{2}{\lambda}\max\left\{\bar{\xi}H+\lambda,\ \frac{\rho^2}{\lambda}\right\} \text{ or } k < \frac{K}{2} \\ \frac{4}{\lambda\left(\frac{8}{\lambda}\max\left\{\bar{\xi}H+\lambda,\ \frac{\rho^2}{\lambda}\right\}+k-\frac{K}{2}\right)} & K > \frac{2}{\lambda}\max\left\{\bar{\xi}H+\lambda,\ \frac{\rho^2}{\lambda}\right\} \text{ and } k \ge \frac{K}{2} \end{cases}$	Reg. quad. stepsizes
$w_k(\lambda) := \begin{cases} (1-\lambda\eta_\lambda+\eta_\lambda^2\rho^2)^{-k-1} & K \le \frac{2}{\lambda}\max\left\{\bar{\xi}H+\lambda,\ \frac{\rho^2}{\lambda}\right\} \\ 0 & K > \frac{2}{\lambda}\max\left\{\bar{\xi}H+\lambda,\ \frac{\rho^2}{\lambda}\right\} \text{ and } k < \frac{K}{2} \\ \frac{8}{\lambda}\max\left\{\bar{\xi}H+\lambda,\ \frac{\rho^2}{\lambda}\right\} + k - \frac{K}{2} - 1 & K > \frac{2}{\lambda}\max\left\{\bar{\xi}H+\lambda,\ \frac{\rho^2}{\lambda}\right\} \text{ and } k \ge \frac{K}{2} \end{cases}$	Reg. quad. weights

Table 3: Parameter functions $\eta_k(\lambda)$ and $w_k(\lambda)$, as used by FEDSN and its subroutines, where $\bar{\xi}$ is as defined in Table 4, and where η_λ denotes $\eta(\lambda) := \frac{1}{2}\min\left\{\frac{1}{\bar{\xi}H+\lambda},\ \frac{\lambda}{\rho^2}\right\}$.

unconstrained minimization problem $\min_{\Delta x} \frac{\xi}{2}\Delta x^\top \nabla^2 F(x)\Delta x + \nabla F(x)^\top \Delta x$, which, as we may recall, forms the basis of the standard (damped) Newton method. Carmon et al. (2020) further use their trust-region subroutine to approximately implement a certain ℓ_2-ball minimization oracle, which they combine with an acceleration scheme (Monteiro and Svaiter, 2013).

These results show, at a high level, that as long as the radius of the constrained quadratic (trust-region) subproblem is not too large, it is possible to make sufficient progress on the *global* problem by approximately solving the quadratic subproblem. Our method proceeds in a similar fashion: each iteration of Algorithm 1 provides an approximate solution to a constrained quadratic problem. To begin, we follow Karimireddy et al. (2018) in defining $\delta(r)$-local (Hessian) stability.

Definition 1. *Let $\delta : \mathbb{R}^+ \mapsto \mathbb{R}^+$. We say that a twice-differentiable and convex function F is $\delta(r)$-locally stable if, for any $r > 0$ and any $x, y \in \mathbb{R}^d$ $(x \neq y)$ such that $\|x - y\| \le r$ and $\|x - y\|_{\nabla^2 F(x)} > 0$, we have $\|x - y\|_{\nabla^2 F(y)} \le \delta(r)\|x - y\|_{\nabla^2 F(x)}$.*

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

Lemma 1 (Theorem I (Karimireddy et al., 2018)). *If F is α-quasi-self-concordant, then F is $\delta(r) = \exp(\alpha r)$-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 lemma. Note that this lemma 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 B.

Lemma 2. *Let F satisfy Assumption 2 and be $\delta(r)$-locally stable for $\delta : \mathbb{R}^+ \mapsto \mathbb{R}^+$, let $x_0 \in \mathbb{R}^d$ be as input to* FedSN *(Algorithm 1), let $c > 0$, let $\theta \in [0, 1)$, and define $Q_t(\Delta x) := \frac{\delta(5c)}{2}\Delta x^\top \nabla^2 F(x_t)\Delta x + \nabla F(x_t)^\top \Delta x$, where x_t is the t^{th} iterate of Algorithm 1. Furthermore, suppose we are given that in each iteration of Algorithm 1, $\|\Delta\tilde{x}_t\| \leq 5c$ and*

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

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

$$\mathbb{E}F(x_T) - F^* \leq \mathbb{E}[F(x_0) - F^*]\exp\left(-\frac{Tc(1-\theta)}{2B\delta(\frac{1}{2}c)\delta(5c)}\right) + \frac{2B\delta(\frac{1}{2}c)d(5c)\epsilon}{c(1-\theta)}\,.$$

We have now seen how to turn approximate solutions of constrained quadratic problems into an approximate minimizer of the overall objective. We next need to ensure that the output of our method Constrained-Quadratic-Solver (Algorithm 2) meets the conditions of Lemma 2. As previously discussed, Woodworth et al. (2020a) showed that first-order methods can very accurately optimize *unconstrained* quadratic objectives using a single round of communication; however, here we need to optimize a quadratic problem subject to a norm constraint. Our constrained quadratic solver 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. Lemma 3, proven in Appendix C, provides the relevant guarantees.

Lemma 3. *Let F satisfy Assumption 2, let x be as input to* Constrained-Quadratic-Solver *(Algorithm 2), let $\bar{\xi}$ be as in Table 4, define*

$$Q(u) := \frac{\bar{\xi}}{2}u^\top \nabla^2 F(x)u + \nabla F(x)^\top u, \qquad Q_\lambda(u) := \frac{1}{2}u^\top(\bar{\xi}\nabla^2 F(x) + \lambda\mathbf{I})u + \nabla F(x)^\top u,$$

$$u^*_\lambda := \arg\min_u Q_\lambda(u), \qquad r^*(\lambda) := \|u^*_\lambda\|,$$

and let λ_r denote, for any $r > 0$, the value such that $r^*(\lambda_r) = r$. Let $\hat{u}$ be the output of Algorithm 2 for hyperparameters $\bar{r}$, $\bar{\xi}$, $\bar{\lambda}$, N and C as in Table 4, and suppose the output $\tilde{u}_\lambda$ of Regularized-Quadratic-Solver(x, λ) *satisfies for all $\lambda \geq \bar{\lambda}$ that*

$$\mathbb{E}Q_\lambda(\tilde{u}_\lambda) - \min_u Q_\lambda(u) \leq \epsilon(\lambda) := \frac{\lambda(r^*(\lambda)^2 + \bar{r}^2)}{800}\,.$$

Then $\|\hat{u}\| \leq 5\bar{r}$ and

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

We now show that using one-shot averaging (Zinkevich et al., 2010; Zhang et al., 2012) with M machines—i.e., averaging the results of M independent runs of SGD—suffices to solve each quadratic problem to the desired accuracy. The following lemma, which we prove in Appendix D, establishes that Regularized-Quadratic-Solver (Algorithm 3) supplies Algorithm 2 with an output $\hat{u}$ that satisfies the conditions of Lemma 3.

Algorithm 2 CONSTRAINED-QUADRATIC-SOLVER(x)

```
(Operating on objective $F(\cdot)$ with stochastic gradient $g(\cdot;\cdot)$ and Hessian-vector product $h(\cdot;\cdot,\cdot)$ oracles.)
Input: $x \in \mathbb{R}^d$.
Hyperparameters: $\bar{r}$: trust-region radius; $\bar{\xi}$: local stability; $\bar{\lambda}$: regularization bound; $N$: binary search iterations; and $C$: reg. quadratic repetitions (see Table 4).
Output: Approximate solution to $\min_{u:\|u\|\le\frac{1}{2}\bar{r}} \frac{\bar{\xi}}{2}u^\top \nabla^2 F(x) u + \nabla F(x)^\top u$      ▷ See Lemma 3
  $\Lambda_1 = \left\{\bar{\lambda}\left(\frac{3}{2}\right)^{n-1} : n = 1,\dots,N\right\}$
  $i \leftarrow 1$
  while $\Lambda_i \ne \varnothing$ do
    $\lambda^{(i)} = \text{Median}(\Lambda_i)$
    for $c = 1,\dots,C$ do
      $\tilde{u}^{(i,c)} = $ REGULARIZED-QUADRATIC-SOLVER$(x, \lambda^{(i)})$
    if $\left|\left\{\tilde{u}^{(i,c)} : \left\|\tilde{u}^{(i,c)}\right\| \in \left[\frac{3}{2}\bar{r}, \frac{7}{2}\bar{r}\right]\right\}\right| > \frac{C}{2}$ then
      $\tilde{u} = $ REGULARIZED-QUADRATIC-SOLVER$(x, \lambda^{(i)})$
      Return: $\hat{u} = \min\left\{1, \frac{5\bar{r}}{\|\tilde{u}\|}\right\}\tilde{u}$
    else if $\left|\left\{\tilde{u}^{(i,c)} : \left\|\tilde{u}^{(i,c)}\right\| \le \frac{5}{2}\bar{r}\right\}\right| > \frac{C}{2}$ then
      $\Lambda_{i+1} = \left\{\lambda' \in \Lambda_i : \lambda' < \lambda^{(i)}\right\}$
    else if $\left|\left\{\tilde{u}^{(i,c)} : \left\|\tilde{u}^{(i,c)}\right\| > \frac{5}{2}\bar{r}\right\}\right| > \frac{C}{2}$ then
      $\Lambda_{i+1} = \left\{\lambda' \in \Lambda_i : \lambda' > \lambda^{(i)}\right\}$
    else
      Return: $\hat{u} = 0$
    $i \leftarrow i + 1$
  $\tilde{u} = $ REGULARIZED-QUADRATIC-SOLVER$(x, \bar{\lambda})$
  Return: $\hat{u} = \min\left\{1, \frac{5\bar{r}}{\|\tilde{u}\|}\right\}\tilde{u}$
```

Lemma 4. *Let F satisfy Assumption 2, let $x \in \mathbb{R}^d$, $\lambda \in \mathbb{R}^+$ be as input to* REGULARIZED-QUADRATIC-SOLVER *(Algorithm 3), let $Q_\lambda(u) = \frac{1}{2}u^\top(\bar{\xi}\nabla^2 F(x) + \lambda \mathbf{I})u + \nabla F(x)^\top u$, let $Q_\lambda^* := \min_u Q_\lambda(u)$, let $u^* := \arg\min_u Q_\lambda(u)$, and let stochastic first-order and stochastic Hessian-vector product oracles for F, as defined in Assumptions 1 and 2, respectively, be available for each call to* REGULARIZED-QUADRATIC-GRADIENT-ACCESS *(Algorithm 4), for either Case 1* (`Different-Samples`) *or Case 2* (`Same-Sample`). *Let $\hat{u}$, as output by Algorithm 3, be a weighted average of the iterates of M independent runs of SGD with stepsizes $\eta_0(\lambda),\dots,\eta_{K-1}(\lambda)$, i.e., $\hat{u} = \frac{1}{M\sum_{k=0}^{K-1} w_k}\sum_{m=1}^{M}\sum_{k=0}^{K-1} w_k u_k^m$. Then, for both Cases 1 and 2,*

$$\mathbb{E}Q_\lambda(\hat{u}) - Q_\lambda^* \le \begin{cases} 2\max\left\{\bar{\xi}H + \lambda, \frac{\rho^2}{\lambda}\right\}\|u^*\|^2 \min\left\{\frac{1}{K}, \exp\left(-\frac{K+1}{4}\min\left\{\frac{\lambda}{\bar{\xi}H+\lambda}, \frac{\lambda^2}{\rho^2}\right\}\right)\right\} \\ \qquad + \frac{2(\sigma^2+\rho^2\|u^*\|^2)}{\lambda M K} \quad \text{if } K \le \frac{2}{\lambda}\max\left\{\bar{\xi}H + \lambda, \frac{\rho^2}{\lambda}\right\} \\ 96\lambda\|u^*\|^2 \exp\left(-\frac{K}{8}\min\left\{\frac{\lambda}{\bar{\xi}H+\lambda}, \frac{\lambda^2}{\rho^2}\right\}\right) + \frac{96(\sigma^2+\rho^2\|u^*\|^2)}{\lambda M K} \\ \qquad \text{if } K > \frac{2}{\lambda}\max\left\{\bar{\xi}H + \lambda, \frac{\rho^2}{\lambda}\right\}. \end{cases}$$

Our analysis for Algorithm 3 is based on ideas similar to those of Woodworth et al. (2020a), whereby the algorithm may access the stochastic oracles via REGULARIZED-QUADRATIC-GRADIENT-ACCESS (Algorithm 4). However, additional care must be taken to account for the fact that Algorithm 4 supplies stochastic gradient estimates of the *quadratic subproblems* $Q_\lambda(u)$ as per the oracles models described in Assumptions 1 and 2. Thus, the estimates—based in part on stochastic Hessian-vector products—have variance that scales with the norm of the respective iterates of Algorithm 3 (see Assumption 2(b.ii)).

Algorithm 3 REGULARIZED-QUADRATIC-SOLVER(x, λ)

```
(Operating on objective $F(\cdot)$ with stochastic gradient $g(\cdot;\cdot)$ and Hessian-vector product $h(\cdot;\cdot,\cdot)$
    oracles.)
Input: $x \in \mathbb{R}^d$, $\lambda \in \mathbb{R}^+$.
Hyperparameters: $\beta$: momentum; $\bar{\xi}$: local stability; and parameter functions $\eta_k(\lambda)$, $w_k(\lambda)$ (see
    Tables 3 and 4).
Output: Approximate solution to $\min_u Q_\lambda(u) = \frac{1}{2}u^\top(\bar{\xi}\nabla^2 F(x) + \lambda\mathbf{I})u + \nabla F(x)^\top u$    ▷ See
    Lemma 4
    Initialize: $u_0^1, \ldots, u_0^M = 0$    ▷ Initial iterates on each machine
    for Each machine $m = 1, \ldots, M$ in parallel do
        for $k = 0, \ldots, K-1$ do
            $\gamma(u_k^m; z_k^m, z_k^{'m}) = $ REGULARIZED-QUADRATIC-GRADIENT-ACCESS$(x, u_k^m, \lambda)$
            $u_{k+1}^m = u_k^m - \eta_k(\lambda)\gamma(u_k^m; z_k^m, z_k^{'m}) + \mathbb{1}_{\{k>0\}}\beta(u_k^m - u_{k-1}^m)$ [2]
    Return: $\tilde{u} = \frac{1}{M\sum_{k=1}^K w_k(\lambda)} \sum_{m=1}^M \sum_{k=1}^K w_k(\lambda) u_k^m$
```

We also note two possible cases for the oracle access: Case 1 (`Different-Samples`) in Algorithm 4 requires both a call to a stochastic first-order oracle (which draws $z \sim \mathcal{D}$) and a call to a stochastic Hessian-vector product oracle (which draws a different $z' \sim \mathcal{D}$); while Case 2 (`Same-Sample`) allows both stochastic estimators to be observed for the same random sample $z \sim \mathcal{D}$. These cases differ by only a small constant factor in the final convergence rate, and we base our practical method on this single sample model. We refer the reader to Appendix G.4 for discussion of these settings.

Algorithm 4 REGULARIZED-QUADRATIC-GRADIENT-ACCESS(x, u, λ)

```
(Operating on objective $F(\cdot)$ with stochastic gradient $g(\cdot;\cdot)$ and Hessian-vector product $h(\cdot;\cdot,\cdot)$
    oracles.)
Input: $x, u \in \mathbb{R}^d$, $\lambda \in \mathbb{R}^+$.
Hyperparameters: $\bar{\xi}$: local stability (see Table 4).
Output: $\gamma(u; z, z')$ s.t. $\mathbb{E}_{z,z'}[\gamma(u; z, z')] = \nabla Q_\lambda(u)$ and $\mathbb{E}_{z,z'}\|\gamma(u; z, z') - \nabla Q_\lambda(u)\|^2 \le \sigma^2 +$
    $\rho^2\|u\|^2$
    Case 1: Different-Samples ($z, z'$ drawn independently for each stochastic oracle)
        • Query the stochastic first-order oracle at $x$ (as in Assumption 1(c)), so that the oracle
          draws $z \sim \mathcal{D}$, and observe $g(x; z)$
        • Query the stochastic Hessian-vector product oracle at $x$ and $u$ (as in Assumption 2(b)), so
          that the oracle draws $z' \sim \mathcal{D}$, and observe $h(x, u; z')$
    Case 2: Same-Sample (Same $z' = z$ used for both stochastic oracles)
        • Query the stochastic first-order oracle at $x$ (as in Assumption 1(c)), so that the oracle
          draws $z \sim \mathcal{D}$, and observe $g(x; z)$
        • Query the stochastic Hessian-vector product oracle at $x$ and $u$ (as in Assumption 2(b)) for
          $z' = z$, and observe $h(x, u; z')$
    $\gamma(u; z, z') := \bar{\xi}h(x, u; z') + \lambda u + g(x; z)$
    Return: $\gamma(u; z, z')$
```

Finally, having analyzed Algorithms 1, 2, 3, and 4, we may put them all together to provide our main theoretical result, whose proof can be found in Appendix E.

Theorem 1. *Let F satisfy Assumption 2. Then, for $K \ge 175$ and $R \ge \tilde{\Omega}(1)$, and for hyperparameters T, β, $\bar{r}$, $\bar{\xi}$, $\bar{\lambda}$, N, C and parameter functions $\eta_k(\lambda)$, $w_k(\lambda)$ as in Tables 3 and 4, the output of* FEDSN *(Algorithm 1) with initial point $x_0 \in \mathbb{R}^d$, using Algorithms 2, 3, and 4 (for both Cases 1 and 2)*

[2] We add heavy-ball/Polyak momentum in this step with momentum parameter β. Our theoretical results do not require any momentum, and thus FEDSN is analyzed for $\beta = 0$. For our experiments we compare the algorithms both with and without momentum, i.e., $\beta = 0$ or optimally tuned $\beta \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$.

satisfies

$$\mathbb{E}[F(x_T)] - F^* \le HB^2\left(\exp\left(-\frac{R}{\tilde{O}(\alpha B)}\right) + \exp\left(-\frac{K}{O(1)}\right)\right) + \tilde{O}\left(\frac{\sigma B}{\sqrt{MK}} + \frac{HB^2}{KR} + \frac{\rho B^2}{\sqrt{K}R}\right),$$

where $\tilde{\Omega}$, $\tilde{O}$ hide terms logarithmic in R, K, and αB.

4 Comparison with related methods and lower bounds

In this section, we compare our algorithm's guarantees with those of FEDAC, and we include additional comparisons in Appendix F. A difficulty in making this comparison is determining the "typical" relative scale of the parameters H, σ, U, α, and ρ. Drawing inspiration from training generalized linear models, we consider a natural scaling of the parameters that arises when the objective has the form $F(x) = \mathbb{E}_z \ell(\langle x, z\rangle)$, where $|\ell'|$, $|\ell''|$, and $|\ell'''|$ are $O(1)$, and where $\|z\| \le D$; this holds, e.g., for logistic regression problems (see (2)). In this case, upper bounds on the derivatives of F will generally scale with $\|z\|$. So if we assume that $\|z\| \le D$ for some D, then the derivatives of F would scale as $\|\nabla F(x)\| \lesssim D$, $\|\nabla^2 F(x)\|_{\text{op}} \lesssim D^2$, and $\|\nabla^3 F(x)\|_{\text{op}} \lesssim D^3$, where $\|\cdot\|_{\text{op}}$ denotes the operator norm. Thus, we will take $H = D^2$, $\sigma = D$, $U = D^3$, $\alpha = D$, and $\rho = D^2$. These parameters could have different relationships, but we focus on this regime for simplicity.

