IEEE CSS

Received 25 December 2023; revised 20 March 2024 and 3 April 2024; accepted 4 April 2024. Date of publication 15 April 2024; date of current version 2 May 2024. Recommended by Senior Editor Sonia Martinez.

Digital Object Identifier 10.1109/OJCSYS.2024.3388374

Novel Bounds for Incremental Hessian Estimation With Application to Zeroth-Order Federated Learning

ALESSIO MARITAN [1] (Student Member, IEEE), LUCA SCHENATO [1] (Fellow, IEEE), AND SUBHRAKANTI DEY [2] (Fellow, IEEE)

[1]Department of Information Engineering, University of Padova, 35131 Padova, Italy
[2]Department of Electrical Engineering, Uppsala University, SE-751 21 Uppsala, Sweden

CORRESPONDING AUTHOR: ALESSIO MARITAN (e-mail: alessio.maritan@phd.unipd.it).

The work of Alessio Maritan was supported in part by the Italian MUR under Grant DM352-2022, and in part by the Maschio Gaspardo. The work of Subhrakanti Dey was supported in part by the Swedish Research Council (VR) under Grant 2023-04232.

ABSTRACT The Hessian matrix conveys important information about the curvature, spectrum and partial derivatives of a function, and is required in a variety of tasks. However, computing the exact Hessian is prohibitively expensive for high-dimensional input spaces, and is just impossible in zeroth-order optimization, where the objective function is a black-box of which only input-output pairs are known. In this work we address this relevant problem by providing a rigorous analysis of an Hessian estimator available in the literature, allowing it to be used as a provably accurate replacement of the true Hessian matrix. The Hessian estimator is randomized and incremental, and its computation requires only point function evaluations. We provide non-asymptotic convergence bounds on the estimation error and derive the minimum number of function queries needed to achieve a desired accuracy with arbitrarily high probability. In the second part of the paper we show a practical application of our results, introducing a novel optimization algorithm suitable for non-convex and black-box federated learning. The algorithm only requires clients to evaluate their local functions at certain input points, and builds a sufficiently accurate estimate of the global Hessian matrix in a distributed way. The algorithm exploits inexact cubic regularization to escape saddle points and guarantees convergence with optimal iteration complexity and high probability. Numerical results show that the proposed algorithm outperforms the existing zeroth-order federated algorithms in both convex and non-convex problems. Furthermore, we achieve similar performance to state-of-the-art algorithms for federated convex optimization that use exact gradients and Hessian matrices.

INDEX TERMS Data privacy, estimation, federated learning, finite difference methods, optimization.

I. INTRODUCTION

Hessian estimation is a fundamental primitive with important implications in several machine leaning and engineering areas. For example, the Hessian matrix is a key ingredient to design optimization algorithms that achieve superlinear or quadratic convergence. These algorithms use the Hessian to either reshape the gradient or build a local function approximation, and this allows to reach the optimal solution of the optimization problem in few iterations. However, computing the exact Hessian is onerous, and especially for high-dimensional problems it is often necessary to fall back on cheaper Hessian estimates. Moreover, in many relevant cases such as black-box optimization and simulation-based optimization, it is not even possible to explicitly compute the Hessian matrix, as the objective function is accessible only through point evaluations. In the literature, this scenario is referred to as zeroth-order optimization, and it is assumed that function queries are expensive and possibly constrained in number.

The ideal Hessian estimator should possess all the following desirable qualities: *(i)* To be compatible with black-box functions, the estimator should not require exact gradients for its computation, but only a limited number of function values. *(ii)* To be computationally efficient, the estimator should

incorporate past estimates and not forget previously collected information. *(iii)* To be employed by distributed optimization algorithms, the estimator should be parallelizable.

The Hessian estimator analyzed in this work satisfies all the above conditions, and in particular is well suited for federated learning, a kind of distributed machine learning where clients are connected to a central server in a star topology. Federated learning has gained significant attention from both academia and industry, as it allows to collaboratively train a shared model over the data islands owned by the various participants, while preserving the individual privacy. To satisfy the latter property, transmitting raw data samples is forbidden, and the communications typically consist of model updates. Data exchange between clients is costly and possibly limited or unreliable, for example due to bandwidth restrictions or connection instabilities. This calls for communication-efficient algorithms that converge in few iterations, and the Hessian estimator analyzed in this work is a useful building block to design them.

A. PRELIMINARIES AND RELATED WORK

Below we provide an overview of the state of the art in some research topics related to our work.

Zeroth-order optimization: Zeroth-order optimization allows tackling learning problems where the derivatives of the objective function are not available or difficult to compute. Problems that fall inside this category include policy learning for reinforcement learning, adversarial training and generating explanations for black-box models [1]. Zeroth-order methods can also replace backpropagation for neural networks that are not entirely differentiable, and constitute an alternative to the recent Forward-Forward algorithm [2]. Zeroth-order optimization only requires to evaluate the objective at certain input points, and estimates function derivatives by means of finite-differences. However, even function queries are assumed to be expensive and possibly budgeted in number, which calls for efficient algorithms that can get the most out of them. Alternatives to zeroth-order optimization include derivative-free optimization, Bayesian optimization and genetic algorithms.

Zeroth-order federated learning: Since the in second half of this work we present a novel zeroth-order federated learning, we briefly introduce the main federated zeroth-order algorithms available in the literature: *(i)* FedZO [3] replaces the exact gradient in FedAvg with a stochastic gradient estimator based on forward finite-differences. *(ii)* ZONE-S [4] activates only one client per iteration, and minimizes an augmented Lagrangian function at the server. *(iii)* ZooPFL [5] considers the case where all clients own the same black-box model that cannot be changed, and uses trainable local encoders and linear projections to respectively remap the input and output of the frozen model. *(iv)* FZooS [6] tackles the problem of query inefficiency, but relies on the strong assumption that every local function is sampled from a Gaussian process. *(v)* BAFFLE [7] uses shared random seeds to sample common query points at all clients, and estimates the global gradient using Stein's identity. *(vi)* FedZeN [8] estimates also the Hessian of the global objective in an incremental fashion, and applies a quasi-Newton method at the server to address convex optimization problems. It is worth noting that FedZeN is the only algorithm that takes advantage of second-order derivative information, while the others rely only on gradient estimates.

Zeroth-order Hessian estimation: We quickly review the main techniques to approximate a $d \times d$ Hessian matrix by means of finite-differences. Reference [9] contains several deterministic estimation schemes and stencils that approximate the $d(d+1)/2$ distinct entries of the Hessian once at the time, resulting in $O(d^2)$ overall function queries. To reduce the computational burden of the estimation process, some works choose to approximate only some parts of the matrix. For example, ZO-JADE [10] estimates only the main diagonal of the Hessian by means of central-differences along the canonical basis at the cost of $2d+1$ function evaluations. Alternatively one can resort to stochastic estimation schemes, where the perturbation vectors used to choose the query points are randomly generated. While deterministic estimators usually provide the golden standard, randomized ones allow to trade-off accuracy and computational complexity. For example, [11] performs $4r^2$ function queries along orthogonal directions sampled from the Stiefel manifold, where r is a design parameter. Reference [12] proposes a set of estimators based on second-order Stein's identity to approximate the Hessian of a Gaussian-smoothed objective function. Both [13] and [14] propose randomized zeroth-order Hessian estimators for functions defined over Riemannian manifolds. All the above estimators start from scratch at each new estimation, and miss out on the possibility of exploiting past information to reduce the number of function queries. A zeroth-order version of the popular BFGS method, that updates the previous Hessian estimate according to the secant equation using only gradient information, is presented in [15]. In [16] they propose an incremental Hessian estimator whose squared error norm converges linearly in expectation, but they do not take into account the error due to zeroth-order estimation. Reference [8] explains how to efficiently compute the last estimator in a distributed way, and empirically shows its superiority with respect to other Hessian approximation techniques. In the first half of this paper we provide novel non-asymptotic bounds for the incremental estimator in [16], where we properly take into account the zeroth-order estimation error and derive the minimum number of function queries needed to achieve a given accuracy.

Hessian-based federated algorithms: The most effective way to reach a function minimum in few iterations is to leverage the curvature of the objective function, which is described by the Hessian matrix. Below we list the main second-order federated algorithms that take advantage of the Hessian. *(i)* GIANT [17] addresses convex optimization problems by averaging the local Newton directions of the clients. This is equivalent to approximating the Hessian of the global objective with the harmonic mean of the local Hessian matrices. *(ii)* FedNL and its variants [18] are designed for

convex optimization and make clients transmit to the server compressed innovations of their local Hessian matrices and the corresponding compression errors. *(iii)* In SHED [18] clients compute the eigendecomposition of their local Hessian only when required, and share some of the most relevant eigenvalue-eigenvector pairs with the server to incrementally update the Hessian approximation. *(iv)* FLECS [20] extends FedNL to non-convex problems by changing the Hessian update rule and the computation of the approximate Newton direction.

We remark that all the above algorithm make use of exact derivatives and require clients to compute their local Hessian matrix. In particular, none of them allows approximate Hessian matrices, which makes impossible to directly implement them as zeroth-order algorithms. The only Hessian-aware federated algorithm which is also zeroth-order is FedZeN [8], but it is limited to convex problems.

B. CONTRIBUTION AND ORGANIZATION

The contribution of this paper is twofold:

1) We provide a number of novel results regarding the properties of the incremental Hessian estimator proposed in [16]. Our analysis is profoundly different and more in-depth with respect to the original paper. First, we focus on a practical zeroth-order implementation of the estimator suitable for black-box optimization. We take into account the approximation error on the directional curvature due to finite-differences, which is neglected in [16]. Second, to fully exploit the advantage of incremental estimation, we consider the case of a time-varying Hessian to be tracked. Finally, we derive non-asymptotic bounds on the convergence of the estimation error and on the minimum number of function queries needed to achieve a given accuracy with the desired probability. The above analysis is non-trivial and involves mathematical tools that are not present in [16]. Moreover, we empirically show that $O(d)$ function queries are sufficient to obtain an useful estimate of a $d \times d$ Hessian, whereas standard entry-wise estimators necessarily require $O(d^2)$ queries. Our bounds make the incremental Hessian estimator actually reliable, allowing it to be used whenever a cheap but sufficiently accurate approximation of a Hessian is needed.
2) We present a novel optimization algorithm as a practical application of the incremental Hessian estimator discussed at item 1). The algorithm, named FedZCR, is the first zeroth-order algorithm for non-convex federated optimization to estimate and exploit the Hessian of the objective. The Hessian estimator is built in the federated setting following the distributed approach of [8]. In particular, our algorithm can be seen as an extension of [8] to the non-convex setting, and further improves over [8] by replacing a simplifying assumption with rigorous conditions. The Hessian estimate is combined with cubic regularization at the server to escape saddle points and converge with arbitrarily high probability and accuracy. More specifically, the algorithm is guaranteed to converge to a second-order $(\tau, \sqrt{\tau})$-optimal point in $O(\tau^{-3/2})$ iterations with high probability, which is currently the best iteration complexity for non-convex problems. We also propose the adaptive version FedZACR, and the use of subsampling to decrease the total computational cost and accommodate client heterogeneity. Our numerical results show that FedZACR outperforms the existing zeroth-order federated methods, and is comparable to second-order methods for convex optimization that use the exact derivatives.

The organization of this work reflects the duality of our contributions: in the first half of the paper (Section II) we derive theoretical bounds for the Hessian estimator, while in the second half (Section III) we present a novel algorithm for non-convex zeroth-order federated learning. Finally, Section IV is dedicated to numerical simulations.

II. INCREMENTAL HESSIAN ESTIMATION

The first part of this work is dedicated to the analysis an Hessian estimator, that can be used to relieve the computational burden of computing exact Hessian matrices. We consider an estimator that can be employed also when the target function is only accessible by means of point evaluations, and that can be easily adapted to perform distributed estimation. Our goal is to derive two kind of bounds: *(i)* upper bounds on the approximation error with respect to the computational cost, measured by the number of function queries, and *(ii)* lower bounds on the number of function evaluations needed to achieve a given accuracy. These bounds are especially relevant in zeroth-order optimization, where the common assumption is that function evaluations are expensive, possibly budgeted in number, and come with a significant computational cost. Choosing an estimator that is parsimonious with function queries and does not represent a bottleneck is therefore crucial to the efficiency of the application that uses it. With the above premise, one cannot resort to standard entry-wise estimators based on finite-differences along the canonical basis, that involve $O(d^2)$ queries to estimate an Hessian $\in \mathbb{R}^{d\times d}$ [19].