In addition to working within this natural scaling, we consider the case where we have access to sufficient machines (i.e., $M \gtrsim \frac{KR^3}{D^2B^2}$) and for K large enough. We explore various regimes w.r.t. both the number of rounds of communication R and the "size" of the problem DB. Thus, ignoring constants and terms logarithmic in R, K, and αB, our upper bound from Theorem 1 reduces to

$$\mathbb{E}F(\hat{x}) - F^* \lesssim D^2B^2 \exp\left(-\frac{R}{DB}\right) + D^2B^2\exp(-K) + \frac{D^2B^2}{KR^{3/2}} + \frac{D^2B^2}{KR} + \frac{D^2B^2}{\sqrt{K}R} \approx \frac{D^2B^2}{\sqrt{K}R}.$$

Comparison with FEDAC

The previous best known first-order distributed method under third-order smoothness assumptions is FEDAC (Yuan and Ma, 2020), an accelerated variant of Local SGD, which achieves a guarantee of

$$\mathbb{E}F(\hat{x}) - F^* \le \tilde{O}\left(\frac{HB^2}{KR^2} + \frac{\sigma B}{\sqrt{MKR}} + \frac{H^{1/3}\sigma^{2/3}B^{4/3}}{M^{1/3}K^{1/3}R} + \frac{U^{1/3}\sigma^{2/3}B^{5/3}}{K^{1/3}R^{4/3}}\right).$$

For the setting as outlined above, this bound reduces to

$$\mathbb{E}F(\hat{x}) - F^* \lesssim \frac{D^2B^2}{KR^2} + \frac{D^{5/3}B^{5/3}}{K^{1/3}R^{4/3}}.$$

In the case where DB is not too large ($DB \lesssim K^2R^2$), the dominant term for FEDAC is $\frac{D^{5/3}B^{5/3}}{K^{1/3}R^{4/3}}$, and so we see that our algorithm improves upon FEDAC as long as $R \lesssim \frac{\sqrt{K}}{DB}$, whereas for $R \gtrsim \frac{\sqrt{K}}{DB}$, FEDAC provides better guarantees than FEDSN.

Comparison with first-order lower bounds

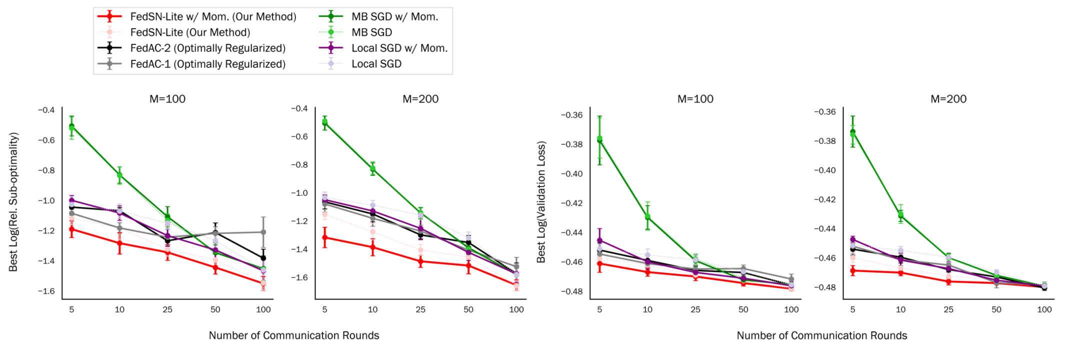
Woodworth et al. (2021) provide lower bounds under other smoothness conditions, including quasi-self-concordance, which are relevant to the current work. Roughly speaking, they show that under Assumption 2(a), no first-order intermittent communication algorithm can guarantee suboptimality less than (ignoring constant and $\log M$ factors)

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

In the same parameter regime as above, the lower bound reduces to

$$\mathbb{E}F(\hat{x}) - F^* \gtrsim \frac{D^2B^2}{K^2R^2} + \min\left\{\frac{D^2B^2}{\sqrt{K}R^2}, \frac{DB}{\sqrt{KR}}\right\}.$$

Comparing this lower bound with our guarantee in Theorem 1, we see that, when $DB = O(1)$ and the number of rounds of communication is small (e.g., $R = O(\log K)$), our approximate Newton method can (ignoring $\log K$ factors) achieve an upper bound of $\mathbb{E}F(\hat{x}) - F^* \lesssim 1/\sqrt{K}$. Therefore, in this important regime, FEDSN matches the lower bound under Assumption 2(a), albeit using a stronger oracle. No prior work has matched this lower bound, and so we do not know whether such an oracle is necessary in order to achieve it, or if perhaps the stronger oracle allows for breaking it.



(a) (b)

Figure 1: Empirical comparison of FEDSN-LITE (Algorithm 6) to other methods (see Appendix G.1) on the LIBSVM a9a (Chang and Lin, 2011; Dua and Graff, 2017) dataset for minimizing: (a) in-sample, and (b) out-of-sample unregularized logistic regression loss using $M \in \{100, 200\}$ machines. We vary the frequency of communication (horizontal axis of each plot), while keeping the total number of steps on each machine (theoretical parallel runtime) fixed at $KR = 100$. Thus, every point in the sub-plot is a separately tuned instance of an algorithm, where each algorithm besides FEDAC solves an unregularized ERM problem and reports (a) the best relative sub-optimality w.r.t. the optimal minimizer and (b) the best validation loss on a held-out dataset. All results are averaged over multiple runs (see Appendix G.3 for full details).

5 Experiments

In Appendix G.1 we present a more practical variant of FEDSN called FEDSN-LITE (Algorithm 6), which does away with the search over the regularization parameter as in Algorithm 2. We compare FEDSN-LITE against the two variants of FEDAC (Algorithm 7, Yuan and Ma (2020)), Minibatch SGD (Algorithm 9, Dekel et al. (2012)), and Local SGD (Algorithm 8, Zinkevich et al. (2010)). We also study the effect of adding Polyak's momentum, which we denote by β, to these algorithms (see Appendix G.1). FEDAC is mainly presented and analyzed for strongly convex functions by Yuan and Ma (2020). In fact, they assume the knowledge of the strong convexity constant to tune FEDAC, which is typically hard to know unless the function is explicitly regularized. To handle general convex functions, Yuan and Ma (2020) build some *internal regularization* into FEDAC (see Appendix E.2 in their paper). However, their hyperparameter recommendations in this setting also depend on unknowns such as the smoothness of the function and the variance of the stochastic gradients. This poses a difficulty in comparing FEDAC to the other algorithms, which do not require the knowledge of these unknowns.

To overcome this we take the more carefully optimized version of FEDAC for strongly convex functions and tune its internal regularization and learning rate. This emulates the setting where the objective is assumed to be just convex but FEDAC sees a strongly convex function instead. In our experiments in Figure 6, we notice that FEDSN-LITE is either competitive with or outperforms the other baselines. This is especially true for the sparse communication settings, which are of most practical interest. A more comprehensive set of experiments can be found in Appendix G.2 along with full implementational details in Appendix G.3.[3]

6 Conclusion

In this work, we have shown how to more efficiently optimize convex quasi-self-concordant objectives by leveraging parallel methods for quadratic problems. Our method can, in some parameter regimes, improve upon existing stochastic methods while maintaining a similar computational cost, and we have further seen how our method may provide empirical improvements in the low communication regime. It remains open whether the same guarantees we achieve here can also be achieved using only independent stochastic gradients (a single stochastic gradient on each sample), or whether in the distributed stochastic setting access to Hessian-vector products is strictly more powerful than access to only independent stochastic gradients.

[3]Code is availabe at https://github.com/kishinmh/Inexact-Newton.

Acknowledgements. This research was partially supported by NSF-BSF award 1718970. BW is supported by a Google Research PhD Fellowship. We also thank all the anonymous reviewers for their time and suggestions.

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 *Advances in 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 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. In *Advances in Neural Information Processing Systems*, volume 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 *Advances in Neural Information Processing Systems*, volume 24, pages 1647–1655, 2011.

Rixon Crane and Fred Roosta. Dingo: Distributed Newton-type method for gradient-norm optimization. In *Advances in Neural Information Processing Systems*, volume 32, 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.

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.

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

Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. First analysis of local gd on heterogeneous data. *arXiv preprint arXiv:1909.04715*, 2019.

Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Tighter theory for local sgd on identical and heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, pages 4519–4529. PMLR, 2020.

Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. A unified theory of decentralized sgd with changing topology and local updates. In *International Conference on Machine Learning*, pages 5381–5393. PMLR, 2020.

Guanghui Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133(1-2):365–397, 2012.

Renato DC Monteiro and Benar Fux Svaiter. An accelerated hybrid proximal extragradient method for convex optimization and its implications to second-order methods. *SIAM Journal on Optimization*, 23(2):1092–1125, 2013.

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.

Barak A Pearlmutter. Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147–160, 1994.

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. In *International Conference on Learning Representations*, 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, 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, 2020a.

Blake Woodworth, Brian Bullins, Ohad Shamir, and Nathan Srebro. The min-max complexity of distributed stochastic convex optimization with intermittent communication. In *Conference on Learning Theory*. PMLR, 2021.

Blake E Woodworth, Jialei Wang, Adam Smith, Brendan McMahan, and Nati Srebro. Graph oracle models, lower bounds, and gaps for parallel stochastic optimization. *Advances in Neural Information Processing Systems*, 31:8496–8506, 2018.

Blake E Woodworth, Kumar Kshitij Patel, and Nati Srebro. Minibatch vs local sgd for heterogeneous distributed learning. *Advances in Neural Information Processing Systems*, 33:6281–6292, 2020b.

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.

Yuchen Zhang, Martin J Wainwright, and John C Duchi. Communication-efficient algorithms for statistical optimization. In *Advances in Neural Information Processing Systems*, pages 1502–1510, 2012.

Yuchen Zhang, John C Duchi, and Martin J Wainwright. Communication-efficient algorithms for statistical optimization. *The Journal of Machine Learning Research*, 14(1):3321–3363, 2013.

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.

Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 2595–2603, 2010.

A Algorithm Sketch

Here we provide a sketch of our main algorithm FEDERATED-STOCHASTIC-NEWTON (a.k.a. FEDSN).

Algorithm (Sketch) FEDERATED-STOCHASTIC-NEWTON, a.k.a., FEDSN(x_0)

```
(Operating on objective $F(\cdot)$ with stochastic gradient $g(\cdot;\cdot)$ and Hessian-vector product $h(\cdot;\cdot,\cdot)$
  oracles.)
Input: $x_0 \in \mathbb{R}^d$.
Hyperparameters: $T$: main iterations; $\bar{\xi}$: local stability parameter; and $N$: binary search iterations
  (see Table 4).
Output: Approximate solution to $\min_x F(x)$
  for $t = 0, 1, \ldots, T-1$ do
    $\Lambda = \{\lambda_1, \ldots, \lambda_N\}$                          ▷ Set of regularization parameters
    while $\Lambda \neq \varnothing$ do
      $\lambda = \text{Median}(\Lambda)$
      Define: $Q_\lambda(u) = \frac{1}{2}u^\top(\bar{\xi}\nabla^2 F(x) + \lambda\mathbf{I})u + \nabla F(x)^\top u$ ▷ Regularized quadratic subproblem
      Run SGD on $Q_\lambda(u)$ independently on each machine (starting with $u_0 = 0$) for $K$ steps,
      then communicate the final iterate $u_K^m$ from each machine $m$, and average them to obtain
      $\tilde{u} = \frac{1}{M}\sum_{m=1}^M u_K^m$.
      if Sufficiently good $\lambda$ is found then
        $\Delta\tilde{x}_t = \tilde{u}$
        break
      else
        Reduce the size of $\Lambda$ by half.
    Update: $x_{t+1} = x_t + \Delta\tilde{x}_t$
  Return: $x_T$
```

B Analysis of Algorithm 1

We will use the following notation in our analysis:

$$Q_t^\sigma(\Delta x) = \frac{\sigma}{2}\Delta x^\top \nabla^2 F(x_t)\Delta x + \nabla F(x_t)^\top \Delta x. \tag{3}$$

Lemma 5 (Lemma 5 (Karimireddy et al., 2018)). *Let F be $\delta(r)$-locally stable for a given $\delta : \mathbb{R}^+ \mapsto \mathbb{R}^+$, let $\|x^*\| \leq B$, let $r > 0$, let $\gamma = r/B$ and $x_\gamma^* = (1-\gamma)x_t + \gamma x^*$, and let $x_{t+1} = x_t + \Delta\tilde{x}_t$ for $\|\Delta\tilde{x}_t\| \leq r$. Then*

$$F(x_{t+1}) - F(x_t) \leq Q_t^{\delta(r)}(\Delta\tilde{x}_t)$$
$$F(x_\gamma^*) - F(x_t) \geq Q_t^{1/\delta(r)}(x_\gamma^* - x_t).$$

Lemma 6 (Lemma 6 (Karimireddy et al., 2018)). *For any convex domain $\mathcal{Q}$ and constants $a \cdot b \geq 1$,*

$$\min_{\Delta x \in \mathcal{Q}} Q^a(\Delta x) \leq \frac{1}{ab}\min_{\Delta x \in \mathcal{Q}} Q^{1/b}(\Delta x).$$

We will now prove Lemma 2 from Section 3.

Lemma 2. *Let F satisfy Assumption 2 and be $\delta(r)$-locally stable for $\delta : \mathbb{R}^+ \mapsto \mathbb{R}^+$, let $x_0 \in \mathbb{R}^d$ be as input to* FEDSN *(Algorithm 1), let $c > 0$, let $\theta \in [0,1)$, and define $Q_t(\Delta x) := \frac{\delta(5c)}{2}\Delta x^\top \nabla^2 F(x_t)\Delta x + \nabla F(x_t)^\top \Delta x$, where x_t is the t^{th} iterate of Algorithm 1. Furthermore, suppose we are given that in each iteration of Algorithm 1, $\|\Delta\tilde{x}_t\| \leq 5c$ and*

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

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

$$\mathbb{E}F(x_T) - F^* \leq \mathbb{E}[F(x_0) - F^*] \exp\left(-\frac{Tc(1-\theta)}{2B\delta(\frac{1}{2}c)\delta(5c)}\right) + \frac{2B\delta(\frac{1}{2}c)d(5c)\epsilon}{c(1-\theta)} \,.$$

Proof. To begin, we are given by the assumptions of the theorem statement that Algorithm 1 chooses the update $\Delta\tilde{x}_t$ such that $\|\Delta\tilde{x}_t\| \leq 5c$ and

$$\mathbb{E}Q_t^{\delta(5c)}(\Delta\tilde{x}_t) - \min_{\|\Delta x\| \leq \frac{1}{2}c} Q_t^{\delta(5c)}(\Delta x) \leq \epsilon + \theta\left(Q_t^{\delta(5c)}(0) - \min_{\|\Delta x\| \leq \frac{1}{2}c} Q_t^{\delta(5c)}(\Delta x)\right). \tag{4}$$

Therefore,

$$\begin{aligned}
\mathbb{E}[F(x_{t+1}) - F(x_t)] &\leq \mathbb{E}Q_t^{\delta(5c)}(\Delta\tilde{x}_t) && (5)\\
&\leq \epsilon + (1-\theta) \min_{\Delta x \,:\, \|\Delta x\| \leq \frac{1}{2}c} Q_t^{\delta(5c)}(\Delta x) && (6)\\
&\leq \epsilon + \frac{1-\theta}{\delta(5c)\delta(\frac{1}{2}c)} \min_{\Delta x \,:\, \|\Delta x\| \leq \frac{1}{2}c} Q_t^{1/\delta(\frac{1}{2}c)}(\Delta x) && (7)\\
&\leq \epsilon + \frac{1-\theta}{\delta(5c)\delta(\frac{1}{2}c)} Q_t^{1/\delta(\frac{1}{2}c)}\left(\left(1 - \frac{c}{2B}\right)x_t + \frac{c}{2B}x^* - x_t\right) && (8)\\
&\leq \epsilon + \frac{1-\theta}{\delta(5c)\delta(\frac{1}{2}c)} \left[F\left(\left(1 - \frac{c}{2B}\right)x_t + \frac{c}{2B}x^*\right) - F(x_t)\right] && (9)\\
&\leq \epsilon + \frac{1-\theta}{\delta(5c)\delta(\frac{1}{2}c)} \left[\left(1 - \frac{c}{2B}\right)F(x_t) + \frac{c}{2B}F^* - F(x_t)\right] && (10)\\
&= \epsilon - \frac{c(1-\theta)}{2B\delta(5c)\delta(\frac{1}{2}c)}[F(x_t) - F^*]. && (11)
\end{aligned}$$

Here we used Lemma 5 for the first inequality, Lemma 6 for the third, Lemma 5 again for the fifth, and the convexity of F for the sixth.

Rearranging, and unravelling the recursion, we conclude that

$$\begin{aligned}
\mathbb{E}[F(x_T) - F^*] &\leq \left(1 - \frac{c(1-\theta)}{2B\delta(5c)\delta(\frac{1}{2}c)}\right)^T \mathbb{E}[F(x_0) - F^*] + \epsilon\sum_{t=0}^{T}\left(1 - \frac{c(1-\theta)}{2B\delta(5c)\delta(\frac{1}{2}c)}\right)^t && (12)\\
&\leq \mathbb{E}[F(x_0) - F^*] \exp\left(-\frac{Tc(1-\theta)}{2B\delta(5c)\delta(\frac{1}{2}c)}\right) + \frac{2B\delta(5c)\delta(\frac{1}{2}c)\epsilon}{c(1-\theta)} \,. && (13)
\end{aligned}$$

This completes the proof. □

C Proof of Lemma 3

Before we analyze Algorithm 2, we recall some key definitions. For a given $x \in \mathbb{R}^d$, we let

$$Q(u) = \frac{\bar{\xi}}{2}u^\top \nabla^2 F(x)u + \nabla F(x)^\top u, \tag{14}$$

and for a regularization penalty λ, we also define

$$Q_\lambda(u) = \frac{1}{2}u^\top(\bar{\xi}\nabla^2 F(x) + \lambda\mathbf{I})u + \nabla F(x)^\top u. \tag{15}$$

We use u_λ^* to denote the (unique) minimizer of $Q_\lambda(u)$, and we use $r^*(\lambda) = \|u_\lambda^*\|$ to denote the norm of the minimizer. For $0 \leq r \leq r^*(0)$, we also use λ_r to denote the value of λ such that $r^*(\lambda_r) = r$ (Lemma 7 below shows that λ_r is unique).

Lemma 7 (Lemmas 35 and 36 (Carmon et al., 2020)). *For any r, there exists a unique $\lambda_r \geq 0$ such that*

$$u_r^* = \arg\min_{u:\|u\| \leq r} Q(u) = \arg\min_{u \in \mathbb{R}^d} Q_{\lambda_r}(u) = -(\nabla^2 F(x) + \lambda_r\mathbf{I})^{-1}\nabla F(x).$$

Also, λ_r is decreasing in r, and if $\lambda_r \neq 0$ then $\|u_r^*\| = r$.

Lemma 8. *For $r \le r^*(0)$, $\lambda_r \in \left[0, \frac{\|b\|}{r}\right]$.*

Proof. By Lemma 7, $\lambda \ge 0$. If $\lambda > 0$, then since $\nabla^2 F(x) \succeq 0$ (by convexity of F), we have that

$$r = \|u_\lambda^*\| = \|(\nabla^2 F(x) + \lambda \mathbf{I})^{-1} \nabla F(x)\| \le \frac{\|\nabla F(x)\|}{\lambda}. \tag{16}$$

Rearranging completes the proof. □

Lemma 9. *For any $r \ge 0$ let $\lambda \ge \gamma \ge 0$ such that $r^*(\lambda) = r$ and $r^*(\gamma) = 2r$. Then*

$$\mathbb{E}\big[Q_\gamma(u) - Q_\gamma(u_\gamma^*)\big] \le \epsilon \implies \mathbb{E}[Q(u) - Q(u_\lambda^*)] \le \epsilon.$$

Furthermore, for any y and γ,

$$\mathbb{E}\big[Q_\gamma(u) - Q_\gamma(u_\gamma^*)\big] \le \epsilon \implies \mathbb{E}[Q(u) - Q(y)] \le \epsilon + \frac{\gamma\|y\|^2}{2}.$$

Proof. For any u,

$$\begin{aligned}\mathbb{E}[Q_\gamma(u) - Q(u_\lambda^*)] &= \mathbb{E}\big[Q_\gamma(u) - Q_\gamma(u_\gamma^*)\big] + Q_\gamma(u_\gamma^*) - Q_\gamma(u_\lambda^*) + \frac{\gamma}{2}\|u_\lambda^*\|^2 && (17)\\ &\le \epsilon + Q_\gamma(u_\gamma^*) - Q_\gamma(u_\lambda^*) + \frac{\gamma}{2}\|u_\lambda^*\|^2. && (18)\end{aligned}$$

By the γ-strong convexity of Q_γ, the fact that either $\gamma = 0$ or $\|u_\lambda^*\| = r$ and $\|u_\gamma^*\| = 2r$, and the reverse triangle inequality,

$$Q_\gamma(u_\gamma^*) - Q_\gamma(u_\lambda^*) + \frac{\gamma}{2}\|u_\lambda^*\|^2 \le -\frac{\gamma}{2}\big\|u_\gamma^* - u_\lambda^*\big\|^2 + \frac{\gamma r^2}{2} \le -\frac{\gamma(2r-r)^2}{2} + \frac{\gamma r^2}{2} = 0. \tag{19}$$

Combining (19) with (18) completes the first part of the proof. For the second part, we observe that

$$\mathbb{E}\Big[Q(u) + \frac{\gamma}{2}\|u\|^2 - Q(y)\Big] = \mathbb{E}\big[Q_\gamma(u) - Q_\gamma(u_\gamma^*)\big] + Q_\gamma(u_\gamma^*) - Q_\gamma(y) + \frac{\gamma}{2}\|y\|^2 \le \epsilon + \frac{\gamma\|y\|^2}{2}. \tag{20}$$

Above, we used that $Q_\lambda(u_\lambda^*) = \min_u Q_\lambda(u) \le Q_\lambda(y)$. □

Lemma 10. *Let $\lambda \ge \gamma \ge 0$. Then ,*

$$\frac{\lambda}{\gamma} \ge \frac{\|u_\gamma^*\|}{\|u_\lambda^*\|}.$$

Proof. By the definition of $u_\gamma^* = \arg\min_u Q_\gamma(u)$ and $u_\lambda^* = \arg\min_u Q_\lambda(u)$ and the γ- and λ-strong convexity of Q_γ and Q_λ, respectively,

$$\frac{\lambda - \gamma}{2}\left(\big\|u_\gamma^*\big\|^2 - \|u_\lambda^*\|^2\right) = Q_\gamma(u_\lambda^*) - Q_\gamma(u_\gamma^*) + Q_\lambda(u_\gamma^*) - Q_\lambda(u_\lambda^*) \ge \frac{\lambda+\gamma}{2}\big\|u_\lambda^* - u_\gamma^*\big\|^2. \tag{21}$$

Therefore, by rearranging and applying the reverse triangle inequality

$$\frac{\lambda - \gamma}{\lambda + \gamma} \ge \frac{\big\|u_\gamma^* - u_\lambda^*\big\|^2}{\big\|u_\gamma^*\big\|^2 - \|u_\lambda^*\|^2} \ge \frac{\left(\|u_\gamma^*\| - \|u_\lambda^*\|\right)^2}{\big\|u_\gamma^*\big\|^2 - \|u_\lambda^*\|^2} = \frac{\|u_\gamma^*\| - \|u_\lambda^*\|}{\|u_\gamma^*\| + \|u_\lambda^*\|}. \tag{22}$$

By Lemma 7, since $\lambda \ge \gamma$, $\big\|u_\gamma^*\big\| \ge \|u_\lambda^*\|$. Therefore, rearranging this inequality completes the proof. □

Lemma 11. *Let $X \sim$ Binomial(C, p) with $p \ge \frac{3}{4}$. Then,*

$$\mathbb{P}\left(X > \frac{C}{2}\right) \ge 1 - \exp\left(-\frac{C}{8}\right).$$

Proof. This is a simple application of Hoeffding's inequality:

$$\mathbb{P}\left(\frac{1}{C}X \le \frac{1}{2} \le p - \frac{1}{4}\right) \le \exp\left(-\frac{C}{8}\right). \tag{23}$$

□

Lemma 12. *Let $x \in \mathbb{R}^d$ be as input to Algorithm 2, let $Q_{\lambda^{(i)}}(u) = \frac{1}{2}u^\top\big(\delta(5\bar{r})\nabla^2 F(x) + \lambda^{(i)}\big)u + \nabla F(x)^\top u$, let $Q^*_{\lambda^{(i)}} := \min_u Q_{\lambda^{(i)}}(u)$, and for each i, let $\tilde{u}^{(i,1)}, \dots, \tilde{u}^{(i,C)}$ be computed as in Algorithm 2 such that*

$$\mathbb{E}Q_{\lambda^{(i)}}(\tilde{u}^{(i,c)}) - Q^*_{\lambda^{(i)}} \le \epsilon(\lambda^{(i)}) \le \frac{\lambda^{(i)}\big(r^*(\lambda^{(i)})^2 + \bar{r}^2\big)}{800}\,.$$

Then,

$$\begin{aligned}
r^*(\lambda^{(i)}) \in [2\bar{r},\, 3\bar{r}] &\implies \mathbb{P}\left(\sum_{c=1}^{C} \mathbb{1}\left\{\left\|\tilde{u}^{(i,c)}\right\| \in \left[\frac{3}{2}\bar{r},\, \frac{7}{2}\bar{r}\right]\right\} > \frac{C}{2}\right) \ge 1 - \exp\left(-\frac{C}{8}\right)\\
r^*(\lambda^{(i)}) \notin [\bar{r},\, 4\bar{r}] &\implies \mathbb{P}\left(\sum_{c=1}^{C} \mathbb{1}\left\{\left\|\tilde{u}^{(i,c)}\right\| \in \left[\frac{3}{2}\bar{r},\, \frac{7}{2}\bar{r}\right]\right\} \le \frac{C}{2}\right) \le \exp\left(-\frac{C}{8}\right)\\
r^*(\lambda^{(i)}) < 2\bar{r} &\implies \mathbb{P}\left(\sum_{c=1}^{C} \mathbb{1}\left\{\left\|\tilde{u}^{(i,c)}\right\| \le \frac{5}{2}\bar{r}\right\} > \frac{C}{2}\right) \ge 1 - \exp\left(-\frac{C}{8}\right)\\
r^*(\lambda^{(i)}) > 3\bar{r} &\implies \mathbb{P}\left(\sum_{c=1}^{C} \mathbb{1}\left\{\left\|\tilde{u}^{(i,c)}\right\| > \frac{5}{2}\bar{r}\right\} > \frac{C}{2}\right) \ge 1 - \exp\left(-\frac{C}{8}\right).
\end{aligned}$$

Proof. By the $\lambda^{(i)}$-strong convexity of $Q_{\lambda^{(i)}}$, for each c,

$$\mathbb{E}\left[\frac{\lambda^{(i)}}{2}\|\tilde{u}^{(i,c)} - u^*_{\lambda^{(i)}}\|^2\right] \le \mathbb{E}Q_{\lambda^{(i)}}(\tilde{u}^{(i,c)}) - Q^*_{\lambda^{(i)}} \le \epsilon(\lambda^{(i)}). \tag{24}$$

Therefore, by Markov's inequality, for each c

$$\mathbb{P}\left(\|\tilde{u}^{(i,c)} - u^*_{\lambda^{(i)}}\|^2 \ge \frac{1}{100}r^*(\lambda^{(i)})^2 + \frac{1}{100}\bar{r}^2\right) \le \frac{200\epsilon(\lambda^{(i)})}{\lambda^{(i)}\big(r^*(\lambda^{(i)})^2 + \bar{r}^2\big)} \le \frac{1}{4}\,. \tag{25}$$

Furthermore, by the reverse triangle inequality,

$$\Big|\|\tilde{u}^{(i,c)}\| - r^*(\lambda^{(i)})\Big| \le \|\tilde{u}^{(i,c)} - u^*_{\lambda^{(i)}}\| < \sqrt{\frac{1}{100}r^*(\lambda^{(i)})^2 + \frac{1}{100}\bar{r}^2} \le \frac{r^*(\lambda^{(i)}) + \bar{r}}{10}\,. \tag{26}$$

Therefore, for each c

$$\mathbb{P}\left(\frac{9}{10}r^*(\lambda^{(i)}) - \frac{1}{10}\bar{r} \le \|\tilde{u}^{(i,c)}\| \le \frac{11}{10}r^*(\lambda^{(i)}) + \frac{1}{10}\bar{r}\right) \ge \frac{3}{4}\,. \tag{27}$$

We now consider several cases:

If $r^*(\lambda^{(i)}) \in [2\bar{r},\, 3\bar{r}]$, then

$$\mathbb{P}\left(\|\tilde{u}^{(i,c)}\| \in \left[\frac{3}{2}\bar{r},\, \frac{7}{2}\bar{r}\right]\right) \tag{28}$$

$$\ge \mathbb{P}\left(\frac{17}{10}\bar{r} \le \frac{9}{10}r^*(\lambda^{(i)}) - \frac{1}{10}\bar{r} \le \|\tilde{u}^{(i,c)}\| \le \frac{11}{10}r^*(\lambda^{(i)}) + \frac{1}{10}\bar{r} \le \frac{34}{10}\bar{r}\right) \ge \frac{3}{4}. \tag{29}$$

Therefore, by Lemma 11

$$r^*(\lambda^{(i)}) \in [2\bar{r},\, 3\bar{r}] \implies \mathbb{P}\left(\sum_{c=1}^{C} \mathbb{1}\left\{\left\|\tilde{u}^{(i,c)}\right\| \in \left[\frac{3}{2}\bar{r},\, \frac{7}{2}\bar{r}\right]\right\} > \frac{C}{2}\right) \ge 1 - \exp\left(-\frac{C}{8}\right). \tag{30}$$

If $r^*(\lambda^{(i)}) > 4\bar{r}$, then

$$\mathbb{P}\left(\|\tilde{u}^{(i,c)}\| \notin \left[\frac{3}{2}\bar{r},\, \frac{7}{2}\bar{r}\right]\right) \ge \mathbb{P}\left(\frac{7}{2}\bar{r} \le \frac{9}{10}r^*(\lambda^{(i)}) - \frac{1}{10}\bar{r} \le \|\tilde{u}^{(i,c)}\|\right) \ge \frac{3}{4}\,. \tag{31}$$

Similarly, if $r^*(\lambda^{(i)}) < \bar{r}$, then

$$\mathbb{P}\bigg(\|\tilde{u}^{(i,c)}\| \notin \bigg[\frac{3}{2}\bar{r},\ \frac{7}{2}\bar{r}\bigg]\bigg) \geq \mathbb{P}\bigg(\|\tilde{u}^{(i,c)}\| \leq \frac{11}{10}r^*(\lambda^{(i)}) + \frac{1}{10}\bar{r} \leq \frac{12}{10}\bar{r}\bigg) \geq \frac{3}{4}\,. \tag{32}$$

Therefore, by Lemma 11,

$$r^*(\lambda^{(i)}) \notin [\bar{r},\, 4\bar{r}] \implies \mathbb{P}\Bigg(\sum_{c=1}^{C} \mathbb{1}\bigg\{\Big\|\tilde{u}^{(i,c)}\Big\| \in \bigg[\frac{3}{2}\bar{r},\ \frac{7}{2}\bar{r}\bigg]\bigg\} \leq \frac{C}{2}\Bigg) \leq \exp\bigg(-\frac{C}{8}\bigg)\,. \tag{33}$$

If $r^*(\lambda^{(i)}) < 2\bar{r}$, then

$$\mathbb{P}\bigg(\|\tilde{u}^{(i,c)}\| \leq \frac{5}{2}\bar{r}\bigg) \geq \mathbb{P}\bigg(\|\tilde{u}^{(i,c)}\| \leq \frac{11}{10}r^*(\lambda^{(i)}) + \frac{1}{10}\bar{r} \leq \frac{23}{10}\bar{r}\bigg) \geq \frac{3}{4}\,. \tag{34}$$

Therefore, by Lemma 11

$$r^*(\lambda^{(i)}) < 2\bar{r} \implies \mathbb{P}\Bigg(\sum_{c=1}^{C} \mathbb{1}\bigg\{\Big\|\tilde{u}^{(i,c)}\Big\| \leq \frac{5}{2}\bar{r}\bigg\} > \frac{C}{2}\Bigg) \geq 1 - \exp\bigg(-\frac{C}{8}\bigg)\,. \tag{35}$$

Finally, if $r^*(\lambda^{(i)}) > 3\bar{r}$, then

$$\mathbb{P}\bigg(\|\tilde{u}^{(i,c)}\| > \frac{5}{2}\bar{r}\bigg) \geq \mathbb{P}\bigg(\frac{26}{10}\bar{r} \leq \frac{9}{10}r^*(\lambda^{(i)}) - \frac{1}{10}\bar{r} \leq \|\tilde{u}^{(i,c)}\|\bigg) \geq \frac{3}{4}\,. \tag{36}$$

Therefore, by Lemma 11

$$r^*(\lambda^{(i)}) > 3\bar{r} \implies \mathbb{P}\Bigg(\sum_{c=1}^{C} \mathbb{1}\bigg\{\Big\|\tilde{u}^{(i,c)}\Big\| > \frac{5}{2}\bar{r}\bigg\} > \frac{C}{2}\Bigg) \geq 1 - \exp\bigg(-\frac{C}{8}\bigg)\,. \tag{37}$$

This completes the proof. □

Lemma 13. *Let $N \geq 1 + \frac{5}{2}\log\frac{\|\nabla F(x)\|}{3\bar{r}\bar{\lambda}}$. Then, either $\bar{\lambda} \geq \lambda_{3\bar{r}}$ or $\exists\lambda \in \Lambda_1$ with $r^*(\lambda) \in [2\bar{r}, 3\bar{r}]$.*

Proof. In the case that $\bar{\lambda} \leq \lambda_{2\bar{r}}$, by Lemma 8, with $N \geq 1 + \frac{5}{2}\log\frac{\|\nabla F(x)\|}{3\bar{r}\bar{\lambda}}$,

$$\bar{\lambda} \leq \lambda_{3\bar{r}} \leq \frac{\|\nabla F(x)\|}{3\bar{r}} \leq \bar{\lambda}c^{N-1}. \tag{38}$$

So, $\lambda_{3\bar{r}}$ is between the largest and smallest elements in Λ_1. It follows that for some $1 \leq n \leq N$,

$$\bar{\lambda}c^{n-1} \leq \lambda_{3\bar{r}} \leq \bar{\lambda}c^{n}. \tag{39}$$

Therefore, by Lemma 10,

$$\frac{3}{2} = \frac{\bar{\lambda}\big(\frac{3}{2}\big)^{n}}{\bar{\lambda}\big(\frac{3}{2}\big)^{n-1}} \geq \frac{\bar{\lambda}\big(\frac{3}{2}\big)^{n}}{\lambda_{3\bar{r}}} \geq \frac{3\bar{r}}{r^*\big(\bar{\lambda}\big(\frac{3}{2}\big)^{n}\big)} \implies r^*(\bar{\lambda}\bigg(\frac{3}{2}\bigg)^{n}) \geq 2\bar{r} \tag{40}$$

Finally, by Lemma 7, $r^*(\bar{\lambda}\big(\frac{3}{2}\big)^{n}) \leq 3\bar{r}$, so $r^*(\bar{\lambda}\big(\frac{3}{2}\big)^{n}) \in [2\bar{r}, 3\bar{r}]$ as claimed. □

Lemma 14. *Let λ satisfy $r^*(\lambda) \leq 4\bar{r}$ and let $\tilde{u}$ be chosen so that*

$$\mathbb{E}Q_\lambda(\tilde{u}) - Q^*_\lambda \leq \epsilon(\lambda) := \frac{\lambda(r^*(\lambda)^2 + \bar{r}^2)}{800}\,.$$

Then if $r^*(\lambda) \geq \bar{r}$,

$$\mathbb{E}Q\bigg(\min\bigg\{1,\ \frac{5\bar{r}}{\|\tilde{u}\|}\bigg\}\tilde{u}\bigg) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} \leq \frac{1}{2}\bigg(Q(0) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} Q(u)\bigg) + \epsilon(\lambda).$$

Otherwise,

$$\mathbb{E}Q\bigg(\min\bigg\{1,\ \frac{5\bar{r}}{\|\tilde{u}\|}\bigg\}\tilde{u}\bigg) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} \leq \frac{1}{2}\bigg(Q(0) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} Q(u)\bigg) + \epsilon(\lambda) + \frac{\lambda\bar{r}^2}{8}\,.$$

Proof. First,

$$\begin{aligned}
&\mathbb{E}Q\left(\min\left\{1,\ \frac{5\bar{r}}{\|\tilde{u}\|}\right\}\tilde{u}\right) - \min_{u:\|u\|\le\frac{1}{2}\bar{r}} Q(u) \\
&\quad= \mathbb{E}Q\left(\left(1-\min\left\{1,\ \frac{5\bar{r}}{\|\tilde{u}\|}\right\}\right)0 + \min\left\{1,\ \frac{5\bar{r}}{\|\tilde{u}\|}\right\}\tilde{u}\right) - \min_{u:\|u\|\le\frac{1}{2}\bar{r}} Q(u) && (41)\\
&\quad\le \mathbb{E}\left[\left(1-\min\left\{1,\ \frac{5\bar{r}}{\|\tilde{u}\|}\right\}\right)\left(Q(0) - \min_{u:\|u\|\le\frac{1}{2}\bar{r}} Q(u)\right)\right. && (42)\\
&\qquad\qquad \left. + \min\left\{1,\ \frac{5\bar{r}}{\|\tilde{u}\|}\right\}\left(Q(\tilde{u}) - \min_{u:\|u\|\le\frac{1}{2}\bar{r}} Q(u)\right)\right] && (43)\\
&\quad\le \mathbb{E}\left[\frac{\|\tilde{u}\|}{\|\tilde{u}\|+5\bar{r}}\right]\left(Q(0) - \min_{u:\|u\|\le\frac{1}{2}\bar{r}} Q(u)\right) + \mathbb{E}Q(\tilde{u}) - \min_{u:\|u\|\le\frac{1}{2}\bar{r}} Q(u) && (44)\\
&\quad\le \frac{\mathbb{E}\|\tilde{u}\|}{\mathbb{E}\|\tilde{u}\|+5\bar{r}}\left(Q(0) - \min_{u:\|u\|\le\frac{1}{2}\bar{r}} Q(u)\right) + \mathbb{E}Q(\tilde{u}) - \min_{u:\|u\|\le\frac{1}{2}\bar{r}} Q(u). && (45)
\end{aligned}$$

For the first inequality we used the convexity of Q, and for the final inequality we used Jensen's inequality on the concave function $x \mapsto \frac{x}{x+5\bar{r}}$. Next, we bound

$$\begin{aligned}
\mathbb{E}\|\tilde{u}\| &\le r^*(\lambda) + \mathbb{E}\|\tilde{u} - u^*_\lambda\| && (46)\\
&\le 4\bar{r} + \sqrt{\mathbb{E}\|\tilde{u} - u^*_\lambda\|^2} && (47)\\
&\le 4\bar{r} + \sqrt{\frac{2}{\lambda}\mathbb{E}Q_\lambda(\tilde{u}) - Q^*_\lambda} && (48)\\
&\le 4\bar{r} + \sqrt{\frac{2\epsilon(\lambda)}{\lambda}} && (49)\\
&\le 4\bar{r} + \sqrt{\frac{r^*(\lambda)^2 + \bar{r}^2}{400}} && (50)\\
&\le 4\bar{r} + \sqrt{\frac{17\bar{r}^2}{400}} && (51)\\
&< 5\bar{r}. && (52)
\end{aligned}$$

Therefore,

$$\mathbb{E}Q\left(\min\left\{1,\ \frac{5\bar{r}}{\|\tilde{u}\|}\right\}\tilde{u}\right) - \min_{u:\|u\|\le\frac{1}{2}\bar{r}} Q(u) \le \frac{1}{2}\left(Q(0) - \min_{u:\|u\|\le\frac{1}{2}\bar{r}} Q(u)\right) + \mathbb{E}Q(\tilde{u}) - \min_{u:\|u\|\le\frac{1}{2}\bar{r}} Q(u). \qquad (53)$$

If $r^*(\lambda) \ge \bar{r}$, then by the first part of Lemma 9

$$\mathbb{E}Q(\tilde{u}) - \min_{u:\|u\|\le\frac{1}{2}\bar{r}} Q(u) \le \mathbb{E}Q(\tilde{u}) - \min_{u:\|u\|\le\frac{1}{2}r^*(\lambda)} Q(u) \le \epsilon(\lambda). \qquad (54)$$

Otherwise, by the second part of Lemma 9

$$\mathbb{E}Q(\tilde{u}) - \min_{u:\|u\|\le\frac{1}{2}\bar{r}} Q(u) = \mathbb{E}Q(\tilde{u}) - Q(u^*_{\lambda_{\frac{1}{2}\bar{r}}}) \le \epsilon(\lambda) + \frac{\lambda\|u^*_{\lambda_{\frac{1}{2}\bar{r}}}\|^2}{2} = \epsilon(\lambda) + \frac{\lambda\bar{r}^2}{8}. \qquad (55)$$

□

We are now ready to prove Lemma 3.