Notation: We denote with I_d and 0_d the d-dimensional identity matrix and the $d \times d$ matrix of all zeros, respectively. We use the concise notation $[n]$ to indicate the set of integers $\{1, \dots, n\}$. We use $\mathcal{U}(\mathbb{S})$ to denote the uniform distribution on the unit sphere $\mathbb{S}^{d-1} = \{z \in \mathbb{R}^d \text{ s.t. } \|z\| = 1\}$. The third order derivative tensor of a function f evaluated at x is denoted by $(D^3 f)(x)$. Given a matrix, $\|\cdot\|$ is the spectral norm while $\|\cdot\|_F$ is the Frobenius norm.

A. BOUNDS FOR THE HESSIAN ESTIMATOR

We start by introducing the Hessian estimator proposed in [16], based on the update

$$H^j = H^{j-1} + \left(u_j^T(\nabla^2 f(x) - H^{j-1})u_j\right)(u_j u_j^T), \quad (1)$$

where the vector $u_j \in \mathbb{R}^d$ is sampled from the uniform distribution on the unit sphere. If H^{j-1} is a symmetric matrix, then H^j is also symmetric. In [16] it is proved that repeatedly applying (1) one converges asymptotically to the exact Hessian almost surely, and that at each iteration the estimator satisfies

$$\mathbb{E}\left[e\left(H^j\right)^2\right] \leq \left(1 - \frac{2}{d(d+2)}\right) e\left(H^{j-1}\right)^2 \tag{2}$$

where $e(\star) := \| \star - \nabla^2 f(x)\|_F$ is the Frobenius norm of the estimation error. Since the exact second-order directional derivatives are not available, we approximate them using central-differences as also done in [8]. Given a small positive scalar μ and $u \sim \mathcal{U}(\mathbb{S})$, the zeroth-order counterpart of (1) can be defined as

$$\hat{H}^j = \hat{H}^{j-1} + \left(\frac{f(x+\mu u_j) - 2f(x) + f(x-\mu u_j)}{\mu^2} - u_j^T \hat{H}^{j-1} u_j\right)(u_j u_j^T). \tag{3}$$

This makes the incremental estimation formula implementable also when the objective function is a black-box, i.e. only input-output pairs are available. It is empirically shown in [8] that the update (3) provides significantly better performance with respect to other randomized zeroth-order Hessian approximations, including estimators based on averages along orthonormal directions or on the second-order Stein's identity. However, no theoretical guarantees are currently available for (3), limiting its applicability and preventing rigorous convergence analyses. In this section we address this problem, providing non-asymptotic bounds and practically implementable conditions on the number of search directions required to attain a given accuracy. All the proofs are contained in Appendix A.

To analyse the estimator we first need to characterize the objective function with the following assumption.

Assumption 1 (Smoothness): Let the global cost f be three times continuously differentiable with Lipschitz continuous derivatives, i.e. there exist positive constants L_0, L_1, L_2, L_3 such that $\forall x, y \in \mathbb{R}^d$

$$\begin{aligned} \|f(x) - f(y)\| &\leq L_0 \|x - y\|, \\ \|\nabla f(x) - \nabla f(y)\| &\leq L_1 \|x - y\|, \\ \left\|\nabla^2 f(x) - \nabla^2 f(y)\right\| &\leq L_2 \|x - y\|, \\ \left\|\left(D^3 f\right)(x) - \left(D^3 f\right)(y)\right\| &\leq L_3 \|x - y\|. \end{aligned}$$

We recall that if $\partial^p f(x)$ is L_p-smooth, then $\|\partial^{p+1} f(x)\| \leq L_p \ \forall x \in \mathbb{R}^d$. For example, Assumption 1 implies that $\|\nabla^2 f(x)\| \leq L_1 \ \forall x \in \mathbb{R}^d$.

We begin by bounding the expected improvement in accuracy provided by the exact update (1). Since the Hessian estimator is incremental, to follow its evolution we need a convergence rate that can be applied recursively. The convergence rate provided in [16] considers the squared norm of the approximation error, while here we look for a bound on the non-squared norm, that is needed for subsequent analysis. This bound can be obtained by choosing $S = \nabla^2 f(x) - H^{j-1}$ in the following Lemma 1.

Lemma 1 (Convergence rate of the update (1)): Let $S \in \mathbb{R}^{d\times d}$ be a symmetric matrix and $u \sim \mathcal{U}(\mathbb{S})$. Then we have

$$\mathbb{E}\left[\left\|S - (u^T S u)uu^T\right\|_F\right] \leq \eta \|S\|_F, \ \eta = \sqrt{1 - \frac{3}{d(d+2)}}.$$

Remark 1 (Alternative convergence rate): The proof of Lemma 1 is quite involved, but it is possible to obtain a convergence rate slightly worse than the one above in a much simpler way. Indeed, applying moment monotonicity to (2) we get

$$\begin{aligned} \mathbb{E}\left[\left\|S - (u^T S u)uu^T\right\|_F\right] &\leq \left(\mathbb{E}\left[\left\|S - (u^T S u)uu^T\right\|_F^2\right]\right)^{1/2} \\ &\leq \left(\left(1 - \frac{2}{d(d+2)}\right)\|S\|_F^2\right)^{1/2} \\ &= \left(1 - \frac{2}{d(d+2)}\right)^{1/2} \|S\|_F. \end{aligned}$$

We now address the error due to the zeroth-order approximation, which is neglected in [16] and not quantified in [8], that affects the convergence rate as shown below.

Lemma 2 (Zeroth-order error of the update (3)): Consider the updates (1) and (3) with $u \sim \mathcal{U}(\mathbb{S})$ and fix $H^{j-1} = \hat{H}^{j-1}$. Then for both the spectral and the Frobenius norm it holds

$$\mathbb{E}\left[\left\|H^j - \hat{H}^j\right\|\right] \leq \frac{\mu^2 L_3}{12\, d}.$$

As a consequence, the convergence rate of (3) towards the exact Hessian is

$$\mathbb{E}\left[\left\|\nabla^2 f(x) - \hat{H}^j\right\|_F\right] \leq \eta \left\|\nabla^2 f(x) - \hat{H}^{j-1}\right\|_F + \frac{\mu^2 L_3}{12\, d}.$$

We denote with $\hat{H}^r$ the matrix obtained applying r times the incremental formula (3) starting from an arbitrary symmetric $\hat{H}^0 \in \mathbb{R}^{d\times d}$. We aim to bound the approximation error $\|\nabla^2 f(x) - \hat{H}^r\|$ as a function of the design parameter r, which determines the number of function evaluations required to build $\hat{H}^r$, namely $2r + 1$. Due to the randomized nature of the estimator, it is not possible to get a deterministic bound on the error or a certain convergence rate. The best we can obtain is a bound guaranteed to hold with high probability, that can be provided by e.g. Markov's or Chebyshev's inequalities. In our case we resort to Markov's inequality, stating that the probability that a non-negative random variable $z \in \mathbb{R}_+$ is larger than an arbitrary scalar ϵ is

$$\mathbb{P}(z \geq \epsilon) \leq \frac{\mathbb{E}[z]}{\epsilon} \quad \forall \epsilon > 0. \tag{4}$$

In our case $z = \|\nabla^2 f(x) - \hat{H}^r\|$, and building on the previous two lemmas we derive a bound on the expected value of

this quantity. Since the bound takes into account the history of the estimator, it depends on the initialization of the latter.

Lemma 3 (Convergence of the zeroth-order Hessian estimator (3)): The expected convergence rate of the estimation error of $\hat{H}^r$, obtained updating an initial symmetric $\hat{H}^0 \in \mathbb{R}^{d\times d}$ according to (3), is

$$\mathbb{E}\left[e\left(\hat{H}^r\right)\middle|\,\hat{H}^0\right] \leq \eta^r e\left(\hat{H}^0\right) + \frac{\mu^2 L_3}{12\,d}\sum_{i=0}^{r-1}\eta^i.$$

Remark 2 (Design parameters μ and r): The bound of Lemma 3 tells us that the approximation error of the Hessian estimator can be made arbitrarily small by tuning the design parameters μ and r. In particular, by increasing the number of search directions, the first term in the right-hand side goes to zero exponentially fast while the second term grows. However, since μ can be chosen arbitrarily small up to machine precision, one can make also the second term negligible.

We are now in a position to apply Markov's inequality, that provides sufficient conditions for the norm of the estimation error to be arbitrarily small with high probability. This requires to choose the maximum tolerable error ϵ and the maximum probability δ that the bound does not hold. Once these parameters are fixed, the dependent variable is the number of updates r, which in turn determines the number of required function evaluations.

Our analysis addresses the general case in which the decision vector x and the corresponding Hessian to be approximated can change over time. In particular, from now on we use the subscript $k \geq 1$ to denote the value of a variable at the k-th iteration of a generic optimization algorithm. Moreover, we allow for a different number of search directions $r(k)$ at each iteration.

Theorem 1 (Sufficient number of random directions): Consider a sequence of Hessian matrices of a function $f(x)$ satisfying Assumption 1. Let $\hat{H}_k^{r(k)}$ be the estimator of the k-th Hessian, obtained updating $\hat{H}_k^0$ along $r(k)$ directions according to (3). Arbitrarily choose an accuracy $\epsilon > 0$ and a failure probability $\delta \in (0, 1)$.

• (Memory-less case) Let $\hat{H}_k^0$ be the matrix of all zeros and

$$\mu \leq \sqrt{\frac{12\,d(1-\eta)}{L_3}}\min\left\{\epsilon\delta,\ \sqrt{d}L_1\right\},$$

$$r(k) \geq \log_\eta\left(\frac{\epsilon\delta(1-\eta) - \frac{\mu^2 L_3}{12\,d}}{\sqrt{d}L_1(1-\eta) - \frac{\mu^2 L_3}{12\,d}}\right),$$

then

$$\mathbb{P}\left(\left\|\nabla^2 f(x_k) - \hat{H}_k^{r(k)}\right\|_F \geq \epsilon\middle|\,\hat{H}_k^0 = 0\right) \leq \delta.$$

Recall that η, defined in Lemma 1, is a function of d. The asymptotic behaviour of μ and $r(k)$ for $d \to \infty$ as a function of only d, ϵ, δ is the following: the upper bound on μ is $O(\epsilon\delta/\sqrt{d})$, the lower bound on r is $O(d^2\log(\sqrt{d}/\epsilon\delta))$.

• (Warm-start case) Arbitrarily choose an initial point x_1 and an initial symmetric Hessian estimator $\hat{H}_1^0$. Let $\hat{H}_k^{r(k)}$ be the Hessian estimator obtained setting $\hat{H}_i^0 = \hat{H}_{i-1}^{r(i-1)}\ \forall i > 1$, and updating $\hat{H}_i^0$ along $r(i)$ directions according to (3) $\forall i \in [k]$. If

$$\mu \leq \sqrt{\frac{12\,d\epsilon\delta(1-\eta)}{L_3}},\ r(k) \geq \log_\eta\left(\frac{\epsilon\delta - \frac{1}{1-\eta}\frac{\mu^2 L_3}{12\,d}}{c_k}\right),$$

$$c_k = \frac{\mu^2 L_3}{12\,d}\left[\sum_{j=1}^{k-1}\left(\eta^{\sum_{z=j+1}^{k-1} r(z)}\frac{1-\eta^{r(j)}}{1-\eta}\right) - \frac{1}{1-\eta}\right]$$
$$+ \sqrt{d}L_2\sum_{j=2}^{k}\left(\eta^{\sum_{z=j}^{k-1} r(z)}\left\|x_j - x_{j-1}\right\|\right)$$
$$+ \eta^{\sum_{j=1}^{k-1} r(j)}\left\|\nabla^2 f(x_1) - \hat{H}_1^0\right\|_F,$$

then

$$\mathbb{P}\left(\left\|\nabla^2 f(x_k) - \hat{H}_k^{r(k)}\right\|_F \geq \epsilon\middle|\,\hat{H}_1^0\right) \leq \delta.$$

Remark 3: The update $\hat{H}_k^0 = \hat{H}_{k-1}^{r(k-1)}$ can be extremely advantageous when the Hessian matrix is slowly changing, e.g. when the algorithm is making small steps or is approaching the minimum. When the Hessian is known to be constant, as it happens for linear least squares and ridge regression problems, one can also refer to the memory-less case and estimate the Hessian only once.