Lemma 3. *Let F satisfy Assumption 2, let x be as input to* Constrained-Quadratic-Solver *(Algorithm 2), let $\bar{\xi}$ be as in Table 4, define*

$$Q(u) := \frac{\bar{\xi}}{2}u^\top \nabla^2 F(x)u + \nabla F(x)^\top u, \qquad Q_\lambda(u) := \frac{1}{2}u^\top(\bar{\xi}\nabla^2 F(x) + \lambda\mathbf{I})u + \nabla F(x)^\top u,$$

$$u_\lambda^* := \arg\min_u Q_\lambda(u), \qquad r^*(\lambda) := \|u_\lambda^*\|,$$

and let λ_r denote, for any $r > 0$, the value such that $r^*(\lambda_r) = r$. Let $\hat{u}$ be the output of Algorithm 2 for hyperparameters $\bar{r}$, $\underline{\xi}$, $\bar{\lambda}$, N and C as in Table 4, and suppose the output $\tilde{u}_\lambda$ of Regularized-Quadratic-Solver(x, λ) *satisfies for all $\lambda \geq \bar{\lambda}$ that*

$$\mathbb{E}Q_\lambda(\tilde{u}_\lambda) - \min_u Q_\lambda(u) \leq \epsilon(\lambda) := \frac{\lambda(r^*(\lambda)^2 + \bar{r}^2)}{800}\,.$$

Then $\|\hat{u}\| \leq 5\bar{r}$ and

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

Proof. First, we note that $|\Lambda_1| = N$, and in each iteration either the algorithm terminates or Λ_{i+1} is chosen such that $|\Lambda_{i+1}| \leq \frac{1}{2}|\Lambda_i|$. Therefore, the algorithm terminates after at most $\lceil \log_2 N \rceil$ iterations.

By Lemma 13, either $\bar{\lambda} \geq \lambda_{3\bar{r}}$ or there exists $\lambda \in \Lambda_1$ for some λ such that $r^*(\lambda) \in [2\bar{r}, 3\bar{r}]$. If such a λ exists, we denote this (not necessarily unique) value λ^*.

By Lemma 12 and the union bound, the following holds for all iteration $i = 1, 2, \dots$ with probability at least $1 - 2\lceil \log_2 N \rceil \exp\left(-\frac{C}{8}\right)$:

$$r^*(\lambda^{(i)}) \in [2\bar{r},\, 3\bar{r}] \implies \sum_{c=1}^{C} \mathbb{1}\left\{\left\|\tilde{u}^{(i,c)}\right\| \in \left[\frac{3}{2}\bar{r},\, \frac{7}{2}\bar{r}\right]\right\} > \frac{C}{2} \tag{56}$$

$$r^*(\lambda^{(i)}) \notin [\bar{r},\, 4\bar{r}] \implies \sum_{c=1}^{C} \mathbb{1}\left\{\left\|\tilde{u}^{(i,c)}\right\| \in \left[\frac{3}{2}\bar{r},\, \frac{7}{2}\bar{r}\right]\right\} \leq \frac{C}{2} \tag{57}$$

$$r^*(\lambda^{(i)}) < 2\bar{r} \implies \sum_{c=1}^{C} \mathbb{1}\left\{\left\|\tilde{u}^{(i,c)}\right\| \leq \frac{5}{2}\bar{r}\right\} > \frac{C}{2} \tag{58}$$

$$r^*(\lambda^{(i)}) > 3\bar{r} \implies \sum_{c=1}^{C} \mathbb{1}\left\{\left\|\tilde{u}^{(i,c)}\right\| > \frac{5}{2}\bar{r}\right\} > \frac{C}{2}\,. \tag{59}$$

For most of the rest of the proof, we condition on this event, which we denote E.

Under E, if $\lambda^{(i)} = \lambda^*$, then the algorithm will terminate on Line 2, and even if $\lambda^{(i)} \neq \lambda^*$, if the algorithm terminates on Line 2, then $\lambda^{(i)} \in [\bar{r}, 4\bar{r}]$. In either case, by the first part of Lemma 14

$$\mathbb{E}Q(\hat{u}) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} Q(u) \leq \frac{1}{2}\left(Q(0) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} Q(u)\right) + \epsilon(\lambda^{(i)}). \tag{60}$$

Finally, since $r^*(\lambda^{(i)}) \leq 4\bar{r}$, $\lambda^{(i)} \geq \lambda_{4\bar{r}}$.

If the algorithm instead updates

$$\Lambda_{i+1} = \Lambda_i \setminus \left\{\lambda \in \Lambda_i \,:\, \lambda \geq \lambda^{(i)}\right\} \tag{61}$$

as on Line 2, then conditioned on E,

$$\sum_{c=1}^{C} \mathbb{1}\left\{\left\|\tilde{u}^{(i,c)}\right\| \in \left[\frac{3}{2}\bar{r},\, \frac{7}{2}\bar{r}\right]\right\} \leq \frac{C}{2} < \sum_{c=1}^{C} \mathbb{1}\left\{\left\|\tilde{u}^{(i,c)}\right\| \leq \frac{5}{2}\bar{r}\right\} \tag{62}$$

implies $\lambda^{(i)} < 2\bar{r}$. By Lemma 7, since $r^*(\lambda^*) \geq 2\bar{r} > r^*(\lambda^{(i)})$, $\lambda^* < \lambda^{(i)}$ and therefore $\lambda^* \in \Lambda_{i+1}$.

If the algorithm instead updates

$$\Lambda_{i+1} = \Lambda_i \setminus \left\{\lambda \in \Lambda_i \,:\, \lambda \leq \lambda^{(i)}\right\} \tag{63}$$

as on Line 2, then conditioned on E

$$\sum_{c=1}^{C} \mathbb{1}\left\{\left\|\tilde{u}^{(i,c)}\right\| \in \left[\frac{3}{2}\bar{r}, \frac{7}{2}\bar{r}\right]\right\} \leq \frac{C}{2} < \sum_{c=1}^{C} \mathbb{1}\left\{\left\|\tilde{u}^{(i,c)}\right\| > \frac{5}{2}\bar{r}\right\} \tag{64}$$

implies $\lambda^{(i)} > 3\bar{r}$. By Lemma 7, since $r^*(\lambda^*) \leq 3\bar{r} < r^*(\lambda^{(i)})$, we have $\lambda^* > \lambda^{(i)}$, and therefore $\lambda^* \in \Lambda_{i+1}$.

Finally, E implies that the algorithm will never reach Line 2.

Therefore, conditioned on E, if $\lambda^* \in \Lambda_1$, then the algorithm will never remove λ^* from the set of λ's under consideration, and it will eventually terminate on Line 2 by returning a point such that

$$\mathbb{E}Q(\hat{u}) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} Q(u) \leq \frac{1}{2}\left(Q(0) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} Q(u)\right) + \epsilon(\lambda^{(i)}). \tag{65}$$

Otherwise, if such a λ^* does not exist and the algorithm does not terminate on Line 2, then it terminates on Line 2 using $\bar{\lambda} \geq \lambda_{3\bar{r}} \geq \lambda_{4\bar{r}}$, which implies $r^*(\bar{\lambda}) \leq 4\bar{r}$. Therefore, by the second part of Lemma 14,

$$\mathbb{E}Q(\tilde{u}) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} Q(u) \leq \frac{1}{2}\left(Q(0) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} Q(u)\right) + \epsilon(\bar{\lambda}) + \frac{\bar{\lambda}\bar{r}^2}{8}. \tag{66}$$

Therefore since $\epsilon(\lambda)$ is decreasing in λ, conditioned on E the algorithm's output satisfies

$$\mathbb{E}Q(\tilde{u}) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} Q(u) \leq \frac{1}{2}\left(Q(0) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} Q(u)\right) + \epsilon(\lambda_{4\bar{r}}) + \frac{\bar{\lambda}\bar{r}^2}{8}. \tag{67}$$

We now consider the case that E does not hold. In this case, the algorithm's output is guaranteed to have norm at most $5\bar{r}$. Therefore,

$$\begin{aligned} Q(\hat{u}) = \frac{\delta(5\bar{r})}{2}\hat{u}^\top \nabla^2 F(x)\hat{u} + \nabla F(x)^\top \hat{u} &\leq \frac{25\delta(5\bar{r})\|\nabla^2 F(x)\|_2 \bar{r}^2}{2} + 5\bar{r}\|\nabla F(x)\| &(68)\\ &= Q(0) + \frac{25\delta(5\bar{r})\|\nabla^2 F(x)\|_2 \bar{r}^2}{2} + 5\bar{r}\|\nabla F(x)\|. &(69)\end{aligned}$$

Therefore, conditioned on $\neg E$,

$$Q(\hat{u}) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} Q(u) \leq Q(0) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} Q(u) + \frac{25\delta(5\bar{r})\|\nabla^2 F(x)\|_2 \bar{r}^2}{2} + 5\bar{r}\|\nabla F(x)\|. \tag{70}$$

We conclude by noting that

$$\begin{aligned} &\mathbb{E}Q(\hat{u}) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} Q(u)\\ &= \mathbb{E}[Q(\hat{u}) \mid E]\mathbb{P}(E) + \mathbb{E}[Q(\hat{u}) \mid \neg E]\mathbb{P}(\neg E) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} Q(u) &(71)\\ &\leq \frac{1+\mathbb{P}(\neg E)}{2}\left(Q(0) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} Q(u)\right) + \epsilon(\lambda_{4\bar{r}}) + \frac{\bar{\lambda}\bar{r}^2}{8} &(72)\\ &\quad + \left(\frac{25\delta(5\bar{r})\|\nabla^2 F(x)\|_2 \bar{r}^2}{2} + 5\bar{r}\|\nabla F(x)\|\right)\mathbb{P}(\neg E) &(73)\\ &\leq \frac{1 + 2\lceil\log_2 N\rceil \exp(-\frac{C}{8})}{2}\left(Q(0) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} Q(u)\right) + \epsilon(\lambda_{4\bar{r}}) + \frac{\bar{\lambda}\bar{r}^2}{8}\\ &\quad + \left(\frac{25\delta(5\bar{r})\|\nabla^2 F(x)\|_2 \bar{r}^2}{2} + 5\bar{r}\|\nabla F(x)\|\right)\cdot 2\lceil\log_2 N\rceil \exp\left(-\frac{C}{8}\right) &(74)\end{aligned}$$

Using the fact that

$$C \geq 8\log\left(\lceil\log_2 N\rceil\left(4 + \frac{8\delta(5\bar{r})\|\nabla^2 F(x)\|_2}{\bar{\lambda}} + \frac{80\|\nabla F(x)\|}{\bar{\lambda}\bar{r}}\right)\right) \tag{75}$$

we conclude

$$\mathbb{E}Q(\hat{u}) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} Q(u) \leq \frac{3}{4}\left(Q(0) - \min_{u:\|u\|\leq\frac{1}{2}\bar{r}} Q(u)\right) + \epsilon(\lambda_{4\bar{r}}) + \frac{\bar{\lambda}\bar{r}^2}{4}\ . \tag{76}$$

This completes the proof. $\square$

D Proof of Lemma 4

Lemma 4. *Let F satisfy Assumption 2, let $x \in \mathbb{R}^d$, $\lambda \in \mathbb{R}^+$ be as input to* Regularized-Quadratic-Solver *(Algorithm 3), let $Q_\lambda(u) = \frac{1}{2}u^\top(\bar{\xi}\nabla^2 F(x) + \lambda\mathbf{I})u + \nabla F(x)^\top u$, let $Q_\lambda^* := \min_u Q_\lambda(u)$, let $u^* := \arg\min_u Q_\lambda(u)$, and let stochastic first-order and stochastic Hessian-vector product oracles for F, as defined in Assumptions 1 and 2, respectively, be available for each call to* Regularized-Quadratic-Gradient-Access *(Algorithm 4), for either Case 1* (`Different-Samples`) *or Case 2* (`Same-Sample`)*. Let $\hat{u}$, as output by Algorithm 3, be a weighted average of the iterates of M independent runs of SGD with stepsizes $\eta_0(\lambda), \ldots, \eta_{K-1}(\lambda)$, i.e., $\hat{u} = \frac{1}{M\sum_{k=0}^{K-1} w_k}\sum_{m=1}^M \sum_{k=0}^{K-1} w_k u_k^m$. Then, for both Cases 1 and 2,*

$$\mathbb{E}Q_\lambda(\hat{u}) - Q_\lambda^* \leq \begin{cases} 2\max\left\{\bar{\xi}H + \lambda,\ \frac{\rho^2}{\lambda}\right\}\|u^*\|^2 \min\left\{\frac{1}{K},\ \exp\left(-\frac{K+1}{4}\min\left\{\frac{\lambda}{\bar{\xi}H+\lambda},\ \frac{\lambda^2}{\rho^2}\right\}\right)\right\} \\ \qquad + \frac{2(\sigma^2+\rho^2\|u^*\|^2)}{\lambda M K} \quad \text{if } K \leq \frac{2}{\lambda}\max\left\{\bar{\xi}H + \lambda,\ \frac{\rho^2}{\lambda}\right\} \\[2ex] 96\lambda\|u^*\|^2 \exp\left(-\frac{K}{8}\min\left\{\frac{\lambda}{\bar{\xi}H+\lambda},\ \frac{\lambda^2}{\rho^2}\right\}\right) + \frac{96(\sigma^2+\rho^2\|u^*\|^2)}{\lambda M K} \\ \qquad \text{if } K > \frac{2}{\lambda}\max\left\{\bar{\xi}H + \lambda,\ \frac{\rho^2}{\lambda}\right\}. \end{cases}$$

Proof. We will use $\bar{u}_k = \frac{1}{M}\sum_{m=1}^M u_k^m$ to denote the average of each independent run of SGD's k^{th} iterate. This quantity is never explicitly computed until the end, but we can nevertheless use it for our analysis. Likewise, we will use $\bar{\gamma}_k = \frac{1}{M}\sum_{m=1}^M \gamma(u_k^m; z_k^m, z_k^{\prime m})$ to denote the average of the stochastic gradients of $Q_\lambda(u)$ computed at time k, whereby we recall that $\gamma(u_k^m; z_k^m, z_k^{\prime m}) := \bar{\xi}h(x, u_k^m; z_k^{\prime m}) + \lambda u_k^m + g(x; z_k^m)$ as defined in Algorithm 3, along with the requisite oracle access as described in Algorithm 4.

We also have that, by Assumptions 1(c) and 2(b), for Case 1:

$$\mathbb{E}_{z_k^m, z_k^{\prime m}}[\gamma(u_k^m; z_k^m, z_k^{\prime m})] = \nabla Q_\lambda(u) \text{ and } \mathbb{E}_{z_k^m, z_k^{\prime m}}\left\|\gamma(u_k^m; z_k^m, z_k^{\prime m}) - \nabla Q_\lambda(u_k^m)\right\|^2 \leq \sigma^2 + \rho^2\|u_k^m\|^2,$$

while for Case 2 we have:

$$\mathbb{E}_{z_k^m}[\gamma(u_k^m; z_k^m, z_k^m)] = \nabla Q_\lambda(u) \text{ and } \mathbb{E}_{z_k^m}\|\gamma(u_k^m; z_k^m, z_k^m) - \nabla Q_\lambda(u_k^m)\|^2 \leq 2\sigma^2 + 2\rho^2\|u_k^m\|^2,$$

where we used the fact that $\|a+b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$ for any $a, b \in \mathbb{R}^d$, and so in either case the variance term is bounded by $2\sigma^2 + 2\rho^2\|u_k^m\|^2$.

A key feature of these stochastic gradients of $Q_\lambda(u)$, which we will use frequently, is that by the linearity of $\nabla Q_\lambda(u) = (\bar{\xi}\nabla^2 F(x) + \lambda\mathbf{I})u + \nabla F(x)$,

$$\mathbb{E}\bar{\gamma}_k = \frac{1}{M}\sum_{m=1}^M \mathbb{E}\gamma(u_k^m; z_k^m, z_k^{\prime m}) = \frac{1}{M}\sum_{m=1}^M \mathbb{E}\nabla Q_\lambda(u_k^m) = \nabla Q_\lambda(\mathbb{E}\bar{u}_k). \tag{77}$$

We begin by expanding,

$$\begin{aligned}
&\mathbb{E}\|\bar{u}_{k+1} - u^*\|^2 \\
&\quad = \mathbb{E}\|\bar{u}_k - u^*\|^2 + \eta_k^2 \mathbb{E}\|\bar{\gamma}_k\|^2 - 2\eta_k \mathbb{E}\left\langle \nabla Q_\lambda(\bar{u}_k),\, \bar{u}_k - u^* \right\rangle && (78)\\
&\quad = \mathbb{E}\|\bar{u}_k - u^*\|^2 + \eta_k^2 \mathbb{E}\|\nabla Q_\lambda(\bar{u}_k)\|^2 - 2\eta_k \left[Q_\lambda(\bar{u}_k) - Q_\lambda^* + \frac{\lambda}{2}\|\bar{u}_k - u^*\|^2 \right] && (79)\\
&\qquad\quad + \eta_k^2 \mathbb{E}\|\bar{\gamma}_k - \nabla Q_\lambda(\bar{u}_k)\|^2 && (80)\\
&\quad \le (1 - \eta_k \lambda)\mathbb{E}\|\bar{u}_k - u^*\|^2 - 2\eta_k(1 - (\bar{\xi}H + \lambda)\eta_k)\mathbb{E}[Q_\lambda(\bar{u}_k) - Q_\lambda^*] && (81)\\
&\qquad\quad + \eta_k^2 \left(\frac{2\sigma^2}{M} + \frac{2\rho^2}{M^2} \sum_{m=1}^M \mathbb{E}\|u_k^m\|^2 \right). && (82)
\end{aligned}$$

Since $\eta_k \le \frac{1}{2(\bar{\xi}H + \lambda)}$ for all k, $(1 - (\bar{\xi}H + \lambda)\eta_k) \ge \frac{1}{2}$ so we can rearrange

$$\begin{aligned}
\mathbb{E}[Q_\lambda(\bar{u}_k) - Q_\lambda^*] &\le \left(\frac{1}{\eta_k} - \lambda \right) \mathbb{E}\|\bar{u}_k - u^*\|^2 - \frac{1}{\eta_k}\mathbb{E}\|\bar{u}_{k+1} - u^*\|^2 && (83)\\
&\qquad + \eta_k \left(\frac{2\sigma^2}{M} + \frac{2\rho^2}{M^2} \sum_{m=1}^M \mathbb{E}\|u_k^m\|^2 \right). && (84)
\end{aligned}$$

From here, we note that since $u_k^1, \ldots, u_k^M$ are i.i.d.,

$$\begin{aligned}
\frac{1}{M^2} \sum_{m=1}^M \mathbb{E}\|u_k^m\|^2 &= \frac{1}{M^2} \sum_{m=1}^M \left[\mathbb{E}\|u_k^m - \mathbb{E}u_k^m\|^2 + \|\mathbb{E}u_k^m\|^2 \right] && (85)\\
&= \mathbb{E}\|\bar{u}_k^m - \mathbb{E}\bar{u}_k^m\|^2 + \frac{1}{M}\|\mathbb{E}\bar{u}_k\|^2 && (86)\\
&\le \mathbb{E}\|\bar{u}_k - u^*\|^2 + \frac{1}{M}\|\mathbb{E}\bar{u}_k\|^2 && (87)\\
&\le \mathbb{E}\|\bar{u}_k - u^*\|^2 + \frac{2}{M}\|\mathbb{E}\bar{u}_k - u^*\|^2 + \frac{2}{M}\|u^*\|^2. && (88)
\end{aligned}$$

Furthermore, for each k, since $\eta_{k-1} \le \frac{1}{\bar{\xi}H + \lambda}$

$$\|\mathbb{E}\bar{u}_k - u^*\|^2 \le \|\mathbb{E}\bar{u}_{k-1} - u^*\|^2 + \eta_{k-1}^2 \|\nabla Q_\lambda(\mathbb{E}\bar{u}_{k-1})\|^2 - 2\eta_{k-1} \left\langle \nabla Q_\lambda(\mathbb{E}\bar{u}_{k-1}),\, \bar{u}_{k-1} - u^* \right\rangle \quad (89)$$

$$\begin{aligned}
&\le \|\mathbb{E}\bar{u}_{k-1} - u^*\|^2 + 2(\bar{\xi}H + \lambda)\eta_{k-1}^2 [Q_\lambda(\mathbb{E}\bar{u}_{k-1}) - Q_\lambda^*] && (90)\\
&\qquad - 2\eta_{k-1}[Q_\lambda(\mathbb{E}\bar{u}_{k-1}) - Q_\lambda^*] && (91)\\
&\le \|\mathbb{E}\bar{u}_{k-1} - u^*\|^2 \le \|\mathbb{E}\bar{u}_0 - u^*\|^2 = \|u^*\|^2. && (92)
\end{aligned}$$

Therefore, returning to (84),

$$\begin{aligned}
\mathbb{E}[Q_\lambda(\bar{u}_k) - Q_\lambda^*] &\le \left(\frac{1}{\eta_k} - \lambda \right) \mathbb{E}\|\bar{u}_k - u^*\|^2 - \frac{1}{\eta_k}\mathbb{E}\|\bar{u}_{k+1} - u^*\|^2 + \frac{2\eta_k \sigma^2}{M} && (93)\\
&\qquad + 2\eta_k \rho^2 \left(\mathbb{E}\|\bar{u}_k - u^*\|^2 + \frac{4}{M}\|u^*\|^2 \right) && (94)\\
&= \left(\frac{1}{\eta_k} - \lambda + \eta_k \rho^2 \right) \mathbb{E}\|\bar{u}_k - u^*\|^2 - \frac{1}{\eta_k}\mathbb{E}\|\bar{u}_{k+1} - u^*\|^2 && (95)\\
&\qquad + \frac{2\eta_k \left(\sigma^2 + \rho^2 \|u^*\|^2 \right)}{M}. && (96)
\end{aligned}$$

We first consider the case $K \le \frac{2\max\left\{\bar{\xi}H + \lambda, \frac{\rho^2}{\lambda}\right\}}{\lambda}$, so that

$$\eta_k = \eta = \min\left\{ \frac{1}{2(\bar{\xi}H + \lambda)},\, \frac{\lambda}{2\rho^2} \right\} \quad \text{and} \quad w_k = (1 - \lambda\eta + \eta^2\rho^2)^{-k-1}.$$

Then,

$$
\begin{aligned}
&\mathbb{E}Q_\lambda\left(\frac{1}{\sum_{k=0}^K w_k}\sum_{k=0}^K w_k\bar{u}_k\right) - Q_\lambda^* \\
&\le \frac{1}{\eta\sum_{k=0}^K w_k}\sum_{k=0}^K\Big[w_k\big(1-\eta\lambda+\eta^2\rho^2\big)\mathbb{E}\|\bar{u}_k-u^*\|^2 - w_k\mathbb{E}\|\bar{u}_{k+1}-u^*\|^2\Big] && (97)\\
&\qquad + \frac{2\eta\Big(\sigma^2+\rho^2\|u^*\|^2\Big)}{M} && (98)\\
&= \frac{1}{\eta\sum_{k=0}^K w_k}\sum_{k=0}^K\Big[\big(1-\eta\lambda+\eta^2\rho^2\big)^{-k}\mathbb{E}\|\bar{u}_k-u^*\|^2 - \big(1-\eta\lambda+\eta^2\rho^2\big)^{-(k+1)}\mathbb{E}\|\bar{u}_{k+1}-u^*\|^2\Big] && (99)\\
&\qquad + \frac{2\eta\Big(\sigma^2+\rho^2\|u^*\|^2\Big)}{M} && (100)\\
&\le \frac{\mathbb{E}\|\bar{u}_0-u^*\|^2}{\eta\sum_{k=0}^K(1-\eta\lambda+\eta^2\rho^2)^{-(k+1)}} + \frac{2\eta\Big(\sigma^2+\rho^2\|u^*\|^2\Big)}{M} && (101)\\
&\le \frac{\|u^*\|^2}{\eta\max\Big\{K,\,(1-\eta\lambda+\eta^2\rho^2)^{-(K+1)}\Big\}} + \frac{2\eta\Big(\sigma^2+\rho^2\|u^*\|^2\Big)}{M} && (102)\\
&\le 2\max\Big\{\bar{\xi}H+\lambda,\,\frac{\rho^2}{\lambda}\Big\}\|u^*\|^2\min\Bigg\{\frac{1}{K},\,\exp\Bigg(-\frac{K+1}{4}\min\Big\{\frac{\lambda}{\bar{\xi}H+\lambda},\,\frac{\lambda^2}{\rho^2}\Big\}\Bigg)^{K+1}\Bigg\}\\
&\qquad\quad + \frac{2(\sigma^2+\rho^2\|u^*\|^2)}{\lambda MK}\,. && (103)
\end{aligned}
$$

For the last line, we used that $K \le \frac{2\max\left\{\bar{\xi}H+\lambda,\frac{\rho^2}{\lambda}\right\}}{\lambda} = \frac{1}{\eta\lambda}$.

In the second case $\left(K > \frac{2\max\left\{\bar{\xi}H+\lambda,\frac{\rho^2}{\lambda}\right\}}{\lambda}\right)$, we consider the first $K/2$ iterations where

$$\eta_k = \eta = \min\left\{\frac{1}{2(\bar{\xi}H+\lambda)},\frac{\lambda}{2\rho^2}\right\}:$$

$$\mathbb{E}Q_\lambda(\bar{u}_{K/2-1}) - Q_\lambda^* \leq \left(\frac{1}{\eta} - \lambda + \eta\rho^2\right)\mathbb{E}\left\|\bar{u}_{K/2-1} - u^*\right\|^2 - \frac{1}{\eta}\mathbb{E}\left\|\bar{u}_{K/2} - u^*\right\|^2 \tag{104}$$

$$+ \frac{2\eta\left(\sigma^2 + \rho^2\|u^*\|^2\right)}{M} \tag{105}$$

$$\implies \mathbb{E}\left\|\bar{u}_{K/2} - u^*\right\|^2 \leq \left(1 - \eta\lambda + \eta^2\rho^2\right)\mathbb{E}\left\|\bar{u}_{K/2-1} - u^*\right\|^2 + \frac{2\eta^2\left(\sigma^2 + \rho^2\|u^*\|^2\right)}{M} \tag{106}$$

$$= \left(1 - \eta\lambda + \eta^2\rho^2\right)^{K/2}\mathbb{E}\|\bar{u}_0 - u^*\|^2 \tag{107}$$

$$+ \frac{2\eta^2\left(\sigma^2 + \rho^2\|u^*\|^2\right)}{M}\sum_{k=0}^{K/2-1}\left(1 - \eta\lambda + \eta^2\rho^2\right)^k \tag{108}$$

$$= \|u^*\|^2\left(1 - \eta\lambda + \eta^2\rho^2\right)^{K/2} \tag{109}$$

$$+ \frac{2\eta^2\left(\sigma^2 + \rho^2\|u^*\|^2\right)}{M}\frac{1 - \left(1 - \eta\lambda + \eta^2\rho^2\right)^{K/2}}{\eta\lambda - \eta^2\rho^2} \tag{110}$$

$$\leq \|u^*\|^2\left(1 - \eta\lambda + \eta^2\rho^2\right)^{K/2} + \frac{2\eta\left(\sigma^2 + \rho^2\|u^*\|^2\right)}{M(\lambda - \eta\rho^2)} \tag{111}$$

$$\leq \|u^*\|^2\left(1 - \frac{1}{2}\eta\lambda\right)^{K/2} + \frac{4\eta\left(\sigma^2 + \rho^2\|u^*\|^2\right)}{\lambda M}. \tag{112}$$

This bounds the distance of the $(K/2)^{\text{th}}$ iterate to the optimum, from which we can upper bound the suboptimality of the averaged iterate. Let $W = \sum_{k=0}^{K} w_k = \sum_{k=K/2}^{K} w_k$. Then by the convexity of $Q_\lambda(u)$,

$$\mathbb{E}Q_\lambda\left(\frac{1}{W}\sum_{k=0}^{K} w_k\bar{u}_k\right) - Q_\lambda^*$$

$$\leq \frac{1}{W}\sum_{k=0}^{K}\left[w_k\left(\frac{1}{\eta_k} - \lambda + \eta_k\rho^2\right)\mathbb{E}\|\bar{u}_k - u^*\|^2 - \frac{w_k}{\eta_k}\mathbb{E}\|\bar{u}_{k+1} - u^*\|^2 + \frac{2w_k\eta_k\left(\sigma^2 + \rho^2\|u^*\|^2\right)}{M}\right] \tag{113}$$

$$\leq \frac{w_{K/2}\left(\frac{1}{\eta_{K/2}} - \lambda + \eta_{K/2}\rho^2\right)\mathbb{E}\left\|\bar{u}_{K/2} - u^*\right\|^2}{W}$$

$$+ \frac{1}{W}\sum_{k=K/2+1}^{K}\left[\left(\frac{w_k}{\eta_k} - w_k\lambda + w_k\eta_k\rho^2 - \frac{w_{k-1}}{\eta_{k-1}}\right)\mathbb{E}\|\bar{u}_k - u^*\|^2 + \frac{2w_k\eta_k\left(\sigma^2 + \rho^2\|u^*\|^2\right)}{M}\right]. \tag{114}$$

With our setting of $w_k = (a+k-K/2-1)$ and $\eta_k = \frac{4}{\lambda(a+k-K/2)}$ with $a = \frac{8}{\lambda}\max\left\{\frac{\rho^2}{\lambda},\ \bar{\xi}H+\lambda\right\}$, we have for $k > K/2$,

$$\frac{\eta_{k-1}}{\eta_k} - \eta_{k-1}\lambda + \eta_{k-1}\eta_k\rho^2 = \frac{a+k-K/2}{a+k-K/2-1} - \frac{4}{a+k-K/2-1} \tag{115}$$

$$+ \frac{16\rho^2}{\lambda^2(a+k-K/2)(a+k-K/2-1)} \tag{116}$$

$$= \frac{a+k-K/2}{a+k-K/2-1} \tag{117}$$

$$+ \frac{1}{a+k-K/2-1}\left(2-4+\frac{16\rho^2}{\lambda^2(a+k-K/2)}\right) \tag{118}$$

$$\leq \frac{w_{k-1}}{w_k} + \frac{1}{a+k-K/2-1}\left(-2+\frac{16\rho^2}{\lambda^2 a}\right) \tag{119}$$

$$\leq \frac{w_{k-1}}{w_k}\,. \tag{120}$$

Therefore, we have

$$\mathbb{E}Q_\lambda(\hat{x}) - Q_\lambda^*$$

$$\leq \frac{w_{K/2}\left(\frac{1}{\eta_{K/2}} - \lambda + \eta_{K/2}\rho^2\right)\mathbb{E}\left\|\bar{u}_{K/2}-u^*\right\|^2}{W} + \frac{2}{W}\sum_{k=K/2+1}^{K}\frac{w_k\eta_k\left(\sigma^2+\rho^2\|u^*\|^2\right)}{M} \tag{121}$$

$$= \frac{(a-1)\left(\frac{\lambda a}{4} - \lambda + \frac{4\rho^2}{\lambda a}\right)\mathbb{E}\left\|\bar{u}_{K/2}-u^*\right\|^2}{W} \tag{122}$$

$$+ \frac{2(\sigma^2+\rho^2\|u^*\|^2)}{WM}\sum_{k=K/2+1}^{K}\frac{4(a+k-K/2-1)}{\lambda(a+k-K/2)} \tag{123}$$

$$\leq \frac{\lambda a^2\mathbb{E}\left\|\bar{u}_{K/2}-u^*\right\|^2}{4W} + \frac{8K(\sigma^2+\rho^2\|u^*\|^2)}{\lambda WM} \tag{124}$$

$$\leq \frac{\lambda a^2}{4W}\left(\|u^*\|^2\left(1-\frac{1}{2}\eta\lambda\right)^{K/2} + \frac{4\eta\left(\sigma^2+\rho^2\|u^*\|^2\right)}{\lambda M}\right) + \frac{8K(\sigma^2+\rho^2\|u^*\|^2)}{\lambda WM} \tag{125}$$

$$= \frac{\lambda a^2\|u^*\|^2}{4W}\left(1-\frac{1}{2}\eta\lambda\right)^{K/2} + \frac{(\sigma^2+\rho^2\|u^*\|^2)}{\lambda WM}\left(8K+\eta\lambda a^2\right) \tag{126}$$

$$= \frac{\lambda a^2\|u^*\|^2}{4W}\left(1-\min\left\{\frac{\lambda}{4(\bar{\xi}H+\lambda)},\ \frac{\lambda^2}{4\rho^2}\right\}\right)^{K/2} + \frac{(8K+4a)(\sigma^2+\rho^2\|u^*\|^2)}{\lambda WM}\,, \tag{127}$$

where, for the third-to-last line we used (112), and the last line we used that $\eta = \min\left\{\frac{1}{2(\bar{\xi}H+\lambda)},\ \frac{\lambda}{2\rho^2}\right\} = \frac{4}{\lambda a}$. Finally, we lower bound

$$W = \sum_{k=K/2}^{K}(a+k-K/2) = \sum_{i=0}^{K/2} a+i \geq \frac{1}{2}aK + \frac{1}{8}K^2. \tag{128}$$

Thus, since $K \geq \frac{2}{\lambda}\max\left\{\frac{\rho^2}{\lambda},\ \bar{\xi}H+\lambda\right\} = \frac{a}{4}$, we conclude

$$\mathbb{E}Q_\lambda(\hat{x}) - Q_\lambda^* \leq 96\lambda\|u^*\|^2\left(1-\min\left\{\frac{\lambda}{4(\bar{\xi}H+\lambda)},\ \frac{\lambda^2}{4\rho^2}\right\}\right)^{K/2} + \frac{96(\sigma^2+\rho^2\|u^*\|^2)}{\lambda MK}\,. \tag{129}$$

Combining this and (103) completes the proof. □

E Proof of Theorem 1

Theorem 1. *Let F satisfy Assumption 2. Then, for $K \geq 175$ and $R \geq \tilde{\Omega}(1)$, and for hyperparameters T, β, $\bar{r}$, $\bar{\xi}$, $\bar{\lambda}$, N, C and parameter functions $\eta_k(\lambda)$, $w_k(\lambda)$ as in Tables 3 and 4, the output of* FedSN *(Algorithm 1) with initial point $x_0 \in \mathbb{R}^d$, using Algorithms 2, 3, and 4 (for both Cases 1 and 2) satisfies*

$$\mathbb{E}[F(x_T)] - F^* \leq HB^2\left(\exp\left(-\frac{R}{\tilde{O}(\alpha B)}\right) + \exp\left(-\frac{K}{O(1)}\right)\right) + \tilde{O}\left(\frac{\sigma B}{\sqrt{MK}} + \frac{HB^2}{KR} + \frac{\rho B^2}{\sqrt{KR}}\right),$$

where $\tilde{\Omega}$, $\tilde{O}$ hide terms logarithmic in R, K, and αB.

Proof. We first recall the hyperparameters from Table 4, whose settings we will refer to throughout the course of the proof:

Hyperparameter Setting	Description
$T := \left\lfloor \frac{R}{4\zeta} \log^2\left(\left(\frac{R}{\zeta}\right)\right) \right\rfloor$ (for $\zeta = 4096 + 4(80 + 32\log K + 24\log(1 + 2\alpha B))^2$)	Main iterations
$\beta := 0$	Momentum
$\bar{r} := \min\{\frac{32B}{T}\log(TK), \frac{1}{5\alpha}\}$	Trust-region radius
$\bar{\xi} := \exp(\alpha \bar{r})$	Local stability
$\bar{\lambda} := \max\left\{\frac{2eH}{K-2}, \frac{2\rho}{\sqrt{K}}, \frac{32eH\log(51200)}{K}, \frac{4\rho\sqrt{2\log(51200)}}{\sqrt{K}}, \frac{320\sqrt{2}\rho}{\sqrt{MK}}, \frac{320\sigma}{\bar{r}\sqrt{MK}}, \frac{8eH}{K-16}\right\}$	Regularization bound
$N := \left\lceil 1 + \frac{5}{2}\log\frac{H(B+5T\bar{r})}{3\bar{\lambda}\bar{r}} \right\rceil$	Binary search iterations
$C := \left\lceil 8\log\left(\lceil \log_2 N \rceil \left(4 + \frac{eH}{\bar{\lambda}} + \frac{80H(B+5T\bar{r})}{\bar{\lambda}\bar{r}}\right)\right) \right\rceil$	Reg. quadratic repetitions

Table 4: Hyperparameters T, β, $\bar{r}$, $\bar{\xi}$, $\bar{\lambda}$, N, and C, as used by FedSN and its subroutines.

We also note that since all of the updates $\Delta\tilde{x}_t$ have norm at most $5\bar{r}$, $\|x_t - x^*\| \leq B + 5T\bar{r}$ for all t, and therefore by the H-smoothness of F, $\|\nabla F(x_t)\| \leq H(B + 5T\bar{r})$ for all t. Furthermore, since F is H-smooth and $\bar{r} \leq \frac{1}{5\alpha}$, $\bar{\xi}\nabla^2 F(x_t) \preceq eH\mathbf{I}$ for all t. Therefore, our settings of N and C satisfy the conditions of Lemma 3, and for each t,

$$\mathbb{E}Q_t(\Delta\tilde{x}_t) - \min_{\Delta x:\|\Delta x\| \leq \frac{1}{2}\bar{r}} Q_t(\Delta x) \leq \frac{3}{4}\left(Q_t(0) - \min_{\Delta x:\|\Delta x\| \leq \frac{1}{2}\bar{r}} Q_t(\Delta x)\right) + \epsilon(\lambda_{4\bar{r}}) + \frac{\bar{\lambda}\bar{r}^2}{4} \quad (130)$$

as long as the error guarantee of Algorithm 3 satisfies for all $\lambda \geq \bar{\lambda}$

$$\mathbb{E}Q_\lambda(\hat{u}) - \min_u Q_\lambda(u) \leq \epsilon(\lambda) = \frac{\lambda(r^*(\lambda)^2 + \bar{r}^2)}{800}. \quad (131)$$

By Lemma 4, since the objectives are such that $\left\|\bar{\xi}\nabla^2 F(x_t) + \lambda \mathbf{I}\right\|_2 \leq eH + \lambda$ for all t and λ, the output with optimally chosen stepsizes have error at most

$$\mathbb{E}Q_\lambda(\hat{u}) - \min_u Q_\lambda(u) \leq \tilde{\epsilon}(\lambda) \tag{132}$$

$$:= \begin{cases} 2\max\left\{eH+\lambda, \frac{\rho^2}{\lambda}\right\} r^*(\lambda)^2 \exp\left(-\frac{K+1}{4}\min\left\{\frac{\lambda}{eH+\lambda}, \frac{\lambda^2}{\rho^2}\right\}\right) + \frac{2(\sigma^2+\rho^2 r^*(\lambda)^2)}{\lambda MK} & K \leq \frac{2}{\lambda}\max\left\{eH+\lambda, \frac{\rho^2}{\lambda}\right\} \\ 96\lambda r^*(\lambda)^2 \exp\left(-\frac{K}{8}\min\left\{\frac{\lambda}{eH+\lambda}, \frac{\lambda^2}{\rho^2}\right\}\right) + \frac{96(\sigma^2+\rho^2 r^*(\lambda)^2)}{\lambda MK} & K > \frac{2}{\lambda}\max\left\{eH+\lambda, \frac{\rho^2}{\lambda}\right\} \end{cases}. \tag{133}$$

With our choice of

$$\bar{\lambda} = \max\left\{\frac{2eH}{K-2}, \frac{2\rho}{\sqrt{K}}, \frac{32eH\log(51200)}{K}, \frac{4\rho\sqrt{2\log(51200)}}{\sqrt{K}}, \frac{320\sqrt{2}\rho}{\sqrt{MK}}, \frac{320\sigma}{\bar{r}\sqrt{MK}}, \frac{8eH}{K-16}\right\}, \tag{134}$$

we note that

$$K \geq \frac{2}{\lambda}\max\left\{eH+\lambda, \frac{\rho^2}{\lambda}\right\}, \tag{135}$$

so for $\lambda \geq \bar{\lambda}$

$$\tilde{\epsilon}(\lambda) \leq 96\lambda r^*(\lambda)^2 \exp\left(-\frac{K}{8}\min\left\{\frac{\lambda}{eH+\lambda}, \frac{\lambda^2}{\rho^2}\right\}\right) + \frac{96(\sigma^2+\rho^2 r^*(\lambda)^2)}{\lambda MK}. \tag{136}$$

Furthermore, $K \geq 175$ and $\lambda \geq \bar{\lambda} \geq \max\left\{\frac{32eH\log(51200)}{K}, \frac{4\rho\sqrt{2\log(51200)}}{\sqrt{K}}\right\}$ implies

$$96\exp\left(-\frac{K}{8}\min\left\{\frac{\lambda}{eH+\lambda}, \frac{\lambda^2}{\rho^2}\right\}\right) \leq \frac{1}{1600}. \tag{137}$$