The asymptotic scaling $r \geq O(d^2\log(\sqrt{d}/\epsilon\delta))$ shows that the number of search directions prescribed by Theorem 1 may grow very large for high-dimensional problems or small values of the product $\epsilon\delta$. However, this is not a concern because of the following two reasons. First, the lower bound on r considers the worst case scenario and results from conservative approximations and Taylor expansions. In particular, the term $\log(\sqrt{d})$ comes from initializing the Hessian estimator with the zero matrix, while typically one chooses a better initial guess. For example, when tracking a time-varying Hessian it is natural to warm-start the estimator starting from the estimate obtained at the previous time step. The second and most important argument is that the incremental Hessian estimator is meant to be used while keeping $r \ll d^2$, as building $\hat{H}^r$ involves $2r+1$ function values. In fact, the purpose of (3) is not to achieve the lowest possible approximation error, but to provide a cheap Hessian estimate in cases where obtaining $O(d^2)$ function values is infeasible, which often happens in practice. Indeed, if $O(d^2)$ function evaluations could be afforded one could simply employ a deterministic entry-wise estimator, which typically also provides better estimates for the same number of function queries. Our experiments in Section IV show that setting $r = O(d)$, $r \geq d$ (for example, we use $r = d$ in the numerical experiment in Fig. 6) is sufficient to obtain very good results.

Remark 4: The incremental randomized procedure allows an arbitrary small number of function queries, provides an improved estimate at each update and can be stopped at any moment. Moreover, it allows to track a time-varying Hessian

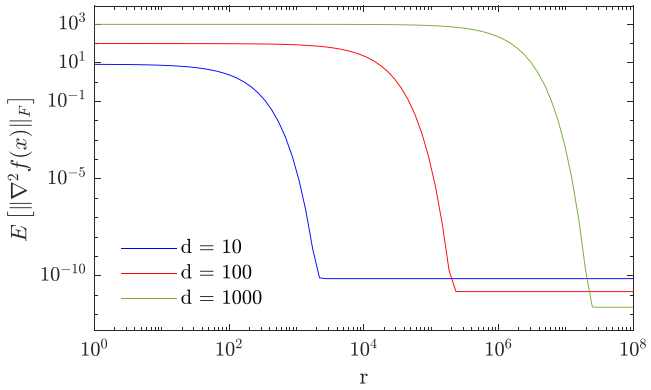

FIGURE 1. Visual representation of the bound provided by Lemma 3, setting $\hat{H}^0 = 0_d$, $L_3 = 0.05$, $\mu = 10^{-4}$. The Hessian matrices are random matrices whose elements are i.i.d. $\sim \mathcal{N}(0, 1)$.

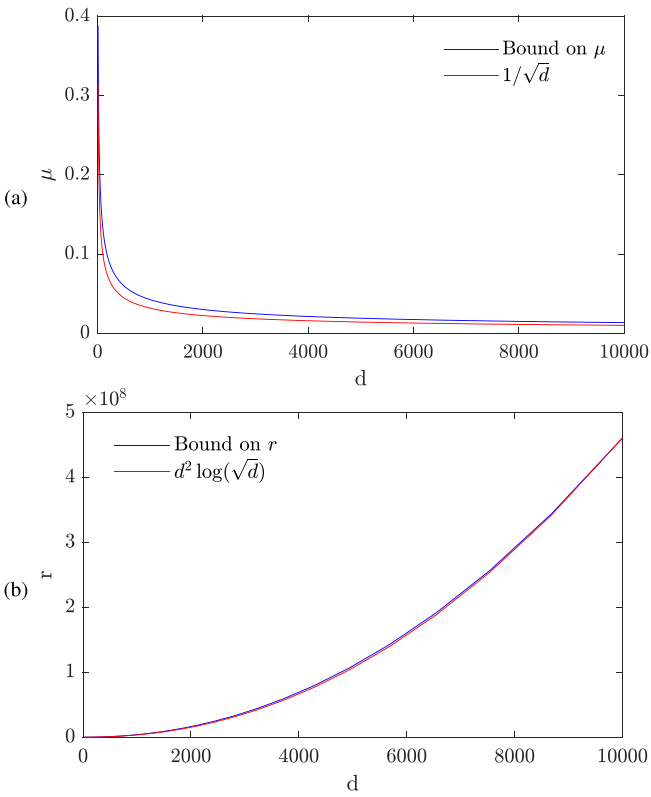

FIGURE 2. Plot of the upper bound on μ and the lower bound on r for various values of d, according to the memory-less case of Theorem 1. The blue lines have been obtained setting $\epsilon = 0.5$, $\delta = 0.01$, $L_1 = L_3 = 0.05$, and the bound on r in (b) is computed using the realistic value $\mu = 10^{-4}$. The red lines show the asymptotic behaviour for large values of d.

using the previous estimate as starting point. In comparison, deterministic entry-wise estimators always start from scratch and necessarily require $O(d^2)$ function evaluations to approximate the $d(d+1)/2$ distinct entries of the Hessian. A possible alternative to keep the number of function queries linear in d is given by the Jacobi estimator employed in [10], that estimates only the diagonal of the Hessian by means of finite-differences along the canonical basis. However, neglecting the off-diagonal entries of the Hessian leads to high approximation errors in case of skewed objectives.

B. GRADIENT ESTIMATION AND IMPROVED HESSIAN ESTIMATION VIA ORTHONORMAL SEARCH DIRECTIONS

The incremental Hessian estimator requires to uniformly sample a set of vectors from the unit sphere. This is commonly done by normalizing random vectors $\sim \mathcal{N}(0, I_d)$, possibly leading to a set directions not evenly distributed over the search space [8]. Intuitively, a simple way to efficiently span the neighborhood of the decision vector and minimize redundancy is to choose orthogonal search directions. Since $r \geq d$, we can only ask for partial orthogonality, where each direction is orthogonal to at most other $d-1$ directions. In practice, we generate $\lceil r/d \rceil$ orthonormal bases of $\mathbb{R}^d$, merge them and select the first r vectors. As noted in [8], empirically this approach provides concrete advantages over using totally random directions, allowing to achieve a given accuracy with a smaller number of search directions. In Appendix B we provide two methods to easily generate orthonormal vectors whose marginal distribution is $\mathcal{U}(\mathbb{S})$. These methods can be used to build the $\lceil r/d \rceil$ bases of $\mathbb{R}^d$ from which to sample the r search directions for the Hessian estimator.

Picking the search directions from the union of multiple orthonormal bases can improve also gradient estimation. Indeed, it opens up the possibility to use the gradient estimator

$$\hat{g} = \sum_{j=1}^{d} \frac{f(x+\mu u_j) - f(x-\mu u_j)}{2\mu} u_j, \tag{5}$$

where $\{u_1, \ldots, u_d\}$ is an orthonormal basis, which can be thought as a rotated version of the standard entry-wise estimator that computes finite-differences along the canonical basis. Numerical simulations show that typically the Hessian estimator (3) needs $r \geq d$ search directions to provide an acceptable estimate. This is common to most randomized Hessian estimators, see for example [11] and [12]. Therefore, by picking an orthonormal set $\{u_1, \ldots, u_d\}$ from the $r \geq d$ directions used to build the Hessian estimator, no additional function queries are needed and the gradient estimate is obtained for free. As shown in [8] of this estimator satisfies the deterministic bound

$$\|\nabla f(x) - \hat{g}\| \leq \frac{dL_2\mu^2}{6}, \tag{6}$$

which implies that for sufficiently small values of μ the estimate is almost perfect. In practice (5) is very reliable, and outperforms other randomized gradient estimators even for $r \gg d$. This is in agreement with [20], who claims that randomized gradient estimators offer no theoretical or practical advantage over the standard ones when $r \geq d$.

III. APPLICATION TO FEDERATED LEARNING

In this second part of the paper we propose a concrete application of the bounds developed so far. We first show how the incremental Hessian estimator can be efficiently built in

a distributed way, and then we describe a novel zeroth-order algorithm for non-convex federated optimization of black-box functions. While in what follows we focus on zeroth-order federated learning, we remark that the bounds in Section II can be used whenever an Hessian estimate with bounded approximation error is needed, i.e. also in centralized or fully-distributed optimization settings.

A. PROBLEM FORMULATION

We address the horizontal federated learning setting, where a set of clients are connected to a central server in a star topology. The objective is to collaboratively train a shared model to achieve higher accuracy and generalization capabilities, while preserving the individual privacy. Clients own disjoint sets of private data samples belonging to the same feature space, and the distribution of the local data can vary across participants. We consider the empirical risk minimization problem

$$f(x^\star) = \min_{x\in\mathbb{R}^d} \left\{ f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x) \right\}. \tag{7}$$

where $f_i(x)$ is the loss function of participant i. The local functions take the form

$$f_i(x) = \frac{1}{m_i}\sum_{j=1}^{m_i} f_{ij}(x). \tag{8}$$

where $f_{ij}(x)$ is the loss of the j-th data sample stored by client i, who owns m_i data samples. We denote the total number of samples available for training with $m = \sum_{i=1}^{n} m_i$. The exact function derivatives may be not available or be prohibitively expensive to compute, and any function may be accessible only through point evaluations. Also function queries are assumed to be costly and time-consuming, and their number should be limited as much as possible.

To devise an algorithm that can work with limited communication bandwidth, we require that all the devices that take part to the training are equipped with a pseudo-random number generator (PRNG). This requirement is reasonable and barely restrictive, and can be found also in [7], [8] and [20].

Assumption 2 (PRNGs): All the clients and the central server have access to a pseudo-random number generator, whose output sequence is repeatable and is uniquely determined by the internal seed.

B. FEDERATED GRADIENT AND HESSIAN ESTIMATION

The zeroth-order gradient and Hessian estimators presented in Section II can be built in a distributed way by following the procedure described in [8]. Consider the federated scenario introduced in the previous section, namely a set of clients connected to a central server according to a star topology. The key idea is to leverage Assumption 2 to generate a common set of search direction at all nodes, removing the communication cost associated with broadcasting the directions. Below we summarize the distributed estimation procedure, as it will be employed also by the federated algorithm proposed in the next subsection.

During the initialization phase, the central server sends to all the clients the same seed, possibly using a private transmission channel, and the clients use it to synchronize their pseudo-random number generator. Suppose that we want to estimate the global gradient and Hessian at the current decision vector x_k using (3) and (5). To do so, the clients and the server agree on a sampling procedure (one of the two described in Appendix B) and execute it $\lceil r/d \rceil$ times, using the synchronized PRNG whenever it is needed to generate random quantities. As a result, they all get the same $\lceil r/d \rceil$ bases of $\mathbb{R}^d$, each composed of orthonormal vectors whose marginal distribution is $\mathcal{U}(\mathbb{S})$. Selecting the first r vectors from the union of these bases ensures that all nodes come up with the same set of search directions $\{u_1, \dots, u_r\}$. Each client i evaluates his local function at the points x_k and $\{x_k \pm \mu u_j\}$, $j \in [r]$ to compute the d gradient coefficients

$$c_{ij} = \frac{f_i(x_k + \mu u_j) - f_i(x_k - \mu u_j)}{2\mu} \approx \nabla f_i(x_k)^T u_j, \tag{9}$$

and the r directional curvatures

$$b_{ij} = \frac{f_i(x_k + \mu u_j) - 2f_i(x_k) + f_i(x_k - \mu u_j)}{\mu^2}. \tag{10}$$

The server receives these scalars and computes the average over the set of clients. Since the search directions are the same for all nodes, the central server can build the gradient estimator and update the current Hessian estimate $\hat{H}_k^0$.

$$g_k = \sum_{j=1}^{d}\left(\frac{1}{n}\sum_{i=1}^{n} c_{ij}\right) u_j, \tag{11}$$

$$H_k^j = H_k^{j-1} + \left(\frac{1}{n}\sum_{i=1}^{n} b_{ij} - u_j^T H_k^{j-1} u_j\right) u_j u_j^T, \quad j \in [r].$$

Remark 5: This distributed estimation of the Hessian is advantageous also in case of non-zeroth-order federated optimization to build an Hessian estimate at the server in a communication-efficient way. Indeed, if clients are able compute the exact Hessian of their local function, they can simply send to the server the exact directional curvatures, i.e. $b_{ij} = u_j^T \nabla^2 f_i(x_k) u_j$. If clients transmitted to the server all distinct entries of the exact Hessian, this would result in sending $d(d+1)/2$ scalars. Instead, with this procedure each client transmits to the server only r scalars, where r is a tunable parameter.