Likewise, $\lambda \geq \bar{\lambda} \geq \frac{320\sqrt{2}\rho}{\sqrt{MK}}$ implies

$$\frac{96\rho^2}{\lambda^2 MK} \leq \frac{1}{1600}. \tag{138}$$

Finally, $\lambda \geq \bar{\lambda} \geq \frac{320\sigma}{\bar{r}\sqrt{MK}}$ implies

$$\frac{96\sigma^2}{\lambda^2 MK} \leq \frac{\bar{r}^2}{800}. \tag{139}$$

Putting these together, we conclude that for $\lambda \geq \bar{\lambda}$

$$\tilde{\epsilon}(\lambda) \leq \frac{\lambda(r^*(\lambda)^2 + \bar{r}^2)}{800} = \epsilon(\lambda). \tag{140}$$

Combining this with (130), we conclude that the output of Algorithm 2 satisfies

$$\begin{aligned} &\mathbb{E}Q_t(\Delta\tilde{x}_t) - \min_{\Delta x:\|\Delta x\|\leq\frac{1}{2}\bar{r}} Q_t(\Delta x) \\ &\leq \frac{3}{4}\left(Q_t(0) - \min_{\Delta x:\|\Delta x\|\leq\frac{1}{2}\bar{r}} Q_t(\Delta x)\right) \\ &\qquad + 512\lambda\bar{r}^2\exp\left(-\frac{K}{8}\min\left\{\frac{\lambda}{eH+\lambda}, \frac{\lambda^2}{\rho^2}\right\}\right) + \frac{96(\sigma^2+16\rho^2\bar{r}^2)}{\lambda MK} + \frac{\bar{\lambda}\bar{r}^2}{4} \qquad (141) \\ &\leq \frac{3}{4}\left(Q_t(0) - \min_{\Delta x:\|\Delta x\|\leq\frac{1}{2}\bar{r}} Q_t(\Delta x)\right) \\ &\qquad + 512\lambda\bar{r}^2\exp\left(-\frac{K}{8}\min\left\{\frac{\lambda}{eH+\lambda}, \frac{\lambda^2}{\rho^2}\right\}\right) + \frac{\sigma\bar{r}+16\rho\bar{r}^2}{2\sqrt{MK}} + \frac{\bar{\lambda}\bar{r}^2}{4}. \qquad (142) \end{aligned}$$

Now, we upper bound $\lambda \mapsto \lambda \exp\Big(-\frac{K}{8}\min\Big\{\frac{\lambda}{eH+\lambda},\ \frac{\lambda^2}{\rho^2}\Big\}\Big)$ for $\lambda \in \Lambda_1$. First,

$$\lambda \exp\bigg(-\frac{K}{8}\min\bigg\{\frac{\lambda}{eH+\lambda},\ \frac{\lambda^2}{\rho^2}\bigg\}\bigg) = \max\bigg\{\lambda \exp\bigg(-\frac{\lambda K}{8eH+8\lambda}\bigg),\, \lambda\exp\bigg(-\frac{\lambda^2 K}{8\rho^2}\bigg)\bigg\}\,. \tag{143}$$

Considering each term separately,

$$\frac{d}{d\lambda}\bigg[\lambda\exp\bigg(-\frac{\lambda K}{8eH+8\lambda}\bigg)\bigg] = \exp\bigg(-\frac{\lambda K}{8eH+8\lambda}\bigg)\bigg(1-\frac{eHK\lambda}{8(eH+\lambda)^2}\bigg)\,. \tag{144}$$

This is less than zero if $8(eH+\lambda)^2 \leq eHK\lambda$, i.e.,

$$\frac{eH}{16}\Big(K-16-\sqrt{(K-16)^2-16}\Big) \leq \lambda \leq \frac{eH}{16}\Big(K-16+\sqrt{(K-16)^2-16}\Big). \tag{145}$$

With our choice of $\bar{\lambda} \geq \frac{8eH}{K-16}$, for any $\lambda \geq \bar{\lambda}$ and $K \geq 175$,

$$\lambda \geq \frac{8eH}{K-16} \geq \frac{eH}{16}\Big(K-16-\sqrt{(K-16)^2-16}\Big), \tag{146}$$

so the left side of this inequality is satisfied. Thus, for $\lambda \in \Lambda_1$ such that $\lambda \leq \frac{eH}{16}\Big(K-16+\sqrt{(K-16)^2-16}\Big)$,

$$\lambda\exp\bigg(-\frac{\lambda K}{8eH+8\lambda}\bigg) \leq \bar{\lambda}\exp\bigg(-\frac{\bar{\lambda}K}{8eH+8\bar{\lambda}}\bigg)\,. \tag{147}$$

Also, if $\lambda > \frac{eH}{16}\Big(K-16+\sqrt{(K-16)^2-16}\Big)$, then since $K \geq 175$

$$\lambda\exp\bigg(-\frac{\lambda K}{8eH+8\lambda}\bigg) \leq \lambda\exp\left(-\frac{\frac{eH(K-16)}{16}K}{8eH+\frac{eH(K-16)}{2}}\right) \leq \lambda\exp\bigg(-\frac{K}{10}\bigg)\,. \tag{148}$$

Furthermore, for $\lambda \in \Lambda_1$,

$$\lambda \leq \bar{\lambda}\bigg(\frac{3}{2}\bigg)^{N-1} \leq \frac{3\bar{\lambda}}{2}\frac{H(B+5T\bar{r})}{3\bar{\lambda}\bar{r}} = \frac{3H(B+5T\bar{r})}{6\bar{r}}\,. \tag{149}$$

Therefore, for any $\lambda \in \Lambda_1$

$$\lambda\exp\bigg(-\frac{\lambda K}{8eH+8\lambda}\bigg) \leq \max\bigg\{\bar{\lambda}\exp\bigg(-\frac{\bar{\lambda}K}{8eH+8\bar{\lambda}}\bigg),\, \frac{3H(B+5T\bar{r})}{6\bar{r}}\exp\bigg(-\frac{K}{10}\bigg)\bigg\}\,. \tag{150}$$

Similarly,

$$\frac{d}{d\lambda}\bigg[\lambda\exp\bigg(-\frac{\lambda^2 K}{8\rho^2}\bigg)\bigg] = \exp\bigg(-\frac{\lambda^2 K}{8\rho^2}\bigg)\bigg(1-\frac{\lambda^2 K}{4\rho^2}\bigg)\,. \tag{151}$$

This is negative for all $\lambda \geq \bar{\lambda} \geq \frac{2\rho}{\sqrt{K}}$, so

$$\lambda\exp\bigg(-\frac{\lambda^2 K}{8\rho^2}\bigg) \leq \bar{\lambda}\exp\bigg(-\frac{\bar{\lambda}^2 K}{8\rho^2}\bigg)\,. \tag{152}$$

We conclude that

$$\begin{aligned}
&\mathbb{E}Q_t(\Delta\tilde{x}_t) - \min_{\Delta x:\|\Delta x\|\leq\frac{1}{2}\bar{r}} Q_t(\Delta x) \\
&\quad\leq \frac{3}{4}\bigg(Q_t(0) - \min_{\Delta x:\|\Delta x\|\leq\frac{1}{2}\bar{r}} Q_t(\Delta x)\bigg) + \frac{\sigma\bar{r}+16\rho\bar{r}^2}{2\sqrt{MK}} + \frac{\bar{\lambda}\bar{r}^2}{4} \\
&\qquad + 512\bar{r}^2\max\bigg\{\bar{\lambda}\exp\bigg(-\frac{\bar{\lambda}K}{8eH+8\bar{\lambda}}\bigg),\, \frac{3H(B+5T\bar{r})}{6\bar{r}}\exp\bigg(-\frac{K}{10}\bigg),\, \bar{\lambda}\exp\bigg(-\frac{\bar{\lambda}^2K}{8\rho^2}\bigg)\bigg\}\,.
\end{aligned} \tag{153}$$

Now, because F is α-quasi-self-concordant, by Lemma 1, F is $\exp(\alpha r)$-locally stable, so with our choice of $\bar{r} \leq \frac{1}{5\alpha}$, we have that $\exp(\frac{1}{2}\bar{r})\exp(5\bar{r}) \leq e^{1.1} \leq 4$. Thus, it follows from Lemma 2, for $\theta = \frac{3}{4}$, combined with the guarantee on the output of Algorithm 2 from Lemma 3, that

$$\begin{aligned}
&\mathbb{E}F(x_T) - F^* \\
&\leq \mathbb{E}[F(x_0) - F^*]\exp\left(-\frac{T\bar{r}}{32B}\right) + \frac{32B}{\bar{r}}\left(\frac{\sigma\bar{r} + 16\rho\bar{r}^2}{2\sqrt{MK}} + \frac{\bar{\lambda}\bar{r}^2}{4}\right) \\
&\qquad + \frac{32B}{\bar{r}} \cdot 512\bar{r}^2 \max\left\{\bar{\lambda}\exp\left(-\frac{\bar{\lambda}K}{8eH + 8\bar{\lambda}}\right), \frac{3H(B + 5T\bar{r})}{6\bar{r}}\exp\left(-\frac{K}{10}\right), \bar{\lambda}\exp\left(-\frac{\bar{\lambda}^2 K}{8\rho^2}\right)\right\} \qquad (154) \\
&\leq \mathbb{E}[F(x_0) - F^*]\exp\left(-\frac{T\bar{r}}{32B}\right) + \frac{32\sigma B + 512\rho B\bar{r}}{2\sqrt{MK}} + 8\bar{\lambda}B\bar{r} \\
&\qquad + 2^{14}\max\left\{\bar{\lambda}B\bar{r}\exp\left(-\frac{\bar{\lambda}K}{8eH + 8\bar{\lambda}}\right), \frac{3H(B^2 + 5TB\bar{r})}{6}\exp\left(-\frac{K}{10}\right), \bar{\lambda}B\bar{r}\exp\left(-\frac{\bar{\lambda}^2 K}{8\rho^2}\right)\right\} \qquad (155) \\
&\leq \mathbb{E}[F(x_0) - F^*]\exp\left(-\frac{T\bar{r}}{32B}\right) + \frac{32\sigma B + 512\rho B\bar{r}}{2\sqrt{MK}} \\
&\qquad + \bar{\lambda}B\bar{r}\left(8 + 2^{14}\max\left\{\exp\left(-\frac{\bar{\lambda}K}{8eH + 8\bar{\lambda}}\right), \frac{3H(B + 5T\bar{r})}{6\bar{\lambda}\bar{r}}\exp\left(-\frac{K}{10}\right), \exp\left(-\frac{\bar{\lambda}^2 K}{8\rho^2}\right)\right\}\right). \qquad (156)
\end{aligned}$$

We have, for a constant c,

$$\begin{aligned}
\bar{\lambda} &= \max\left\{\frac{2eH}{K-2}, \frac{2\rho}{\sqrt{K}}, \frac{32eH\log(51200)}{K}, \frac{4\rho\sqrt{2\log(51200)}}{\sqrt{K}}, \frac{320\sqrt{2}\rho}{\sqrt{MK}}, \frac{320\sigma}{\bar{r}\sqrt{MK}}, \frac{8eH}{K-16}\right\} \qquad (157) \\
&= \max\left\{\frac{32eH\log(51200)}{K}, \frac{4\rho\sqrt{2\log(51200)}}{\sqrt{K}}, \frac{320\sqrt{2}\rho}{\sqrt{MK}}, \frac{320\sigma}{\bar{r}\sqrt{MK}}\right\} \qquad (158) \\
&= c \cdot \max\left\{\frac{H}{K}, \frac{\rho}{\sqrt{K}}, \frac{\sigma}{\bar{r}\sqrt{MK}}\right\}. \qquad (159)
\end{aligned}$$

So, for a constant c', and using $\mathbb{E}F(x_0) - F^* \leq \frac{HB^2}{2}$,

$$\begin{aligned}
&\mathbb{E}F(x_T) - F^* \\
&\leq c' \cdot \left(HB^2\exp\left(-\frac{T\bar{r}}{32B}\right) + \frac{\sigma B + \rho B\bar{r}}{\sqrt{MK}}\right. \\
&\qquad \left. + \bar{\lambda}B\bar{r}\left(1 + \max\left\{\exp\left(-\frac{\bar{\lambda}K}{H + \bar{\lambda}}\right), \frac{H(B + T\bar{r})}{\bar{\lambda}\bar{r}}\exp\left(-\frac{K}{10}\right), \exp\left(-\frac{\bar{\lambda}^2 K}{\rho^2}\right)\right\}\right)\right) \qquad (160) \\
&\leq c' \cdot \left(HB^2\exp\left(-\frac{T\bar{r}}{32B}\right) + \frac{\sigma B + \rho B\bar{r}}{\sqrt{MK}} + \max\left\{\bar{\lambda}B\bar{r}, H(B^2 + TB\bar{r})\exp\left(-\frac{K}{10}\right)\right\}\right) \qquad (161) \\
&= c' \cdot \left(HB^2\exp\left(-\frac{T\bar{r}}{32B}\right) + \frac{\sigma B + \rho B\bar{r}}{\sqrt{MK}}\right. \\
&\qquad \left. + \max\left\{\frac{HB\bar{r}}{K}, \frac{\rho B\bar{r}}{\sqrt{K}}, \frac{\sigma B}{\sqrt{MK}}, H(B^2 + TB\bar{r})\exp\left(-\frac{K}{10}\right)\right\}\right) \qquad (162) \\
&= c' \cdot \left(HB^2\exp\left(-\frac{T\bar{r}}{32B}\right) + \frac{\sigma B}{\sqrt{MK}} + \frac{HB\bar{r}}{K} + \frac{\rho B\bar{r}}{\sqrt{K}} + H(B^2 + TB\bar{r})\exp\left(-\frac{K}{10}\right)\right). \qquad (163)
\end{aligned}$$

So, since $\bar{r} = \min\{\frac{32B}{T}\log(TK),\, \frac{1}{5\alpha}\}$, and using the fact that, for $\zeta, a, b > 0$, $e^{-\zeta\min\{a,b\}} \le e^{-\zeta a} + e^{-\zeta b}$, we have, for a constant c'',

$$\begin{aligned}
\mathbb{E}F(x_T) - F^* &\le c'' \cdot \Bigg(HB^2\exp\Big(-\frac{T}{160\alpha B}\Big) + \frac{HB^2}{TK} + \frac{\sigma B}{\sqrt{MK}} + \frac{HB^2\log TK}{TK} \\
&\qquad + \frac{\rho B^2\log TK}{T\sqrt{K}} + HB^2\log(TK)\exp\Big(-\frac{K}{10}\Big)\Bigg) && (164)\\
&\le c'' \cdot \Bigg(HB^2\exp\Big(-\frac{T}{160\alpha B}\Big) + \frac{\sigma B}{\sqrt{MK}} + \frac{HB^2\log TK}{TK} \\
&\qquad + \frac{\rho B^2\log TK}{T\sqrt{K}} + HB^2\log(TK)\exp\Big(-\frac{K}{10}\Big)\Bigg), && (165)
\end{aligned}$$

where the last inequality follows from the fact $\log(TK) \ge 1$, since $TK \ge 175$.

Finally, each call to Algorithm 2 requires at most $C\lceil\log N\rceil$ rounds of communication (one for each call to Algorithm 3). Therefore, we can implement up to $R/(C\lceil\log N\rceil)$ iterations of Algorithm 1 using our R rounds of communication. We recall that N and C are set as

$$\begin{aligned}
N &= \left\lceil 1 + \frac{5}{2}\log\frac{H(B+5T\bar{r})}{3\bar{\lambda}\bar{r}}\right\rceil && (166)\\
C &= \left\lceil 8\log\left(\lceil\log_2 N\rceil\left(4 + \frac{eH}{\bar{\lambda}} + \frac{80H(B+5T\bar{r})}{\bar{\lambda}\bar{r}}\right)\right)\right\rceil. && (167)
\end{aligned}$$

Therefore, we need to choose T such that $T \le \frac{R}{C\lceil\log N\rceil}$. To provide an explicit lower bound on how large T can be, we therefore lower bound the right hand side. First, we have

$$\begin{aligned}
N &= \left\lceil 1 + \frac{5}{2}\log\frac{H(B+5T\bar{r})}{3\bar{\lambda}\bar{r}}\right\rceil && (168)\\
&\le 2 + \frac{5}{2}\log\frac{H\big(B + 5T\frac{B}{T}\log(TK)\big)}{3\frac{2eH}{K-2}\min\{\frac{32B}{T}\log(TK),\, \frac{1}{5\alpha}\}} && (169)\\
&\le 2 + \frac{5}{2}\log\frac{BK\log(TK)}{e\min\{\frac{32B}{T}\log(TK),\, \frac{1}{5\alpha}\}} && (170)\\
&\le 2 + \frac{5}{2}\max\{\log(TK\log(TK)),\, \log(2\alpha BK\log(TK))\} && (171)\\
&\le 2 + 5\log(1+2\alpha B) + 5\log(TK). && (172)
\end{aligned}$$

Similarly,

$$\begin{aligned}
C &= \left\lceil 8\log\left(\lceil\log_2 N\rceil\left(4 + \frac{eH}{\bar{\lambda}} + \frac{80H(B+5T\bar{r})}{\bar{\lambda}\bar{r}}\right)\right)\right\rceil && (173)\\
&\le 1 + 8\log\left(\lceil\log_2 N\rceil\left(4 + \frac{eH}{\frac{2eH}{K-2}} + 240\max\{TK\log(TK),\, 2\alpha BK\log(TK)\}\right)\right) && (174)\\
&\le 1 + 8\log\big(\lceil\log_2 N\rceil\big(4K + 240\max\{T^2K^2,\, 2\alpha BTK^2\}\big)\big) && (175)\\
&\le 1 + 8\log\big((1+\log N)\big(353T^2K^2 + 693\alpha BTK^2\big)\big) && (176)\\
&\le 1 + 8\log(693) + 16\log(TK) + 8\log(1+\alpha B) + 8\log(1+\log(N)) && (177)\\
&\le 54 + 16\log(TK) + 8\log(1+\alpha B) + 16\log\log N && (178)\\
&\le 80 + 32\log(TK) + 24\log(1+2\alpha B). && (179)
\end{aligned}$$

Therefore,

$$\frac{R}{C\lceil \log N \rceil} \geq \frac{R}{(80 + 32\log(TK) + 24\log(1+2\alpha B))\lceil \log(2 + 5\log(1+2\alpha B) + 5\log(TK)) \rceil} \tag{180}$$

$$\geq \frac{R}{(80 + 32\log(TK) + 24\log(1+2\alpha B))^2} \tag{181}$$

$$\geq \frac{R}{2048 + 2(80 + 32\log K + 24\log(1+2\alpha B))^2}\ . \tag{182}$$

So, it suffices to choose T such that

$$T\log^2(T) \leq \frac{R}{2048 + 2(80 + 32\log K + 24\log(1+2\alpha B))^2}\ . \tag{183}$$

Note that equality holds for

$$T = \exp\left(2W\left(\left(\frac{R}{4096 + 4(80 + 32\log K + 24\log(1+2\alpha B))^2}\right)^{1/2}\right)\right),$$

where $W(\cdot)$ denotes the Lambert W function. Thus, because $W(x) \geq \log(x) - \log\log(x)$ for $x \geq e$, and since we assume

$$R \geq \tilde{\Omega}(1) = e^2\Big(4096 + 4(80 + 32\log K + 24\log(1+2\alpha B))^2\Big),$$

we have that

$$\begin{aligned}
&\exp\left(2W\left(\left(\frac{R}{4096 + 4(80 + 32\log K + 24\log(1+2\alpha B))^2}\right)\right)^{1/2}\right) \\
&\geq \exp\left(2\left(\log\left(\frac{R}{4096 + 4(80 + 32\log K + 24\log(1+2\alpha B))^2}\right)^{1/2}\right.\right. \\
&\qquad \left.\left. - \log\log\left(\frac{R}{4096 + 2(80 + 32\log K + 24\log(1+2\alpha B))^2}\right)^{1/2}\right)\right) \\
&= \frac{1}{4}\left(\frac{R}{4096 + 4(80 + 32\log K + 24\log(1+2\alpha B))^2}\right) \\
&\qquad \cdot \log^2\left(\left(\frac{R}{4096 + 4(80 + 32\log K + 24\log(1+2\alpha B))^2}\right)\right).
\end{aligned}$$

Thus, letting

$$\begin{aligned}
T = \Bigg\lfloor &\frac{1}{4}\left(\frac{R}{4096 + 4(80 + 32\log K + 24\log(1+2\alpha B))^2}\right) \\
&\cdot \log^2\left(\left(\frac{R}{4096 + 4(80 + 32\log K + 24\log(1+2\alpha B))^2}\right)\right)\Bigg\rfloor,
\end{aligned}$$

and using $\tilde{O}$ notation to hide polylogarithmic factors in R, K, and αB, we have

$$\mathbb{E}[F(x_T)] - F^* \leq HB^2\left(\exp\left(-\frac{R}{\tilde{O}(\alpha B)}\right) + \exp\left(-\frac{K}{O(1)}\right)\right) + \tilde{O}\left(\frac{\sigma B}{\sqrt{MK}} + \frac{HB^2}{KR} + \frac{\rho B^2}{\sqrt{K}R}\right)\ . \tag{184}$$

This completes the proof. □

F Additional Comparisons

Here we include additional comparisons, recalling that we we are working in the natural scaling of parameters that arises when the objective has the form $F(x) = \mathbb{E}_z \ell(\langle x,\, z\rangle)$, where $|\ell'|$, $|\ell''|$, and $|\ell'''|$ are $O(1)$, and where $\|z\| \leq D$, which provides for setting the parameters as $H = D^2$, $\sigma = D$, $U = D^3$, $\alpha = D$, and $\rho = D^2$.

Comparison with Local SGD

Recall that, under third-order smoothness assumptions (Yuan and Ma, 2020), Local SGD converges at a rate of

$$\mathbb{E}F(\hat{x}) - F^* \leq \tilde{O}\left(\frac{HB^2}{KR} + \frac{\sigma B}{\sqrt{MKR}} + \frac{U^{1/3}\sigma^{2/3}B^{5/3}}{K^{1/3}R^{2/3}}\right).$$

For the setting as outlined above, this bound reduces to

$$\mathbb{E}F(\hat{x}) - F^* \lesssim \frac{D^2B^2}{KR} + \frac{D^2B^2}{KR^2} + \frac{D^{5/3}B^{5/3}}{K^{1/3}R^{2/3}} \approx \frac{D^2B^2}{KR} + \frac{D^{5/3}B^{5/3}}{K^{1/3}R^{2/3}}.$$

In the case where DB is not too large ($DB \lesssim K^2R$), the dominant term for Local SGD is $\frac{D^{5/3}B^{5/3}}{K^{1/3}R^{2/3}}$, and so we see that our algorithm improves upon Local SGD as long as $R \gtrsim \frac{D^{1/3}B^{1/3}}{K^{1/6}}$.

Comparison with min-max method under Assumption 1

Woodworth et al. (2021) identified the min-max optimal (up to logarithmic factors) stochastic first-order method in the distributed setting we consider, and under Assumption 1. Namely, a min-max optimal method can be obtained by combining Minibatch-Accelerated-SGD (Cotter et al., 2011), which enjoys a guarantee of

$$\mathbb{E}F(\hat{x}) - F^* \leq O\left(\frac{HB^2}{R^2} + \frac{\sigma B}{\sqrt{MKR}}\right),$$

with Single-Machine Accelerated SGD, which runs KR steps of an accelerated variant of SGD known as AC-SA (Lan, 2012), and enjoys a guarantee of

$$\mathbb{E}F(\hat{x}) - F^* \leq O\left(\frac{HB^2}{K^2R^2} + \frac{\sigma B}{\sqrt{KR}}\right).$$

Therefore, an algorithm which returns the output of Minibatch Accelerated SGD when $K \leq \frac{\sigma^2R^3}{H^2B^2}$, and otherwise returns the output of Single-Machine Accelerated SGD, achieves a guarantee of

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

As shown by Woodworth et al. (2021), this matches the lower bound for stochastic distributed first-order optimization under Assumption 1, up to $O(\log^2 M)$ factors.

For the setting as outlined above, this bound reduces to

$$\mathbb{E}F(\hat{x}) - F^* \lesssim \frac{D^2B^2}{K^2R^2} + \frac{D^2B^2}{KR^2} + \min\left\{\frac{D^2B^2}{R^2}, \frac{DB}{\sqrt{KR}}\right\} \approx \frac{D^2B^2}{K^2R^2} + \min\left\{\frac{D^2B^2}{R^2}, \frac{DB}{\sqrt{KR}}\right\}.$$

Thus, for $DB \lesssim \frac{R^{3/2}}{\sqrt{K}}$, the dominant term is $\frac{D^2B^2}{R^2}$, in which case FEDSN is better as long as $R \lesssim \sqrt{K}$. Furthermore, for $\frac{R^{3/2}}{\sqrt{K}} \lesssim DB \lesssim K^{3/2}R^{3/2}$, the dominant term is $\frac{DB}{\sqrt{KR}}$, in which case FEDSN is better as long as $R \gtrsim D^2B^2$. In these regimes, we see that Assumption 2 allows FEDSN to improve over the best possible when relying only on Assumption 1. We also note that when $DB \gtrsim K^{3/2}R^{3/2}$, the combined algorithm described above is better than the guarantee we prove for FEDSN—we do not know whether this is a weakness of our analysis or represents a true deficiency of FEDSN in this regime.

G Additional Information for Experiments

G.1 Baselines

We first present a more practical version of our algorithm FEDSN, called FEDSN-LITE. It differs from FEDSN in two major ways: first, it directly uses REGULARIZED-QUADRATIC-SOLVER (Algorithm 3) (with $\bar{\xi} = 1$) as an approximate quadratic solver without requiring a search over the regularization parameter as in Algorithm 2; and in addition, it scales the Newton update with a stochastic approximation of the Newton decrement (see Nesterov (1998); Boyd and Vandenberghe (2004)) and a constant stepsize ν (we use $\nu = 1.25$ throughout our experiments). Because of these changes, the Newton update is fairly robust to the choice of ν, so we then only need to tune the learning rate of REGULARIZED-QUADRATIC-SOLVER, making it as usable as any other first-order method.

Algorithm 6 FEDSN-LITE(x_0)

```
(Operating on objective $F(\cdot)$ with stochastic gradient $g(\cdot;\cdot)$ and Hessian-vector product $h(\cdot;\cdot,\cdot)$
  oracles.)
Input: $x_0 \in \mathbb{R}^d$.
Hyperparameters: $T$: main iterations; $\nu$: Newton stepsize scale; and $\beta$: momentum.
  for $t = 0, 1, \ldots, T-1$ do
    REGULARIZED-QUADRATIC-SOLVER$(x_t, 0)$
    $\nu_t := \nu \left(1 + (\Delta\tilde{x}_t^\top h(x_t, \Delta\tilde{x}_t; z))^{1/2}\right)^{-1}$    ▷ $h(x_t, \Delta\tilde{x}_t; z)$ is s.t. $\mathbb{E}_z[h(x_t, \Delta\tilde{x}_t; z)] = \nabla^2 F(x_t)\Delta\tilde{x}_t$ [4]
    Update: $x_{t+1} = x_t + \nu_t \Delta\tilde{x}_t$
  Return: $x_T$
```

Note that in each step of FEDSN-LITE, the subroutine REGULARIZED-QUADRATIC-SOLVER approximately solves a quadratic subproblem using a variant of one-shot averaging (we set the parameter functions as $\eta(\lambda) = \eta$ and $w(\lambda) = \frac{1}{K}$). Moreover, we implement FEDSN-LITE such that each call to the stochastic oracle from within REGULARIZED-QUADRATIC-SOLVER uses only a single sample (Case 2 in Algorithm 4), and so it is asymptotically as expensive as the gradient oracle (i.e., $O(d)$, if d is the dimension of the problem). For a discussion of the computational cost of using a Hessian-vector product oracle, see Appendix G.4.

We have compared FEDSN-LITE against the two variants of FEDAC (Yuan and Ma, 2020), Minibatch SGD (Dekel et al., 2012), and Local SGD (Zinkevich et al., 2010). Two settings of hyperparameters are considered for FEDAC in Yuan and Ma (2020) for strongly convex functions:

- **FEDAC-I**: $\eta \in (0, 1/H],\ \gamma = \max\left\{\sqrt{\frac{\eta}{\lambda K}}, \eta\right\},\ \alpha = \frac{1}{\gamma\lambda}, \beta = \alpha + 1;$
- **FEDAC-II**: $\eta \in (0, 1/H],\ \gamma = \max\left\{\sqrt{\frac{\eta}{\lambda K}}, \eta\right\},\ \alpha = \frac{3}{2\gamma\lambda} - \frac{1}{2}, \beta = \frac{2\alpha^2-1}{\alpha-1},$

where H is the smoothness constant as in Assumption 1, λ is an estimate of the strong convexity, and η is the learning rate which has to be tuned. Thus, a limitation of FEDAC is that it requires either the knowledge of λ (say, through explicit λ-regularization), or that the algorithm adds regularization to the objective itself (this is how Yuan and Ma (2020) present FEDAC for general convex functions). In our experiments we have both of these settings, i.e., FEDAC with internal regularization λ (c.f., figures 1a and 1b) or explicitly λ-regularized objectives (c.f., figures 4 and 5). For brevity, in Algorithm 7 we present FEDAC with five hyperparameters: α, β, η, γ, and λ. When the objective is regularized we use $\lambda = 0$, whereas otherwise we tune λ, to ensure the best possible performance for FEDAC.

[4]We can calculate an approximation to the Newton decrement at $\Delta\tilde{x}_t$, i.e., $(\Delta\tilde{x}_t^\top \nabla^2 F(x_t)\Delta\tilde{x}_t)^{1/2}$, using a call to the Hessian-vector product oracle (see Assumption 2(b)), along with an additional dot product, namely $\Delta\tilde{x}_t^\top h(x_t, \Delta\tilde{x}_t; z)$.

Algorithm 7 FEDAC$(x_0, \alpha, \beta, \eta, \gamma, \lambda)$

```
(Operating on objective $F(\cdot) + \lambda/2\|.\|^2$ as opposed to $F(.)$ with stochastic gradient oracle $g_\lambda(\cdot;\cdot)$[5].)
  Intitialize: $x_0^{ag,m} = x_0^m = x_0$ for all $m \in [M]$
  for $t = 0, 1, \ldots, T-1$ do
    for every worker $m \in [M]$ in parallel do
      $x_t^{md,m} \leftarrow \beta^{-1} x_t^m + (1 - \beta^{-1}) x_t^{ag,m}$
      $g_t^m \leftarrow g_\lambda(x_t^{md,m}, z_t^m)$    ▷ Query the stochastic first-order oracle at $x_t^{md,m}$, for $z_t^m \sim \mathcal{D}$
      $v_{t+1}^{ag,m} \leftarrow x_t^{md,m} - \eta \cdot g_t^m$
      $v_{t+1}^m \leftarrow (1 - \alpha^{-1}) x_t^m + \alpha^{-1} x_t^{md,m} - \gamma g_t^m$
      if $t$ mod $K = -1$ then
        $x_{t+1}^m \leftarrow \frac{1}{M} \sum_{m'=1}^M v_{t+1}^{m'}$
        $x_{t+1}^{ag,m} \leftarrow \frac{1}{M} \sum_{m'=1}^M v_{t+1}^{ag,m'}$
      else
        $x_{t+1}^m \leftarrow v_{t+1}^m$
        $x_{t+1}^{ag,m} \leftarrow v_{t+1}^{ag,m}$
  Return: $\bar{x}_T^{ag} = \frac{1}{M} \sum_{m'=1}^M x_T^{ag,m'}$
```

In Algorithm 8 we describe Local SGD (a.k.a. FEDAVG) (Zinkevich et al., 2010) with learning rate η and Polyak's momentum (a.k.a. heavy ball method) parameter β. Setting $\beta = 0$ recovers the familiar algorithm as analyzed in Woodworth et al. (2020a). Finally, in Algorithm 9 we describe Minibatch SGD with fixed learning rate η and momentum parameter β.

Algorithm 8 Local SGD(x_0, β, η)

```
  (Operating on objective $F(\cdot)$ with stochastic gradient oracle $g(\cdot;\cdot)$.)
  Intitialize: $x_{0,0}^m \leftarrow x_0$ for all $m \in [M]$
  for $r = 0, \ldots, R-1$ do
    for every worker $m \in [M]$ in parallel do
      for $k = 0, \ldots, K-1$ do
        $g_{r,k}^m \leftarrow g(x_{r,k}^m; z_{r,k}^m)$  ▷ Query the stochastic first-order oracle at $x_{r,k}^m$ (for $z_{r,k}^m \sim \mathcal{D}$ drawn
        by the oracle)
        $x_{r,k+1}^m \leftarrow x_{r,k}^m - \eta g_{r,k}^m + \mathbb{1}_{k>0} \beta (x_{r,k}^m - x_{r,k-1}^m)$
    $x_{r+1} \leftarrow \frac{1}{M} \sum_{m'=1}^M x_{r,K}^{m'}$
  Return: $x_R$
```

Algorithm 9 Minibatch SGD(x_0, β, η)

```
  (Operating on objective $F(\cdot)$ with stochastic gradient oracle $g(\cdot;\cdot)$.)
  Intitialize: $x_0^m = x_0$ for all $m \in [M]$
  for $r = 0, \ldots, R-1$ do
    for every worker $m \in [M]$ in parallel do
      for $k = 0, \ldots, K-1$ do
        $g_{r,k}^m \leftarrow g(x_r; z_{r,k}^m)$  ▷ Query the stochastic first-order oracle at $x_r$ (for $z_{r,k}^m \sim \mathcal{D}$ drawn by
        the oracle)
    $g_r \leftarrow \frac{1}{KM} \sum_{k'=0}^{K-1} \sum_{m'=1}^M g_{r,k'}^{m'}$
    $x_{r+1} \leftarrow x_r - \eta g_r + \mathbb{1}_{r>0} \beta (x_r - x_{r-1})$
  Return: $x_R$
```

Note that in our experiments we compare the algorithms using the same number of machines M, communication rounds R and theoretical parallel runtime T against each other, where $T = KR$.

[5]Note that $g_\lambda(x; z) = g(x; z) + \lambda x$, i.e., the stochastic oracle for the regularized objective can always be obtained using the oracle for the unregularized objective (see Assumption 1).)

Also note that unlike Algorithm 7, there is no internal regularization in Algorithms 8 and 9. While conducting experiments for figures 4 and 5, we assume that $F(.)$ is regularized, so that Algorithms 8 and 9 instead minimize $F(.) + \frac{\lambda}{2}\|.\|^2$ and have access to $g_\lambda(.;.)$. The reason why λ is not presented as a hyperparameter for any algorithms beside FEDAC, is because the regularization is a part of the optimizer FEDAC itself determines its other hyper-parameters.

In addition, we may see that REGULARIZED-QUADRATIC-SOLVER with $\eta_k(\lambda) = \eta$ and $w_k(\lambda) = \frac{1}{K}$ (as in each iteration of FEDSN-LITE) is a special case of Local SGD (Algorithm 8) with $R = 1$. Thus to summarize, in our experiments, we have compared the following algorithms:

- FEDSN-LITE (which uses REGULARIZED-QUADRATIC-SOLVER) without momentum i.e., $\beta = 0$ or with optimally tuned momentum $\beta \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$,
- FEDAC-1 and FEDAC-2, with either no internal regularization i.e., $\lambda = 0$ (in experiment 2 when the objective is explicitly regularized) or optimally tuned internal regularization $\lambda \in \{$1e- 2, 1e-3, 1e-4, 1e-5, 1e-6$\}$ (in experiment 1),
- Local SGD without momentum i.e., $\beta = 0$ or with tuned momentum $\beta \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$,
- Minibatch SGD without momentum i.e., $\beta = 0$ or with tuned momentum $\beta \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$,

We search for the best learning rate $\eta \in \{0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20\}$ for every configuration of each algorithm. We verify (along with Yuan and Ma (2020)) that the optimal learning rate always lies in this range.

G.2 More extensive experiments

First, we present more comprehensive versions of the experiments presented in Section 5.

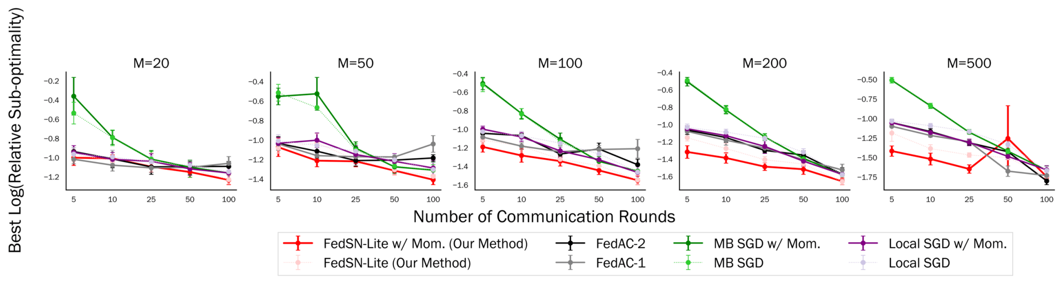


Figure 2: The same experiment as in fig. 1a but with a broader range of values of M. All optimization runs were repeated 30 times.

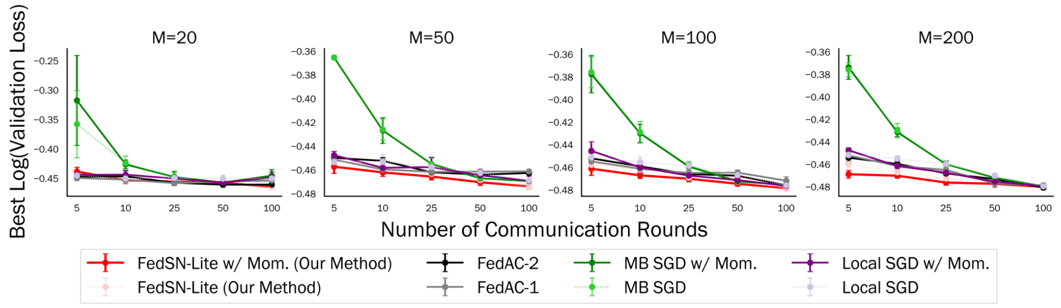


Figure 3: The same experiment as in fig. 1b but with a broader range of values of M. All optimization runs were repeated 20 times.

Recall, how comparison to FEDAC requires the knowledge of the strong convexity constant. To alleviate this in figures 2 and 3 we added an internal regularization parameter to FEDAC and tuned it. Another way to deal with this is to solve a regularized empirical risk minimization problem, where FEDAC knows the level of regularization, which serves as a proxy for strong convexity. Next we consider exactly this setting in figures 4 and 5, where we provide FEDAC the regularization strength μ. Unlike previous experiments, here we report regularized training loss, i.e., we train the models to optimize $F(x) + r(x) = \sum_{i \in S} f_i(x) + \frac{\mu}{2}\|x\|_2^2$, for a finite dataset S. We also vary the regularization strength μ, to understand the algorithms' dependence on the problem's conditioning. This was precisely the experiment conducted by Yuan and Ma (2020) (c.f., Figures 3 and 5 in their paper).