As noted in [8], the above procedure has two properties that are relevant for federated learning. First, since it estimates the derivative of global objective, it allows for clients with heterogeneous data distributions. Second, if in the initialization phase the seed is shared through a private channel, then the procedure conceals the estimated derivatives from external eavesdroppers, who can only obtain the coefficients $\{b_{ij}, c_{ij}\}$ which by themselves do not leak any information.

In case of Big Data applications it may not be feasible to compute the derivative estimators using all the data samples. To address these scenarios, in Appendix C we generalize the

bounds on the estimation error to include the cases where subsampling is used.

Remark 6: The aforementioned procedure was first introduced by [8], that proposes the federated zeroth-order algorithm for convex optimization FedZeN. In the theoretical analysis of the latter, the approximation error the Hessian estimator is bounded using a simplifying assumption that involves unknown quantities. Using the bounds developed in Section II, it is possible to remove that assumption and get a lower bound on the number of search directions needed by FedZeN to achieve local quadratic convergence.

C. CUBIC-REGULARIZED FEDERATED LEARNING

We now show how our analysis regarding the incremental Hessian estimator can be applied to design provably convergent algorithms for non-convex federated optimization. Building over the previous sections, we design a novel federated algorithm that can be applied to non-convex functions whose exact derivatives are not available. The algorithm arises from the union of the following three elements. First, the procedure described in Section III-B is used to estimate the derivatives of the global objective at the central server. Second, the bounds derived in Section II are used to tune the number of search directions and make the Hessian estimator sufficiently accurate. Finally, the estimated derivatives are used to solve a cubic-regularized subproblem, allowing to optimize non-convex objectives with solid convergence guarantees.

1) CENTRALIZED CUBIC-REGULARIZED NEWTON

Non-convex optimization problems are typically characterized by multiple local minima and saddle points, which require additional care. For this reason, in our algorithm we employ the cubic-regularized Newton method introduced by [21], that for general non-convex optimization problems ensures fast convergence to approximate second-order stationary points, i.e. approximate local minima, defined below.

Definition 1 ((ϵ_g, ϵ_H)-optimality): Given $\epsilon_g > 0$ and $\epsilon_H < 1$, x is an (ϵ_g, ϵ_H)-optimal point of the function f if

$$\|\nabla f(x)\| \leq \epsilon_g, \quad \lambda_{\min}(\nabla^2 f(x)) \geq -\epsilon_H,$$

where the operator $\lambda_{\min}(\star)$ returns the smallest eigenvalue of a matrix argument.

More specifically, under the assumption of L_2-Lipschitz Hessian, the cubic-regularized Newton method provides monotone convergence to an $(\tau, \sqrt{\tau})$-optimal point in $O(\tau^{-3/2})$ iterations, which is the best known worst-case iteration complexity.

The key feature of this method is the capability of escaping saddle points that are non-degenerate, i.e. for which the Hessian is full-rank. As argued in [22], in general most critical points are saddles, and statistically the number of saddles grows exponentially with the size of the problem. Therefore, cubic-regularized Newton provides possibly better solutions compared to other optimization algorithms such as stochastic gradient descent, that may get stuck in saddle points and make the loss function plateau [22].

Remark 7: In alternative to cubic-regularized Newton method, one can use Approximate saddle-free Newton method [22], that takes the absolute value of the eigenvalues of the Hessian to avoid getting trapped in saddle points.

At each iteration the method updates the decision vector using $x_{k+1} = x_k + s_k$, where s_k is the step that minimizes the model below, composed by the second-order Taylor expansion of the objective plus a cubic regularization term with coefficient $M > 0$:

$$s_k = \text{argmin}_{s \in \mathbb{R}^d} f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T \nabla^2 f(x_k) s + \frac{M}{6} \|s\|^3.$$

The purpose of the cubic term is to ensure that the selected step is not too big, since the second-order Taylor approximation is only locally accurate. Minimizing the above cubic model is equivalent to solving a constrained one-dimensional problem [21], and several specialized solvers are available in the literature, for example [23] and [24].

To use our Hessian estimator, we resort to a version of the cubic-regularized Newton method that allows for approximate function derivatives, minimizing the cubic model

$$m_k(s) = f(x_k) + g_k^T s + \frac{1}{2} s^T H_k^{r(k)} s + \frac{M_k}{6} \|s\|^3 . \tag{12}$$

2) THE NOVEL FEDZCR AND FEDZACR

Combining the federated derivative estimation described in Section III-B with cubic regularization, we obtain a novel algorithm for non-convex federated optimization, that we call FedZCR (Federated Zeroth-order Cubic Regularization). The first part of Algorithm 1 implements the federated estimation procedure, which builds the gradient and Hessian estimators at the central server. In the second part of the pseudocode the server updates the model parameters by solving the subproblem (12).

Remark 8: Cubic-regularized Newton methods are advantageous also for convex optimization problems, since *(i)* compared to standard Newton methods they provide global convergence guarantees, and *(ii)* they do not require the Hessian to be positive-definite. This allows us to directly use the incremental Hessian estimator (1), that is not guaranteed to be positive-definite. In comparison, in FedZeN [8] it is needed to either add a regularization term or to clip the eigenvalues of the estimated Hessian.

We also introduce the variant FedZACR, where A stands for adaptive, that updates the coefficient M at each iteration according to the performance of the algorithm. Our approach is similar to the one of [23], that treats M as the reciprocal of a trust-region radius. Trust-region algorithms set a maximum value for the norm of the step, which results in a constrained minimization problem. Cubic regularization replaces the hard constraint with the cubic penalty term, that allows a smooth trade-off between model minimization and algorithmic wariness. In particular, [23] defines the performance indicator ρ as the ratio between the actual function value improvement and

Algorithm 1: FedZCR and FedZACR.

/* Initialization at the Central Server (CS) */
Choose $x_1 \in \mathbb{R}^d$, $M_1 > 0$, $H_1^0 \in \mathbb{R}^{d\times d}$ symmetric.
Broadcast a random seed for the PRNGs.
Only for FedZACR: $\gamma_1, \gamma_2 > 1$, $0 < t_1 < t_2 < 1$.

for each iteration $k = 1, 2, \dots$ **do**
  CS: Broadcast x_k and $r(k)$.
  /* Federated derivative estimation: */
  All nodes: Execute $\lceil r/d \rceil$ times one of the sampling procedures in Appendix B using the synchronized PRNG to generate $\lceil r/d \rceil$ bases of $\mathbb{R}^d$. Merge them and pick the first r vectors $\{u_1 \dots u_{r(k)}\}$.
  for each client $i \in [n]$ **do**
    Compute c_{ij}, b_{ij} for $j \in [r(k)]$ using (9, 10).
  CS: If $k > 1$ set $H_k^0 = H_{k-1}^{r(k)}$.
    Compute g_k, H_k^r using (11).
  /* Cubic-regularized Newton method: */
  CS: $s_k = \operatorname{argmin}_{s\in\mathbb{R}^d} m_k(s)$
    $x_{k+1} = x_k + s_k$.
  Only for FedZACR:
    /* involves an extra communication round */
    for each client $i \in [n]$ **do**
      Compute $f_i(x_k + s_k)$.
    CS: $\rho_k = \frac{f(x_k)-f(x_k+s_k)}{f(x_k)-m(s_k)}$

$$x_{k+1} = \begin{cases} x_k + s_k & \text{if } \rho_k \geq t_1 \\ x_k & \text{otherwise} \end{cases}$$

$$M_{k+1} = \begin{cases} M_k\gamma_1 & \text{if } \rho_k > t_2 \\ M_k & \text{if } t_1 \leq \rho_k \leq t_2 \\ M_k/\gamma_2 & \text{otherwise} \end{cases}$$

expected one. Given two threshold parameters $0 < t_1 < t_2 < 1$, if $t_1 \leq \rho \leq t_2$ the iteration is marked as successful, and the current value of M is regarded as appropriate. If $\rho > t_2$ the iteration is considered very successful and the penalty coefficient is lowered to allow for bigger steps. If instead $\rho < t_1$ the iteration is deemed unsuccessful and the step is rejected. This implies that the model in (12) is not sufficiently accurate, and therefore M should be increased to reduce the size of the next step. Typical values for the threshold parameters are $t_1 = 0.1$, $t_2 = 0.9$.

3) CONVERGENCE ANALYSIS OF FEDZCR

Below we exploit the bounds regarding the Hessian estimator to derive the values of μ and r for which the non-adaptive version of the algorithm converges to a second-order stationary point with optimal iteration complexity and arbitrary precision.

In order to preserve the appealing features of the original cubic-regularized Newton method while using inexact gradient and Hessian, the estimators in (12) must satisfy certain accuracy requirements. Many choices are available in the literature for these requirements, which correspond to different convergence results. In particular, it is shown in [25] that the convergence properties of the exact cubic-regularized Newton method can be retained provided that at all iterations

$$\left\| H_k^{r(k)} - \nabla^2 f(x_k) \right\| \leq \alpha \left\| s_{k-1} \right\|, \tag{13}$$

$$\|g_k - \nabla f(x_k)\| \leq \beta \left\| s_{k-1} \right\|^2, \quad M_k \geq \frac{2}{3}L_2 + 8\alpha + 8\beta,$$

where $\alpha, \beta \in \mathbb{R}$ are positive constants. It should be noted that the above conditions are implementable in practice, as they involve the step s_{k-1} computed in the previous iteration. In comparison, most other works similar to [25] end up losing the saddle-avoidance property, converging to first-order stationary points, or require the knowledge of the step s_k before it is even computed (e.g. [26]).

The following Theorem 2 provides sufficient conditions for which FedZCR encounters a $(\tau, \sqrt{\tau})$-optimal point in $O(\tau^{-3/2})$ iterations with high probability.

Theorem 2: Let the objective function satisfy the smoothness properties in Assumption 1. Fix two positive constants $\alpha, \beta \in \mathbb{R}$. At each iteration k of FedZCR, choose $\delta_k \in (0, 1)$, set

$$\epsilon_k \leq \alpha \left\| s_{k-1} \right\|, \qquad M_k \geq \frac{2}{3}L_2 + 8\alpha + 8\beta,$$

$$\mu_k \leq \min\left\{ \sqrt{\frac{6\beta}{dL_2}} \left\| s_{k-1} \right\|, \sqrt{\frac{12\, d\epsilon_k\delta_k(1-\eta)}{L_3}} \right\},$$

and choose $r(k)$ greater or equal than the minimum number of search directions prescribed by Theorem 1. Then, after τ iterations, with probability at least $\prod_{i=1}^{\tau}(1-\delta_i)$ the sequence $\{x_i \mid i \in [\tau]\}$ generated by FedZCR contains a point $\tilde{x}$ such that

$$\|\nabla f(\tilde{x})\| \leq \frac{C_1}{(\tau-1)^{2/3}}, \quad \lambda_{\min}(\nabla^2 f(\tilde{x})) \geq -\frac{C_2}{(\tau-1)^{1/3}}$$

where C_1 and C_2 are universal constants.

Proof: The proof is a straightforward application of (6) and Theorem 1 to the requirements in (13). Imposing the condition

$$\|\nabla f(x) - \hat{g}\| \leq \frac{dL_2\mu^2}{6} \leq \beta \left\| s_{k-1} \right\|^2$$

leads to $\mu_k \leq \sqrt{6\beta/(dL_2)}\|s_{k-1}\|$. The definition of ϵ_k and μ_k is a consequence of invoking the warm-start case of Theorem 1. The condition on M_k and the values of C_1 and C_2 can be found in [25]. ■

IV. NUMERICAL RESULTS

We now present a set of numerical simulations that showcases the performance of the algorithm proposed in Section III-C2. We focus on the adaptive version FedZACR, that can be directly applied to any problem as it does not require tuning the parameter M_k. We use the standard thresholds $t_1 = 0.1$, $t_2 = 0.9$ and set $\gamma_1 = 5$, $\gamma_2 = 20$, that work well for all our experiments. We consider both convex and non-convex optimization

problems and compare FedZACR with other state-of-the-art federated algorithms, including both zeroth-order methods and methods for convex optimization that use the exact derivatives. We use $f^\star$ to denote the minimum value of the loss function to be optimized.