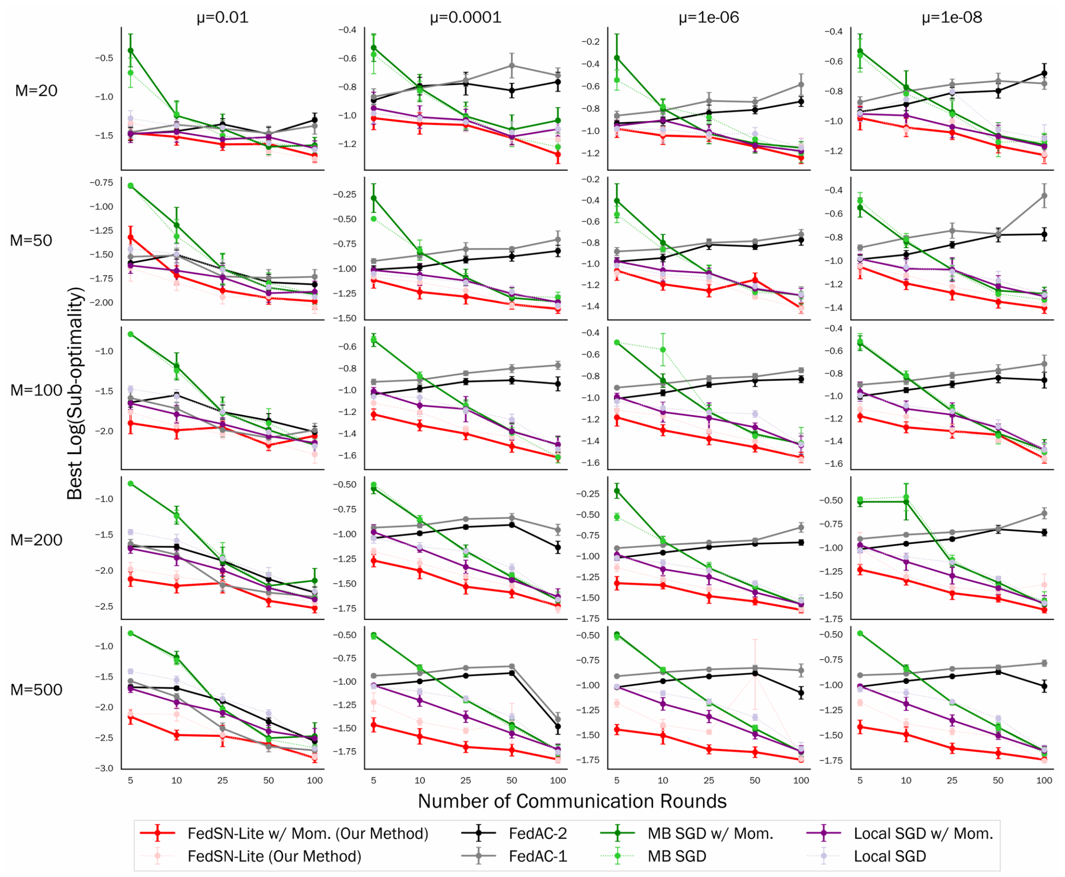


Figure 4: Empirical comparison of FEDSN-LITE (Algorithm 6) to other methods (see Appendix G.1 for complete details) on the LIBSVM a9a (Chang and Lin, 2011; Dua and Graff, 2017) dataset for minimizing μ-regularized logistic regression loss using M machines and μ regularization strength. We vary the frequency of communication (horizontal axis of each plot), while keeping the total number of steps on each machine (theoretical parallel runtime) fixed at $KR = 100$. Each algorithm besides FEDAC solves a regularized ERM problem, and reports the best relative sub-optimality w.r.t. the optimal minimizer. For FEDAC we use μ as the strong convexity constant to tune its hyperparameters. For the other algorithms we tune the learning rate for either $\beta = 0$, i.e., without momentum, or for all $\beta \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, to choose the optimum level of momentum. We repeat this tuning procedure for each value of μ, M, R, and thus each point in the plot represents an optimal configuration of that algorithm for that setting. All optimization runs were repeated 70 times with the tuned hyperparameters.

G.3 More implementation details

Dataset. Following the setup in Yuan and Ma (2020), we use the full LIBSVM a9a (Chang and Lin, 2011; Dua and Graff, 2017) dataset. It is a binary classification dataset with 32,561 points and 123 features. For generating figures 1a, 2, 4 and 5 we use the entire dataset as a training set. On the other hand, For generating figures 1b and 3 we split the dataset, using 20,000 points as the training set and the rest as the validation set.

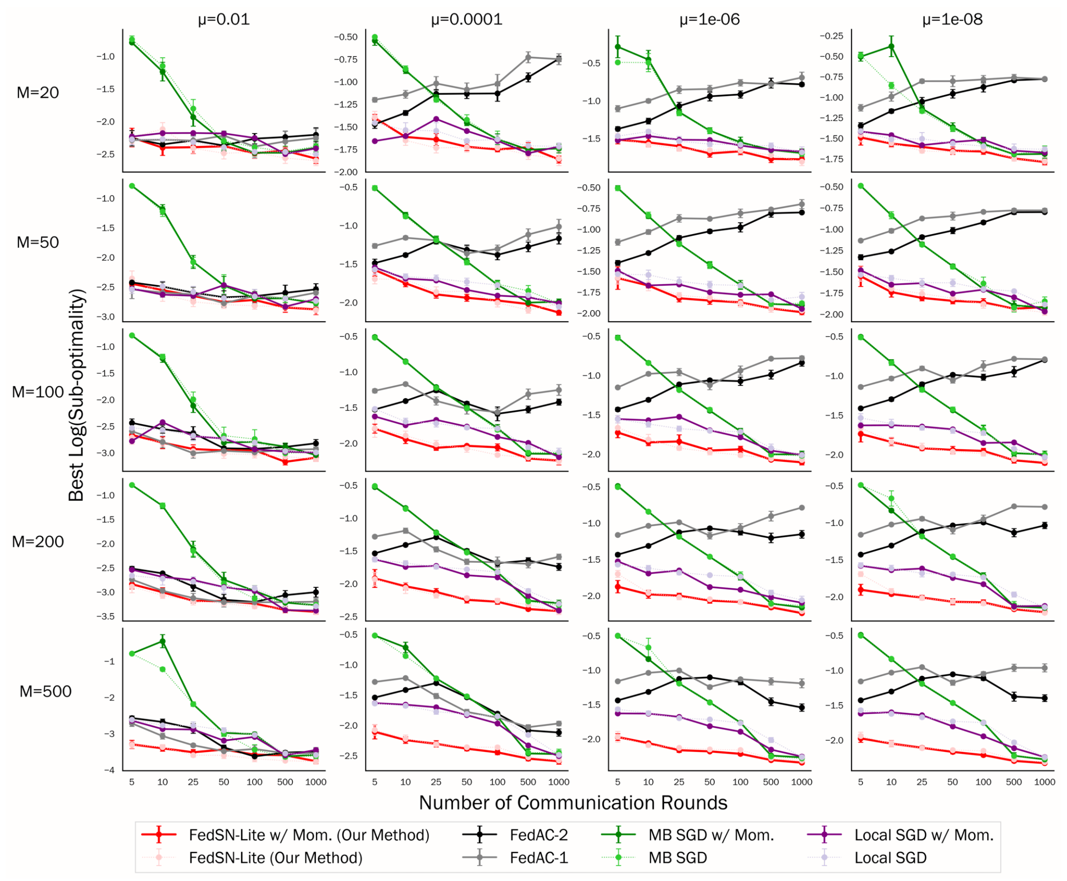


Figure 5: Same setting as figure 4 but with a longer parallel run-time of $T = 1000$.

Figures 4 and 5. As described in the body of the paper, each sub-plot (in Figure 4) shows the performance of the different algorithms and their variants (as discussed above) on μ-regularized logistic regression. For the experiments we vary the values of μ, M, and R, for fixed theoretical parallel runtime KR. Each of these is a different *setting*, reflecting different computation-communication-accuracy trade-offs. For each of these settings, we run an algorithm configuration (with some β and $\lambda = 0$ for FEDAC), with different learning rates η, and record the best loss (which is the Regularized ERM-loss on the entire dataset) obtained *at any point* during optimization. Based on this run we pick the optimal learning rate for each setting for each algorithm from the set $\{0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20\}$.

Note that this means every point in a sub-plot is an individual experiment, for which we have tuned. Since we are only concerned with optimization, we tune and report the accuracy only on the training set (with 32,561 points). Both the stochastic first-order oracle and the Hessian-vector product oracle, used by the respective algorithms, do sampling with replacement. Following Yuan and Ma (2020), we ensure that our learning rate tuning grid was both wide and fine enough for the algorithms to achieve their optimal training losses. Since we are using stochastic algorithms which sample with replacement, without any definite order on the dataset (as opposed to making a single pass), the best suboptimality is potentially different for each run of the algorithm, even when choosing the same learning rate and initialization. Thus, once we have the optimal parameters (η, β, λ), we rerun the algorithm multiple times on the corresponding setting. The error bars represent the standard deviation of the best suboptimality obtained when using this optimal learning rate.

Note that changing the regularization strength changes the optimal error value on the RERM task. Thus, to get the optimal loss value, we run exact Newton's method separately on each of these settings (for different values of μ), until the algorithm had converged up to ≈ 12 decimal places. We also ensure that our optimal loss values look similar to Yuan and Ma (2020). The reported suboptimality in Figure 1a is obtained after subtracting this optimal error value from the algorithms' error, followed by dividing this excess error with the optimal value.

Figures 1a and 2. As described in the body of the paper, each sub-plot in Figure 1a shows the performance of the different algorithms and their variants (as discussed above) on logistic regression for getting the best possible unregularized training loss. We train FEDAC on an appropriately regularized RERM problem on the training set. This is achieved through its internal regularization parameter λ as described above. All other algorthms minimize an ERM problem on the dataset. We vary M and R while keeping the parallel runtime KR to be fixed. Our hyperparameter tuning and repetitions are similar to the description for the experiment above, though we additionally tune the regularization strength λ in the RERM problem for FEDAC. All the algorithms do sampling with replacement on the training set, and can make multiple passes on the dataset. The optimal training loss is again obtained using 100 iterations of the Newton method.

Figures 1b and 3. As described in the body of the paper, each sub-plot in Figure 1b shows the performance of the different algorithms and their variants (as discussed above) on logistic regression for getting the best possible validation loss. We train FEDAC on an appropriately regularized RERM problem on the training set. All other variants minimize an ERM problem on the training set. This is achieved through the internal regularization parameter for FEDAC as discussed above. We vary M and R while keeping the parallel runtime KR to be fixed. Our hyperparameter tuning and repetitions are similar to the description for the experiments above, though we additionally tune the regularization strength λ in the RERM problem for FEDAC. We use the same split of training and validation datasets across the multiple repetitions. All the algorithms do sampling without replacement on the training set, and can make at most one pass on the training dataset.

Note that in all our experiments, we are concerned with minimizing a convex function $F(x)$. In some cases we are supposedly minimizing a finite sum, i.e., $F(x) = \sum_{i\in S} f_i(x)$ where S is our training dataset, for e.g., in Figure 1a. In others $F(x) = \mathbb{E}_{z\in\mathcal{D}}[f(x;z)]$, i.e., a stochastic optimization problem where we access the distribution $\mathcal{D}$ through our finite sample S, for e.g., in Figure 1b. To estimate the true-error on $\mathcal{D}$, we split S into S_{train} and S_{val}, then sampled without replacement from S_{train} to train our models, and reported the final performance on S_{val}.

Hardware details. All the experiments were performed on a personal computer, Dell XPS 7390. The total compute time (CPU) on the machine was about 300 hours. No GPUs were used in any of the experiments in this paper.

G.4 Computational cost of a Hessian-vector product oracle

Consider minimizing the function $F(x) = \mathbb{E}_z[f(x;z)]$ given access to various stochastic oracles. Note that FEDAC, Minibatch SGD, and Local SGD are all first-order algorithms, in that they use a first-order stochastic oracle. Thus, each time they observe a stochastic gradient (as per Assumption 1), the oracle uses a single unit of randomness (e.g., a single data point, when thinking in terms of a finite training dataset).

In contrast, our main theoretical method FEDSN proceeds via two possible options: for Case 1 as established in Algorithm 4, the algorithm queries both a stochastic gradient and a stochastic Hessian-vector product oracle at two independent samples, while in Case 2, we consider a different setting which allows the algorithm to observe, for any x, u, both a stochastic gradient and stochastic Hessian-vector product using the *same* random sample z. We may note that the final theoretical guarantee for Case 1 differs by only a small constant factor from that of Case 2, and our results as presented in Theorem 1 apply to both cases.

Thus, in order to maintain a fair comparison between these first-order methods and our practical method FEDSN-LITE, we have implemented FEDSN-LITE so that it also uses a single random sample (single data point), as outlined via Case 2 (`Same-Sample`) in Algorithm 4.

In addition to keeping the number of samples consistent, it is important to understand how both oracle models compare computationally. Clearly every oracle call for FEDSN-LITE is at least as expensive as a first-order stochastic oracle, as it subsumes the latter. Moreover, it is unclear a priori if the Hessian-vector product can be computed efficiently (say, in as much time as vector addition or multiplication). However, it turns out that the Hessian-vector product for logistic regression can be efficiently computed, since the Hessian matrix for a given sample is actually rank one (i.e., an outer product of known vectors). This is generally true for loss functions which belong to the family of *generalized linear models*. To see this, note that for loss functions of the form $F(x) = \sum_{i=1}^{N} \phi(b_i x^\top a_i)$, we have by a simple calculation that

$$\nabla_x F(x) = \sum_{i=1}^{N} \phi'(b_i x^\top a_i) b_i \cdot a_i \quad \text{and} \quad \nabla_x^2 F(x) := \nabla_x(\nabla_x F(x)) = \sum_{i=1}^{N} \phi''(b_i x^\top a_i) b_i^2 \cdot a_i a_i^\top,$$

and so for any $v \in \mathbb{R}^d$,

$$\nabla_x^2 F(x) v = \sum_{i=1}^{N} \phi''(b_i x^\top a_i) b_i^2 \cdot a_i \cdot (a_i^\top v),$$

which means each summand can be calculated in $\mathcal{O}(d)$ time. When maximizing the log-likelihood in the logistic regression model with labels in $\{-1, 1\}$, we need to minimize the following function,

$$F(x) = \sum_{i=1}^{N} \left(-b_i x^\top a_i + \ln\left(1 + \exp\left(b_i x^\top a_i\right)\right)\right),$$

which is an instance of the generalized linear models as considered above. Thus, in terms of vector operations, the Hessian-vector product oracle and the stochastic gradient oracle are *asymptotically similar* in the logistic regression model. We also note that for a general class of differentiable functions, Pearlmutter (1994) provides a technique to compute the Hessian-vector product using two passes of backpropagation (in the context of neural networks). For instance, we may consider a twice-differentiable function $F : \mathbb{R}^d \mapsto \mathbb{R}$ and note that for a vector $v \in \mathbb{R}^d$,

$$\nabla_x^2 F(x) v = \nabla_x \left(\nabla_x F(x)^\top v\right),$$

which can be obtained using two passes of backpropagation.

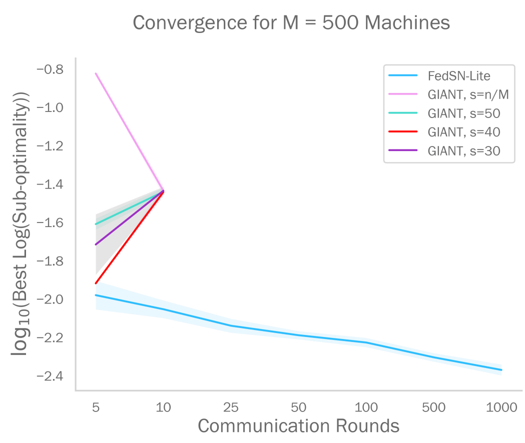
For the scale of our problem, it turns out that the difference in the number of vector operations is outweighed by other implementational overhead (for, e.g., loops, memory operations, etc.). In Table 5 we show the average per-step runtimes (over 250 runs) for three different algorithms for $M = 1, R = 1, K = 100000$.

Algorithm	**Avg. Runtime/Step** (in 10^{-5} sec.)	**Std. Dev. of Runtime/Step** (in 10^{-5} sec.)
FedSN-Lite	6.67	1.42
Local SGD	6.39	1.37
FedAc	7.48	1.58

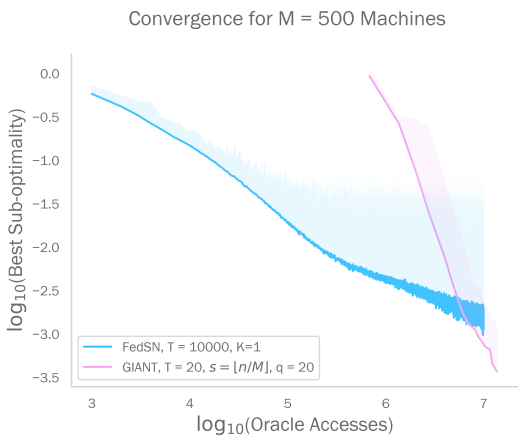
Table 5: Comparing the wall-clock runtimes of different algorithms in our implementation. We run every repetition of each algorithm as an individual job, so that the runs are *independent*, i.e., we don't introduce extraneous biases for any one algorithm (e.g., thermal throttling affecting the runtimes for a single algorithm). We run each algorithm 1700 times so that the 95% error margins for the average runtime/step estimate are within 1% of its value. Specifically, we calculate the empirical standard deviation σ of our average runtime/step estimate $\bar{x}$ by running it for $n = 1700$ runs. Then we report the error margin $\delta = z\sigma/\sqrt{n}$, where $z = 1.96$, is the z*-value from the standard normal distribution for a 95% confidence level. We ensure that $n = 1700$ is large enough, so that $\delta/\bar{x} \leq 0.01$. Finally we round $\bar{x}$ to two significant digits, respecting the 1% error margin. We also report the standard deviation estimate of the runtime/step so that its 95% error margins are within 4% of its value.

G.5 Comparison with GIANT

Recall that we are optimizing a convex function F, which is accessible only through stochastic oracles in an online fashion. To do so, we implement an inexact Newton method, where we reformulate the Newton step as the solution to a convex quadratic problem, $\min_{\Delta x} \frac{1}{2}\Delta x^\top \nabla^2 F(x) \Delta x + \nabla F(x)^\top \Delta x$. We then solve this problem using one-shot averaging, with access to stochastic gradient and Hessian-vector product oracles. If instead we were solving a batch problem, where we had a finite dataset, we could use it to determine the exact Newton step.



(a)



(b)

Figure 6: (a) Empirical comparison between GIANT and FEDSN-LITE, when equalizing for the number of data-accesses per communication round. We vary the frequency of communication (horizontal axis of each plot), while keeping the total number of steps on each machine (theoretical parallel runtime) fixed at $KR = 1000$ for FEDSN-LITE. We choose s, q such that $n/M + sq = K$ for GIANT. For $R > 10$, GIANT could not be run in this setup, as q was less than one. Note that this is the same reason why GIANT performs worse going from $K = 200$ to $K = 100$, as the value of q gets closer to one. We suspect that this worsening of performance does not happen for $s = \lfloor n/M \rfloor$, as in that case the effect of increasing Newton steps is more dominant than decreasing q. All experiments were repeated 20 times. (b) Empirical comparison between GIANT and FEDSN-LITE when keeping the number of oracle accesses equal. For both the methods we count the stochastic oracle access as well as the Hessian-vector product oracle access as two oracle accesses. The dark lines depict the minimum error obtained across multiple runs, and the error strips denote the maximum error across those runs.

More specifically, if $F(x) = \frac{1}{n}\sum_{i\in[n]} f(x; z_i)$, where z_i indexes an example in the training dataset, then we can obtain a sketch of the Hessian matrix $\nabla^2 \tilde{F}_m(x)$ on each machine, by distributing the examples across M machines. Then we can solve the linear system $\nabla^2 \tilde{F}_m(x) p_m = \nabla F(x)$ where $\nabla F(x)$ could be computed exactly and broadcasted to each machine. The linear system can be effectively solved with a conjugate gradient method. Finally the directions p_m can be aggregated to get an approximate Newton direction. This was precisely the idea proposed by Wang et al. (2018), in their algorithm called GIANT. In fact, there has been much work in the batch setting (Shamir et al., 2014; Zhang and Xiao, 2015; Reddi et al., 2016; Crane and Roosta, 2019; Islamov et al., 2021; Gupta et al., 2021) similar to this, but as we point out before, we work in a strictly more difficult setting. The ability to index the examples in the training set, and possibly re-use them in multiple rounds is commonly used in the batch setting, but it cannot be exploited in the online setting (except perhaps with an additional generalization error when $F(x) := \mathbb{E}_{z\sim\mathcal{D}}[f(x; z)]$).

Moreover, typically the computation per communication round for batch methods is much higher than our method. For instance if GIANT uses s examples on each machine for making a sketch of the Hessian and uses q steps of the conjugate gradient method, then since it still needs to compute the full gradient per communication round, the computation per round is $\propto n + Msq$, as opposed to KM for FEDSN-LITE. Nevertheless, we compared GIANT to our method for a range of values of s in Figure 6a. We note that in the sparse communication regime we beat GIANT. If communication is not a bottleneck, we can also compare both these methods while keeping the number of oracle accesses equal. In such a setting, as we note in Figure 6b, FEDSN-LITE has an advantage initially (when GIANT is not even applicable), but eventually GIANT's updates are more accurate, and it outperforms FEDSN-LITE. The bottom line is that these methods perform very similar updates, i.e., inexact Newton steps, but FEDSN-LITE is applicable in the online setting while GIANT is not. Moreover, FEDSN-LITE can have an advantage in the sparse communication regime.