A. CONVEX COST FUNCTIONS

Our first experiments concern logistic regression, that is the most common benchmark function for convex optimization. We consider the same convex optimization problem of [8], namely binary classification via logistic regression using two classes of the dataset Covertype [27]. The feature size is $d = 55$ and the data points are evenly split across $n = 100$ clients, so that each participant has 500 samples.

1) COMPARISON WITH ZEROTH-ORDER METHODS

We compare FedZACR with the following zeroth-order federated methods:

- FedZO [3], that is the zeroth-order counterpart of FedAvg, setting learning rate $\eta = 0.1$, $H = 5$ local epochs and $b_2 = d$ perturbation directions.
- ZONE-S [4], where at each iteration only one client is updated and the server minimizes an augmented Lagrangian function. In our implementation we use Nesterov accelerated gradient to solve the Lagrangian subproblem.
- A federated version of ZO-JADE [10], which is a fully-distributed algorithm for convex optimization that allows for general network topologies. ZO-JADE estimates only the diagonal of the Hessian matrix to use it as a preconditioner.
- FedZeN [8], from which our algorithm inherits the distributed estimation procedure. FedZeN is specifically designed for strongly-convex optimization and enforces the estimated Hessian to be positive-definite applying either regularization or eigenvalue clipping. Since we replicate the test performed in [8], we tune the parameters of FedZeN as described there.

Differently from [8], which keeps the number of search directions fixed for all iterations in the simulations, we try some heuristics for $r(k)$. We compare increasing, constant and decreasing schedules, and we observe how they affect the convergence rate, the total communication cost and the overall number of function queries. The approach that works best in our tests is an increasing schedule of the form $r(k) = \min\{r_{\max}, \lfloor \nu^k r(1) \rfloor\}$, $\nu > 1$, where the number of search directions grows exponentially up to a budget threshold $r_{\max}$. Empirically, in the first iterations it suffices to move in an approximately correct direction to obtain substantial improvements, so we decide not to aim for perfection and settle for a cheap Hessian estimate. Indeed, in the first iterations the decision vector and consequently the optimization landscape typically evolve rapidly, so it is not worth spending many function queries at this stage. On the contrary, as we get closer to the optimum it becomes increasingly more important to make precise updates, which requires an accurate estimate of the Hessian. For this reason, our schedule progressively increases the amount of function queries to allow convergence in few iterations. In our tests, sparing function queries in the early iterations and performing a few computationally expensive final iterations achieves the best trade-off between number of iterations and total number of function evaluations needed for convergence, and is a better choice than just keeping a constant $r(k)$. For a fair comparison in our test we apply this heuristics to both FedZACR and FedZeN, setting $r(1) = 2\,d$, $r_{\max} = 8\,d$ and $\nu = 1.03$.

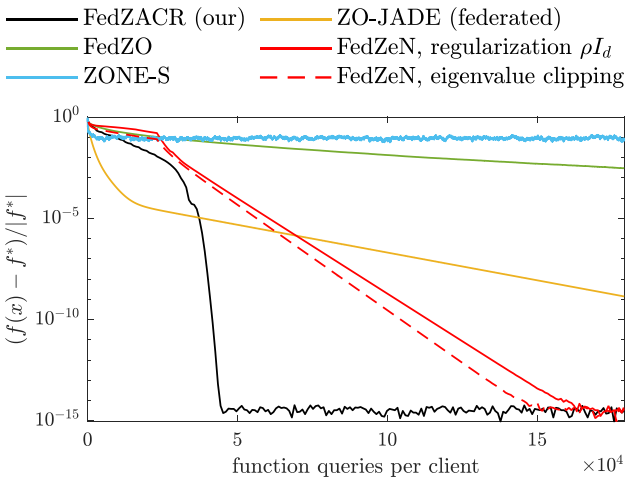

FIGURE 3. Training loss of zeroth-order logistic regression. The competitors of FedZACR are other federated zeroth-order algorithms.

Fig. 3 shows that FedZACR outperforms all the other algorithms including FedZeN [8], that is based on the same distributed derivative estimation procedure. Moreover, FedZeN has an unfair advantage in knowing that the problem is convex, which allows it to "cheat" and tweak the Hessian estimate to be positive-definite. In comparison, FedZACR is not aware of convexity and makes up for Hessian uncertainty by using the cubic penalty to limit the stepsize. The training loss is plotted against the number of function queries, which are assumed to be the most expensive computations and represent the main evaluation metric in the zeroth-order optimization field.

2) COMPARISON WITH NON-ZEROTH-ORDER METHODS

We now consider the scenario where clients are able to directly compute the exact derivatives of their local functions. We compare the zeroth-order FedZACR (with the same parameters as in the previous test) with the following competitors that make use of the exact derivatives:

- GIANT [17], that computes an approximate Newton direction by replacing the arithmetic mean of the local Hessian matrices with the harmonic mean. To prevent algorithmic divergence we use a fixed stepsize equal to 0.2, which is the largest stepsize the algorithm can handle in our experiment.

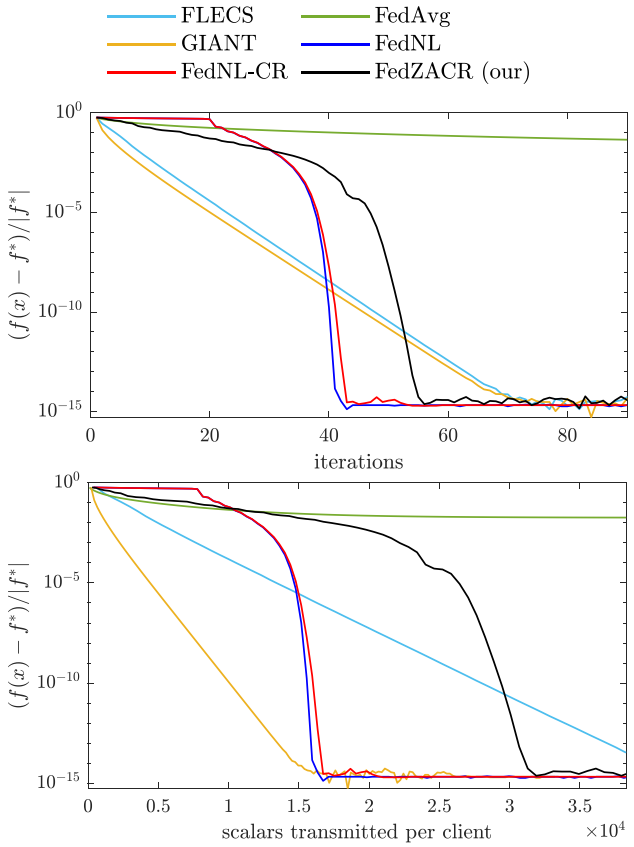

FIGURE 4. Training loss of standard logistic regression: unfair comparison between the zeroth-order FedZACR versus other federated algorithms that make use of the exact derivatives. While FedZACR is clearly disadvantaged, its performance is comparable to second-order state-of-the-art algorithms.

- FedNL and FedNL-CR [18], where clients send to the server compressed Hessian innovations and the corresponding compression errors. The vanilla FedNL applies a Newton-type step, while FedNL-CR assumes the knowledge of the Lipschitz constant of the Hessian to apply cubic-regularized Newton method. In our implementation we use stepsize $\alpha = 1$ and the rank-3 compression operator based on singular value decomposition.
- FLECS [20], that starts from FedNL and proposes alternative ways to update the Hessian estimate at the server and compute the quasi-Newton step. We consider FLECS with "direct update" and "truncated inverse Hessian approximation", which are respectively Algorithms 2 and 3 in [20].
- FedAvg [28], that is the equivalent of stochastic gradient descent for federated learning. While FedAvg is a first-order method and cannot compete with the above second-order algorithms, we still include it as it represents the baseline for federated learning. We set learning rate $\eta = 0.1$, $E = 10$ local epochs and batchsize $B = 50$.

Fig. 4 shows that while being a zeroth-order algorithm, FedZACR can compete with state-of-the-art second-order methods that have access to the exact derivatives. The two

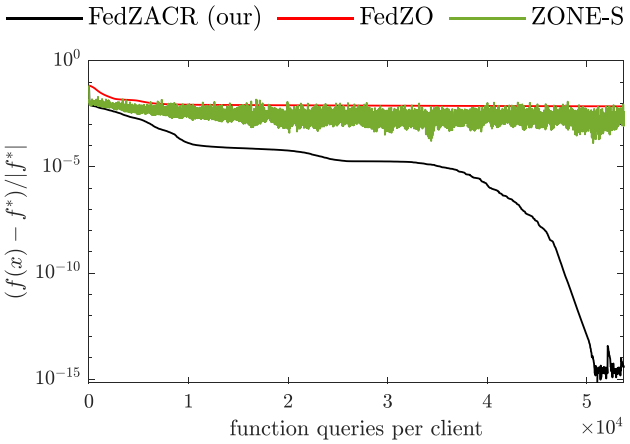

FIGURE 5. Function value suboptimality of zeroth-order linear regression using a non-convex loss. We compare FedZACR to other federated zeroth-order algorithms.

variants of FedNL achieve the best performance, but together with GIANT they can be applied only when the objective function is strongly-convex. We remark that all the second-order competitors require the computation of the local Hessian matrices, and do not support black-box objectives or approximate derivatives.

B. NON-CONVEX COST FUNCTIONS

The following experiments highlight the distinguishing features of FedZACR: our algorithm allows tackling non-convex federated learning problems, does not require the objective function to be differentiable, and exploits the second-order derivative to achieve faster convergence.

1) NON-CONVEX ROBUST LINEAR REGRESSION

In this test the Bike Sharing dataset [29] is evenly partitioned over $n = 100$ clients. The task is to predict the number of rented bikes using robust linear regression, that is less sensitive to the presence of outlier data samples. Each client i owns a matrix $A_i \in \mathbb{R}^{m_i \times d}$, whose rows are data samples, and a response vector $b_i \in \mathbb{R}^{m_i}$. We adopt the loss function proposed in [30], that uses two parameters (α, c) to generalize several standard loss functions. In particular, we choose $\alpha = -0.5$ and $c = 3$, that correspond to the non-convex loss

$$f_i(r_i) = \frac{|\alpha - 2|}{\alpha}\left(\left(\frac{(r_i/c)^2}{|\alpha - 2|} + 1\right)^{\alpha/2} - 1\right), \qquad (14)$$

where in our case $r_i = A_i x - b_i$ is the residual error of linear regression at client i. The results of this test are shown in Fig. 5.

2) BLACK-BOX NEURAL NETWORK FINETUNING

The test consists in fine-tuning a black-box convolutional neural network on a classification task. The network is an off-the-shelf model composed of a convolutional layer, two fully-connected layers and ReLU activation functions. The

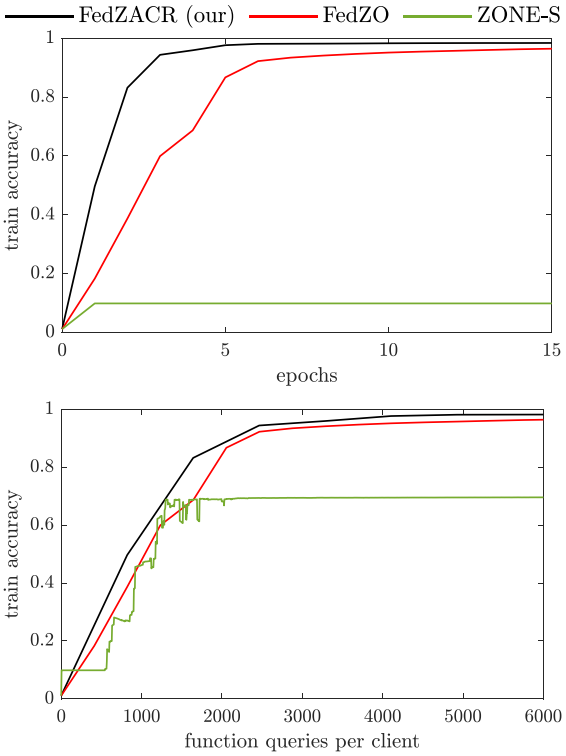

FIGURE 6. Training accuracy achieved while fine-tuning a black-box neural network on a classification task. FedZACR is compared with other federated zeroth-order algorithms for non-convex optimization.

first two layers are frozen and treated as a black-box opaque to the optimizer. The last layer is trained from scratch, and the total amount of learnable parameters is $d = 410$. We use the standard MNIST dataset [31] and split it over a pool of $n = 100$ clients. We generate a non-i.i.d. data distribution with the following procedure: sort the data samples according to their label, partition them in n shards, and assign a different shard to each client. As a result, all participants have the same amount of data samples, but the local data distributions are different.

Since the objective function is non-convex and non-differentiable, we can compare FedZACR only with the zeroth-order methods FedZO and ZONE-S. The hyperparameters are the same as in the previous test (Section IV-A), with the only difference that for FedZACR we set a constant $r(k) = d$ at all iterations. To reduce the computational complexity all algorithms employ subsampling, discussed in Appendix C: each client trains on a batch containing 10% of its data samples, randomly chosen at each local iteration.

Fig. 6 shows that FedZACR trains the network faster and more efficiently compared to its contenders, requiring fewer iterations and function queries to achieve the same training accuracy.

V. CONCLUSION

We have addressed two different but related topics. The common thread is a randomised estimator of the Hessian matrix suitable for zeroth-order optimization, where functions are black boxes accessible only through input-output pairs and function derivatives are not available. Moreover the estimator is incremental, meaning that it is built iteratively updating the last estimate as new data comes.

In the first part of the paper we have analyzed the convergence properties of the estimator, bounding the estimation error after a finite number of iterations and deriving the minimum number of search directions needed to achieve a given accuracy with high probability.

In the second part we have proposed a novel algorithm for federated optimization of non-convex black-box functions. The algorithm estimates the Hessian matrix of the global objective function and runs an approximate cubic-regularized Newton method. Applying our results regarding the Hessian estimator we have proved convergence to a second-order optimal point with high probability and optimal iteration complexity. We have validated the theory with numerical results, showing that our algorithm outperforms several state-of-the-art methods in multiple types of federated optimization problems.

APPENDIX A
PROOFS OF SECTION II

1) PROOF OF LEMMA 1

Proof: Since S is symmetric, its singular value decomposition is $S = V\Sigma V^T$ and its singular values are equal to the absolute value of the eigenvalues, i.e. $\sigma_i = |\lambda_i|$, $i \in [d]$. Since V is an orthonormal basis of $\mathbb{R}^d$, we can write the generic vector $u \sim U(\mathbb{S})$ as $u = V\alpha$ for suited $\alpha \in \mathbb{R}^d$. Since the uniform spherical distribution is invariant with respect to orthogonal transformations (rotational invariance) and V is orthonormal, $\alpha = V^T u \sim U(\mathbb{S})$. Then we have

$$\begin{aligned}
&\left\| S - (u^T S u) u u^T \right\|_F \\
&\quad = \left\| V\Sigma V^T - (\alpha^T V^T V \Sigma V^T V \alpha) V \alpha \alpha^T V^T \right\|_F \\
&\quad = \|V\|_F \left\| \Sigma - (\alpha^T \Sigma \alpha) \alpha\alpha^T \right\|_F \left\| V^T \right\|_F ,
\end{aligned}$$

where $\|V\|_F = 1$, i.e. we can prove the statement of Lemma 1 using the diagonal matrix Σ instead of the general symmetric S. Using the definition of Frobenius norm, namely $\|A\|_F^2 = \mathrm{Tr}(A^T A) = \sum_{i=1}^d \sigma_i^2$, and the invariance of the trace under circular shifts, namely $\mathrm{Tr}(\Sigma\alpha\alpha^T) = \mathrm{Tr}(\alpha^T\Sigma\alpha) = \alpha^T\Sigma\alpha$ and $\mathrm{Tr}(\alpha\alpha^T) = \alpha^T\alpha = \|\alpha\|^2 = 1$, we have

$$\begin{aligned}
&\left\| \Sigma - (\alpha^T \Sigma \alpha) \alpha\alpha^T \right\|_F^2 \\
&\quad = \mathrm{Tr}[(\Sigma - (\alpha^T\Sigma\alpha)\alpha\alpha^T)^T (\Sigma - (\alpha^T\Sigma\alpha)\alpha\alpha^T)] \\
&\quad = \mathrm{Tr}(\Sigma^T\Sigma) + (\alpha^T\Sigma\alpha)^2 \mathrm{Tr}(\alpha\alpha^T\alpha\alpha^T) \\
&\qquad - 2(\alpha^T\Sigma\alpha)\mathrm{Tr}(\Sigma\alpha\alpha^T) \\
&\quad = \|\Sigma\|_F^2 + (\alpha^T\Sigma\alpha)^2 - 2(\alpha^T\Sigma\alpha)^2
\end{aligned}$$

$$= \|\Sigma\|_F^2 - \left(\sum_{i=1}^{d} \alpha_i^2 \sigma_i\right)^2$$
$$\leq \|\Sigma\|_F^2 - \sum_{i=1}^{d} \left(\alpha_i^2 \sigma_i\right)^2,$$

and by monotonicity of the square root

$$\left\|\Sigma - (\alpha^T \Sigma \alpha)\alpha\alpha^T\right\|_F \leq \sqrt{\|\Sigma\|_F^2 - \sum_{i=1}^{d} \left(\alpha_i^2 \sigma_i\right)^2}.$$

We now take the expectation and apply Jensen's inequality, recalling that the square root is a concave operator:

$$\mathbb{E}\left[\left\|\Sigma - (\alpha^T \Sigma \alpha)\alpha\alpha^T\right\|_F\right] \leq \sqrt{\|\Sigma\|_F^2 - \sum_{i=1}^{d} \sigma_i^2 \mathbb{E}\left[\alpha_i^4\right]}$$

Being $\alpha \sim U(\mathbb{S})$, each component of α is identically distributed and

$$\alpha_i^2 \sim Beta\left(\frac{1}{2}, \frac{d-1}{2}\right) \quad i \in [d].$$

The mean and variance of each α_i^2 are respectively

$$\mathbb{E}[\alpha_i^2] = \frac{1}{d}, \quad \text{var}[\alpha_i^2] = \frac{2(d-1)}{d^2(d+2)}.$$

Using basic probability, we have

$$\mathbb{E}[\alpha_i^4] = \mathbb{E}[\alpha_i^2]^2 + \text{var}[\alpha_i^2] = \frac{3}{d(d+2)}.$$

Using this result, we finally get

$$\mathbb{E}\left[\left\|\Sigma - (\alpha^T \Sigma \alpha)\alpha\alpha^T\right\|_F\right] = \sqrt{\|\Sigma\|_F^2 - \left(\sum_{i=1}^{d} \sigma_i^2\right) \mathbb{E}\left[\alpha_1^4\right]}$$
$$= \sqrt{\|\Sigma\|_F^2 - \|\Sigma\|_F^2 \frac{3}{d(d+2)}}$$

and recalling that $\|\Sigma\| = \|S\|$ the proof is concluded. ■

2) PROOF OF LEMMA 2

Proof: For simplicity of notation we drop the subscript of the search direction u_j. Below we use Taylor expansion with integral remainder, the Lipschitz properties of the function and the fact that $\|u\| = 1$.

$$\mathbb{E}\left[\left\|H^j - \hat{H}^j\right\|\right] = \mathbb{E}\left[\left\|\left(u^T \nabla^2 f(x) u - \frac{f(x+\mu u) - 2f(x) + f(x-\mu u)}{\mu^2}\right) uu^T\right\|\right]$$
$$= \mathbb{E}\left[\left\| -\frac{\mu}{2} uu^T \int_0^1 (1-t)^2 \langle (D^3 f)(x + t\mu u) - (D^3 f)(x - t\mu u), u, u, u\rangle \, dt \right\|\right]$$
$$\leq \mathbb{E}\left[\left\| -\frac{\mu}{2} uu^T \int_0^1 (1-t)^2 L_3 \|u\|^3 \|2t\mu u\| \, dt \right\|\right]$$
$$= \mathbb{E}\left[\left\| -\frac{\mu^2}{12} L_3 \, uu^T \right\|\right]$$
$$= \frac{\mu^2 L_3}{12\, d} \mathbb{E}\left[\|uu^T\|\right].$$

If we consider the spectral norm, then the largest singular value of the rank-one matrix uu^T is 1. If instead we take the Frobenius norm, then $\|uu^T\|_F = \sqrt{\text{Tr}(uu^T)} = \sqrt{u^T u} = \|u\| = 1$.

To obtain the convergence rate of (3) towards the exact Hessian we use the triangular inequality and Lemma 1 choosing $S = \nabla^2 f(x) - H^{j-1}$.

$$\mathbb{E}\left[\left\|\nabla^2 f(x) - \hat{H}^j\right\|_F\right] \leq$$
$$\leq \mathbb{E}\left[\left\|\nabla^2 f(x) - H^j\right\|_F + \left\|H^j - \hat{H}^j\right\|_F\right]$$
$$\leq \eta \left\|\nabla^2 f(x) - H^{j-1}\right\|_F + \frac{\mu^2 L_3}{12\, d}.$$

The statement follows recalling that by assumption $H^{j-1} = \hat{H}^{j-1}$. ■

3) PROOF OF LEMMA 3

Proof: We consider the whole trajectory of the estimator, applying multiple times Lemma 2 together with the tower property of conditional expectation.

$$\mathbb{E}\left[\left\|\nabla^2 f(x) - \hat{H}^r\right\|_F \middle| \hat{H}^0\right]$$
$$= \mathbb{E}_{\hat{H}^{r-1}}\left[\mathbb{E}_{\hat{H}^r}\left[\left\|\nabla^2 f(x) - \hat{H}^r\right\|_F \middle| \hat{H}^0, \hat{H}^{r-1}\right]\right]$$
$$\leq \mathbb{E}_{\hat{H}^{r-1}}\left[\eta \left\|\nabla^2 f(x) - \hat{H}^{r-1}\right\|_F + \frac{\mu^2 L_3}{12\, d} \middle| \hat{H}^0\right]$$
$$= \frac{\mu^2 L_3}{12\, d} + \mathbb{E}_{\hat{H}^{r-2}}\left[\mathbb{E}_{\hat{H}^{r-1}}\left[\eta \left\|\nabla^2 f(x) - \hat{H}^{r-1}\right\|_F \middle| \hat{H}^0, \hat{H}^{r-2}\right]\right]$$
$$\leq \frac{\mu^2 L_3}{12\, d} \sum_{i=0}^{1} \eta^i + \mathbb{E}_{\hat{H}^{r-2}}\left[\eta^2 \left\|\nabla^2 f(x) - \hat{H}^{r-2}\right\|_F \middle| \hat{H}^0\right]$$
$$\leq \frac{\mu^2 L_3}{12\, d} \sum_{i=0}^{r-2} \eta^i + \mathbb{E}_{\hat{H}^1}\left[\eta^{r-1} \left\|\nabla^2 f(x) - \hat{H}^1\right\|_F \middle| \hat{H}^0\right]$$
$$\leq \frac{\mu^2 L_3}{12\, d} \sum_{i=0}^{r-1} \eta^i + \eta^r \left\|\nabla^2 f(x) - \hat{H}^0\right\|_F$$

■

4) PROOF OF THEOREM 1

Proof. (Memory-less case): If $\hat{H}_k^0$ is the zero matrix, using the fact that for a matrix A with rank p it holds $\|A\|_F \leq \sqrt{p}\|A\|$, we get

$$\left\|\nabla^2 f(x_k) - \hat{H}_k^0\right\|_F \leq \sqrt{d} \left\|\nabla^2 f(x_k)\right\| \leq \sqrt{d} L_1.$$

Using Markov's inequality and plugging the above inequality in Lemma 3, $\forall \epsilon > 0$ and $\forall k \geq 1$

$$\begin{aligned}
&\mathbb{P}\left(\left\|\nabla^2 f(x_k) - \hat{H}_k^{r(k)}\right\|_F \geq \epsilon \,\middle|\, \hat{H}_k^0 = 0\right) \\
&\leq \frac{1}{\epsilon}\mathbb{E}\left[\left\|\nabla^2 f(x_k) - \hat{H}_k^{r(k)}\right\|_F \,\middle|\, \hat{H}_k^0 = 0\right] \\
&\leq \frac{1}{\epsilon}\left(\eta^{r(k)}\sqrt{d}L_1 + \frac{\mu^2 L_3}{12\,d}\sum_{i=0}^{r(k)-1}\eta^i\right) \leq \delta
\end{aligned}$$

where in the last step we ask the probability to be smaller than an arbitrary $\delta > 0$. The last inequality is equivalent to the following ones.

$$\eta^{r(k)}\sqrt{d}L_1 + \frac{\mu^2 L_3}{12\,d}\frac{1-\eta^{r(k)}}{1-\eta} \leq \epsilon\delta,$$

$$\eta^{r(k)}\sqrt{d}L_1(1-\eta) + \frac{\mu^2 L_3}{12\,d}(1-\eta^{r(k)}) \leq \epsilon\delta(1-\eta),$$

$$\eta^{r(k)}\left(\sqrt{d}L_1(1-\eta) - \frac{\mu^2 L_3}{12\,d}\right) \leq \epsilon\delta(1-\eta) - \frac{\mu^2 L_3}{12\,d}.$$

Solving for $r(k)$ one gets the following conditions, where the upper bound on μ ensures that the logarithm is well defined.

$$\mu \leq \sqrt{\frac{12\,d(1-\eta)}{L_3}\min\left\{\epsilon\delta,\ \sqrt{d}L_1\right\}},$$

$$r(k) \geq \log_\eta\left(\frac{\epsilon\delta(1-\eta) - \frac{\mu^2 L_3}{12\,d}}{\sqrt{d}L_1(1-\eta) - \frac{\mu^2 L_3}{12\,d}}\right).$$

To deduce the limiting behaviour for large values of d we use the definition of η and the following Taylor expansions:

$$\eta = \sqrt{1 - \frac{3}{d(d+2)}},$$

$$\sqrt{1-x} = 1 - \frac{x}{2} + o(x^2) \qquad \text{for } |x| < 1,$$

$$\log(1-x) = -x - \frac{x^2}{2} + o(x^3) \qquad \text{for } |x| < 1.$$

Indeed, for $d \to \infty$ and $\epsilon\delta \leq \sqrt{d}L_1$ we obtain

$$\begin{aligned}
\mu &\leq \sqrt{\frac{12\,d(1-\eta)}{L_3}\min\left\{\epsilon\delta,\ \sqrt{d}L_1\right\}} \\
&\propto \sqrt{d\left(1 - \sqrt{1 - \frac{3}{d(d+2)}}\right)\epsilon\delta} \\
&= \sqrt{d\left(1 - \left(1 - \frac{3}{2d(d+2)} + o\left(\frac{1}{d^4}\right)\right)\right)\epsilon\delta} \\
&\propto \sqrt{d\frac{3}{2\,d^2}\epsilon\delta} = O\left(\frac{\epsilon\delta}{\sqrt{d}}\right).
\end{aligned}$$

Using also the bound on μ and the change of basis rule for the logarithm, the asymptotic scaling of $r(k)$ is

$$\begin{aligned}
r(k) &\geq \log_\eta\left(\frac{\epsilon\delta(1-\eta) - \frac{\mu^2 L_3}{12\,d}}{\sqrt{d}L_1(1-\eta) - \frac{\mu^2 L_3}{12\,d}}\right) \propto \log_\eta\left(\frac{\epsilon\delta}{\sqrt{d}L_1}\right) \\
&\propto \log\left(\frac{\epsilon\delta}{\sqrt{d}}\right) / \log\left(1 - \frac{3}{2d(d+2)}\right) \\
&= \log\left(\frac{\epsilon\delta}{\sqrt{d}}\right) / \left(-\frac{1}{d^2} + o\left(\frac{1}{d^4}\right)\right) = O\left(d^2 \log\left(\frac{\sqrt{d}}{\epsilon\delta}\right)\right).
\end{aligned}$$

(Warm-start case): In this case we use the assumption $\hat{H}_i^0 = \hat{H}_{i-1}^{r(i-1)}$ $\forall i > 1$, and the following relation between spectral norm and Frobenius norm:

$$\begin{aligned}
\left\|\nabla^2 f(x) - \nabla^2 f(y)\right\|_F &\leq \sqrt{d}\left\|\nabla^2 f(x) - \nabla^2 f(y)\right\| \\
&\leq \sqrt{d}L_2\|x-y\| \quad \forall x, y \in \mathbb{R}^d.
\end{aligned}$$

The next steps are contained in (15) shown at the bottom of the next page, where we repeatedly apply the tower property of conditional expectation and Lemma 3. In the last passage we also highlight the variable of interest $r(k)$. For the sake of notation, we will denote the term inside the curly brackets with c_k. Plugging (15) in Markov's inequality, $\forall \epsilon > 0$

$$\begin{aligned}
&\mathbb{P}\left(\left\|\nabla^2 f(x_k) - \hat{H}_k^{r(k)}\right\|_F \geq \epsilon \,\middle|\, \hat{H}_1^0\right) \\
&\leq \frac{1}{\epsilon}\mathbb{E}\left[\left\|\nabla^2 f(x_k) - \hat{H}_k^{r(k)}\right\|_F \,\middle|\, \hat{H}_1^0\right] \\
&\leq \frac{1}{\epsilon}\left(\eta^{r(k)}c_k + \frac{1}{1-\eta}\frac{\mu^2 L_3}{12\,d}\right) \leq \delta.
\end{aligned}$$

Similarly to the previous case, we have imposed that the above probability is less than an arbitrary $\delta > 0$. By solving for $r(k)$ the proof is concluded. Also in this case the upper bound on μ follows by requiring that the logarithm is well defined. ■

APPENDIX B
METHODS TO SAMPLE ORTHONORMAL VECTORS FROM $\mathcal{U}(\mathbb{S})$

A first procedure to generate orthonormal vectors $\sim \mathcal{U}(\mathbb{S})$ is proposed in [32]: given a matrix $X \in \mathbb{R}^{d\times k}$ with $k \leq d$ and entries $X_{ij} \overset{i.i.d.}{\sim} \mathcal{N}(0,1)$, the matrix $U = X(X^TX)^{-1/2}$ is uniformly sampled from the Stiefel manifold $V_{k,d} = \{U \in \mathbb{R}^{d\times k}$ such that $U^TU = I_d\}$. As remarked in [11], that baptizes this procedure Stiefel sampling, the marginal distribution of the columns of U is $\mathcal{U}(\mathbb{S})$. Intuitively, when $k = 1$, $X \in \mathbb{R}^d$ is a random vector $\sim \mathcal{N}(0, I_d)$ and $U = X/\|X\|$ is simply the normalized vector.

A second way to obtain orthogonal vectors $\sim \mathcal{U}(\mathbb{S})$ follows from observing that $V_{d,d}$ is the orthogonal group $O(d)$, and therefore uniformly sampling $O(d)$ is equivalent to uniformly sampling the Stiefel manifold. A method to generate a matrix in $O(d)$ distributed with Haar measure, which is the analogue of the uniform distribution, is described in [33]: take a matrix $X \in \mathbb{R}^{d\times d}$ whose entries are standard complex normal random

variables, compute its QR decomposition $X = QR$ and the matrix

$$\Lambda = \begin{bmatrix} \frac{R_{11}}{|R_{11}|} & & \\ & \ddots & \\ & & \frac{R_{dd}}{|R_{dd}|} \end{bmatrix},$$

then the matrix $U = Q\Lambda$ is distributed in $O(d)$ with Haar measure.

APPENDIX C
IMPROVED EFFICIENCY USING SUBSAMPLING

To keep a reasonable computational complexity in case of huge datasets, it is possible to incorporate subsampling in the procedure described in Section III-B. At each iteration the function value is evaluated using only a random subset of the data samples. We denote the function computed on the data subset $S = \{(i, j),\ i \subseteq [n],\ j \subseteq [m_i]\}$ with $f_S(x)$. The next assumption extends Assumption 1 to the loss functions corresponding to the individual data samples, and is needed to deal with the subsampled case.

Assumption 3 (Smoothness): For each data subset S, all the loss functions $f_{ij}(x)$, $(i, j) \in S$ satisfy Assumption 1.

We now provide lower bounds on the minimum number of data samples required by the zeroth-order derivative estimators to attain a target accuracy with high probability. In the following the subscripts do not refer to the iteration number, but rather to the data samples. For example, $\hat{H}^r_{ij}$ denotes the Hessian estimator corresponding to the loss function $f_{ij}(x)$, associated with the j-th data sample of client i.

Proposition 1: Suppose that at each iteration the Hessian estimator of the function f_{ij} is initialized as $\hat{H}^0_{ij} = 0_d$, $\forall (i, j) \in S$. Then the approximation error of the global Hessian estimator at the server

$$\hat{H}^r_S = \frac{1}{|S|} \sum_{(i,j)\in S} \frac{m}{nm_i} \hat{H}^r_{ij}$$

$$\begin{aligned}
&\mathbb{E}\left[\left\| \nabla^2 f(x_k) - \hat{H}_k^{r(k)} \right\|_F \middle| \hat{H}_1^0 \right] = \mathbb{E}_{\hat{H}_k^0}\left[\mathbb{E}_{\hat{H}_k^{r(k)}}\left[\left\| \nabla^2 f(x_k) - \hat{H}_k^{r(k)} \right\|_F \middle| \hat{H}_1^0, \hat{H}_k^0 \right] \right] \\
&\le \mathbb{E}_{\hat{H}_k^0}\left[\eta^{r(k)} \left\| \nabla^2 f(x_k) - \hat{H}_k^0 \right\|_F + \frac{\mu^2 L_3}{12\, d} \sum_{i=0}^{r(k)-1} \eta^i \middle| \hat{H}_1^0 \right] \\
&= \frac{\mu^2 L_3}{12\, d} \sum_{i=0}^{r(k)-1} \eta^i + \eta^{r(k)} \mathbb{E}_{\hat{H}_{k-1}^{r(k-1)}}\left[\left\| \nabla^2 f(x_k) - \hat{H}_{k-1}^{r(k-1)} \right\|_F \middle| \hat{H}_1^0 \right] \\
&\le \frac{\mu^2 L_3}{12\, d} \sum_{i=0}^{r(k)-1} \eta^i + \eta^{r(k)} \mathbb{E}_{\hat{H}_{k-1}^{r(k-1)}}\left[\left\| \nabla^2 f(x_k) - \nabla^2 f(x_{k-1}) \right\|_F + \left\| \nabla^2 f(x_{k-1}) - \hat{H}_{k-1}^{r(k-1)} \right\|_F \middle| \hat{H}_1^0 \right] \\
&\le \frac{\mu^2 L_3}{12\, d} \sum_{i=0}^{r(k)-1} \eta^i + \eta^{r(k)} \sqrt{d} L_2 \left\| x_k - x_{k-1} \right\| + \eta^{r(k)} \mathbb{E}_{\hat{H}_{k-1}^0}\left[\mathbb{E}_{\hat{H}_{k-1}^{r(k-1)}}\left[\left\| \nabla^2 f(x_{k-1}) - \hat{H}_{k-1}^{r(k-1)} \right\|_F \middle| \hat{H}_1^0, \hat{H}_{k-1}^0 \right] \right] \\
&\le \frac{\mu^2 L_3}{12\, d} \left(\sum_{i=0}^{r(k)-1} \eta^i + \eta^{r(k)} \sum_{i=0}^{r(k-1)-1} \eta^i \right) + \eta^{r(k)} \sqrt{d} L_2 \left\| x_k - x_{k-1} \right\| + \eta^{r(k)+r(k-1)} \mathbb{E}_{\hat{H}_{k-1}^0}\left[\left\| \nabla^2 f(x_{k-1}) - \hat{H}_{k-1}^0 \right\|_F \middle| \hat{H}_1^0 \right] \\
&\le \frac{\mu^2 L_3}{12\, d} \sum_{j=k-1}^{k} \left(\eta^{\sum_{z=j+1}^{k} r(z)} \sum_{i=0}^{r(j)-1} \eta^i \right) + \sqrt{d} L_2 \sum_{j=k-1}^{k} \left(\eta^{\sum_{z=j}^{k} r(z)} \left\| x_j - x_{j-1} \right\| \right) \\
&\quad + \eta^{\sum_{j=k-1}^{k} r(j)} \mathbb{E}_{\hat{H}_{k-2}^{r(k-2)}}\left[\left\| \nabla^2 f(x_{k-2}) - \hat{H}_{k-2}^{r(k-2)} \right\|_F \middle| \hat{H}_1^0 \right] \\
&\le \frac{\mu^2 L_3}{12\, d} \sum_{j=1}^{k} \left(\eta^{\sum_{z=j+1}^{k} r(z)} \frac{1-\eta^{r(j)}}{1-\eta} \right) + \sqrt{d} L_2 \sum_{j=2}^{k} \left(\eta^{\sum_{z=j}^{k} r(z)} \left\| x_j - x_{j-1} \right\| \right) + \eta^{\sum_{j=1}^{k} r(j)} \left\| \nabla^2 f(x_1) - \hat{H}_1^0 \right\|_F \\
&\le \eta^{r(k)} \left\{ \frac{\mu^2 L_3}{12\, d} \left[\sum_{j=1}^{k-1} \left(\eta^{\sum_{z=j+1}^{k-1} r(z)} \frac{1-\eta^{r(j)}}{1-\eta} \right) - \frac{1}{1-\eta} \right] + \sqrt{d} L_2 \sum_{j=2}^{k} \left(\eta^{\sum_{z=j}^{k-1} r(z)} \left\| x_j - x_{j-1} \right\| \right) \right. \\
&\quad \left. + \eta^{\sum_{j=1}^{k-1} r(j)} \left\| \nabla^2 f(x_1) - \hat{H}_1^0 \right\|_F \right\} + \frac{1}{1-\eta} \frac{\mu^2 L_3}{12\, d}. \qquad (15)
\end{aligned}$$

is bounded with probability $(1-\delta_S)(1-\delta_{\mu,r})$ by

$$\begin{aligned}\left\|\nabla^2 f(x)-\hat{H}_S^r\right\| &\leq \left\|\nabla^2 f(x)-\nabla^2 f_S(x)\right\| \\ &\quad+\left\|\nabla^2 f_S(x)-\hat{H}_S^r\right\| \\ &\leq 4L_1\sqrt{\frac{\log\left(2\,d/\delta_S\right)}{|S|}}+\epsilon_{\mu,r},\end{aligned}$$

where $\epsilon_{\mu,r}$ and $\delta_{\mu,r}$ can be made arbitrarily small by choosing μ and r according to.

Proof: The scaling coefficient in the definition of $\hat{H}_S^r$ is needed to reweight the addends. After using triangular inequality, we apply the memory-less case of Theorem 1 together with the fact that the spectral norm is smaller than the Frobenius norm, and we bound the error due to subsampling using Lemma 8 in [26], based on Bernstein's inequality. ■

Proposition 2: The approximation error of the global gradient estimator at the server

$$\hat{g}_S=\frac{1}{|S|}\sum_{(i,j)\in S}\frac{m}{nm_i}\hat{g}_{ij}$$

is bounded with probability $(1-\delta_S)$ by

$$\begin{aligned}\|\nabla f(x)-\hat{g}_S\| &\leq \|\nabla f(x)-\nabla f_S(x)\|+\|\nabla f_S(x)-\hat{g}_S\| \\ &\leq 4\sqrt{2}L_0\sqrt{\frac{\log\left(2\,d/\delta_S\right)+1/4}{|S|}}+\frac{dL_2\mu^2}{6}.\end{aligned}$$

Proof: The proof is similar to the previous one, but in this case we use Lemma 6 in [26] and the bound (6). ■

Remark 9: The approximation error due to subsampling is governed by the cardinality of S, i.e. by the overall number of data samples considered by all clients. This implies that individual participants can use different percentages of their local data, as long as the total amount of samples considered by the union of the client is sufficiently big. This allows to implement load balancing strategies, where the number of data samples processed by each client is proportional to its computational power. By assigning the number of function queries according to the individual capabilities one can address the problem of client heterogeneity and prevent stragglers from slowing down the training process.

REFERENCES

[1] S. Liu et al., "A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications," *IEEE Signal Process. Mag.*, vol. 37, no. 5, pp. 43–54, Sep. 2020.

[2] G. Hinton, "The forward-forward algorithm: Some preliminary investigations," 2022, *arXiv:2212.13345*.

[3] W. Fang, Z. Yu, Y. Jiang, Y. Shi, C. N. Jones, and Y. Zhou, "Communication-efficient stochastic zeroth-order optimization for federated learning," *IEEE Trans. Signal Process.*, vol. 70, pp. 5058–5073, 2022.

[4] D. Hajinezhad, M. Hong, and A. Garcia, "Zeroth order nonconvex multi-agent optimization over networks," Feb. 2019, *arXiv:1710.09997*.

[5] W. Lu et al., "ZooPFL: Exploring black-box foundation models for personalized federated learning," Oct. 2023, *arXiv:2310.05143*.

[6] Y. Shu, X. Lin, Z. Dai, and B. K. H. Low, "Federated zeroth-order optimization using trajectory-informed surrogate gradients," Aug. 2023, *arXiv:2308.04077*.

[7] H. Feng, T. Pang, C. Du, W. Chen, S. Yan, and M. Lin, "Does federated learning really need backpropagation?," May 2023, *arXiv:2301.12195*.

[8] A. Maritan, S. Dey, and L. Schenato, "FedZeN: Towards superlinear zeroth-order federated learning via incremental hessian estimation," 2023, *arXiv:2309.17174*.

[9] M. Abramowitz et al. *Handbook of Mathematical Functions*, vol. 10. New York, NY, USA: Dover, 1968.

[10] A. Maritan and L. Schenato, "ZO-JADE: Zeroth-order curvature-aware distributed multi-agent convex optimization," *IEEE Control Syst. Lett.*, vol. 7, pp. 1813–1818, 2023.

[11] Y. Feng and T. Wang, "Stochastic zeroth-order Gradient and Hessian estimators: Variance reduction and refined bias bounds," *Inf. Inference: A J. IMA*, vol. 12, no. 3, Sep. 2023, Art. no. iaad014.

[12] K. Balasubramanian and S. Ghadimi, "Zeroth-order nonconvex stochastic optimization: Handling constraints, high dimensionality, and saddle points," *Found. Comput. Math.*, vol. 22, no. 1, pp. 35–76, Feb. 2022.

[13] T. Wang, "On sharp stochastic zeroth order Hessian estimators over Riemannian manifolds," *Inf. Inference: A J. IMA*, vol. 12, no. 2, pp. 787–813, 2023.

[14] J. Li, K. Balasubramanian, and S. Ma, "Stochastic zeroth-order Riemannian derivative estimation and optimization," *Math. Operations Res.*, vol. 48, no. 2, pp. 1183–1211, May 2023.

[15] T. D. Choi and C. T. Kelley, "Superlinear convergence and implicit filtering," *SIAM J. Optim.*, vol. 10, no. 4, pp. 1149–1162, 2000.

[16] D. Leventhal and A. Lewis, "Randomized Hessian estimation and directional search," *Optimization*, vol. 60, no. 3, pp. 329–345, Mar. 2011.

[17] S. Wang, F. Roosta, P. Xu, and M. W. Mahoney, "GIANT: Globally improved approximate newton method for distributed optimization," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2018, vol. 31, pp. 2332–2342.

[18] N. D. Fabbro, S. Dey, M. Rossi, and L. Schenato, "SHED: A Newton-type algorithm for federated learning based on incremental hessian eigenvector sharing," *Automatica*, vol. 160, 2024, Art. no. 111460.

[19] J. Nocedal and S. J. Wright, *Numerical Optimization*. Berlin, Germany: Springer, 1999.

[20] A. S. Berahas, L. Cao, K. Choromanski, and K. Scheinberg, "A theoretical and empirical comparison of gradient approximations in derivative-free optimization," *Found. Comput. Math.*, vol. 22, no. 2, pp. 507–560, 2022.

[21] Y. Nesterov and B. Polyak, "Cubic regularization of Newton method and its global performance," *Math. Program.*, vol. 108, no. 1, pp. 177–205, Aug. 2006.

[22] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2014, vol. 27, pp. 2933–2941. [Online]. Available: https://proceedings.neurips.cc/paper/2014/hash/17e23e50bedc63b4095e3d8204ce063b-Abstract.html

[23] C. Cartis, N. I. M. Gould, and P. L. Toint, "Adaptive cubic regularisation methods for unconstrained optimization. Part I: Motivation, convergence and numerical results," *Math. Program.*, vol. 127, no. 2, pp. 245–295, Apr. 2011.

[24] N. Tripuraneni, M. Stern, C. Jin, J. Regier, and M. I. Jordan, "Stochastic cubic regularization for fast nonconvex optimization," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2018, vol. 31, pp. 2904–2913. [Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/db1915052d15f7815c8b88e879465a1e-Abstract.html

[25] Z. Wang, Y. Zhou, Y. Liang, and G. Lan, "A note on inexact gradient and hessian conditions for cubic regularized Newton's method," *Operations Res. Lett.*, vol. 47, no. 2, pp. 146–149, Mar. 2019.

[26] J. M. Kohler and A. Lucchi, "Sub-sampled cubic regularization for non-convex optimization," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1895–1904. [Online]. Available: https://proceedings.mlr.press/v70/kohler17a.html

[27] J. Blackard, "Covertype," *UCI Mach. Learn. Repository*, 1998, doi: 10.24432/C50K5N.

[28] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[29] H. Fanaee-T, "Bike sharing dataset," *UCI Mach. Learn. Repository*, 2013, doi: 10.24432/C5W894.

[30] J. T. Barron, "A general and adaptive robust loss function," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4331–4339.
[31] L. Deng, "The MNIST database of handwritten digit images for machine learning research [Best of the Web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.
[32] Y. Chikuse and Y. Chikuse, *Statistics on Special Manifolds*, vol. 1. Berlin, Germany: Springer, 2003.
[33] F. Mezzadri, "How to generate random matrices from the classical compact groups," Feb. 2007. [Online]. Available: http://arxiv.org/abs/math-ph/0609050

ALESSIO MARITAN (Student Member, IEEE) received the bachelor's degree in information engineering and the master's degree in control systems engineering from the University of Padova, Padova, Italy, in 2020 and 2022, respectively. He is currently working toward the Ph.D. degree in information engineering with the University of Padova. His current research interests include distributed optimization, federated learning, and predictive maintenance.

LUCA SCHENATO (Fellow, IEEE) received the Dr. Eng. degree in electrical engineering from the University of Padova, Padova, Italy, in 1999, and the Ph.D. degree in electrical engineering and computer sciences from the University of California (U.C. Berkeley), Berkeley, CA, USA, in 2003. He held a post-doctoral position in 2004 and a Visiting Professor position during 2013–2014 at U.C. Berkeley. He is currently a Full Professor with the Information Engineering Department, University of Padova, Padova, Italy. His interests include networked control systems, multi-agent systems, wireless sensor networks, distributed optimization, and synthetic biology. He has been awarded the 2004 Researchers Mobility Fellowship by the Italian Ministry of Education, University and Research (MIUR), the 2006 Eli Jury Award in U.C. Berkeley and the EUCA European Control Award in 2014, and IEEE Fellow in 2017. He was an Associate Editor for IEEE TRANSACTIONS ON AUTOMATIC CONTROL, from 2010 to 2014 and he is currently Senior Editor for IEEE TRANSACTIONS ON CONTROL OF NETWORK SYSTEMS and Associate Editor for *Automatica*.

SUBHRAKANTI DEY (Fellow, IEEE) received the Ph.D. degree from the Department of Systems Engineering, Research School of Information Sciences and Engineering, Australian National University, Canberra, ACT, Australia, in 1996. He is currently a Professor and the Head of the Signals and Systems Division, Department of Electrical Engineering, Uppsala University, Uppsala, Sweden. He has also held professorial positions at NUI Maynooth, Ireland, and University of Melbourne, Parkville, VIC, Australia. His current research interests include networked control systems, distributed machine learning and optimization, wireless communications and networks and signal processing for sensor networks. He is a Senior Editor of IEEE TRANSACTIONS ON CONTROL OF NETWORK SYSTEMS and IEEE CONTROL SYSTEMS LETTERS, and an Associate Editor for *Automatica*.

Open Access funding provided by 'Universit? degli Studi di Padova' within the CRUI CARE Agreement