2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

FedZeN: Quadratic convergence in zeroth-order federated learning via incremental Hessian estimation

Alessio Maritan[1], Subhrakanti Dey[2], and Luca Schenato[1]

Abstract— **Federated learning is a distributed learning framework that allows a set of clients to collaboratively train a model under the orchestration of a central server, without sharing raw data samples. Although in many practical scenarios the derivatives of the objective function are not available, only few works have considered the federated zeroth-order setting, in which functions can only be accessed through a budgeted number of point evaluations. In this work we focus on convex optimization and design the first federated zeroth-order algorithm to estimate the curvature of the global objective, with the purpose of achieving superlinear convergence. We take an incremental Hessian estimator whose error norm converges linearly in expectation, and we adapt it to the federated zeroth-order setting, sampling the random search directions from the Stiefel manifold for improved performance. Both the gradient and Hessian estimators are built at the central server in a communication-efficient and privacy-preserving way by leveraging synchronized pseudo-random number generators. We provide a theoretical analysis of our algorithm, named FedZeN, proving local quadratic convergence with high probability and global linear convergence up to zeroth-order precision. Numerical simulations confirm the superlinear convergence rate and show that our algorithm outperforms the federated zeroth-order methods available in the literature.**

Index Terms— **Federated learning, zeroth-order optimization, incremental Hessian estimator, convex optimization.**

I. Introduction

Federated learning (FL) is a large-scale learning framework that allows multiple users to collaboratively train machine learning models while preserving the individual privacy. The goal is to expose the model to as much data as possible, achieving better generalization capabilities than if each client trains a separate model on his own data. Clients never transmit their raw data samples over the network, but rather exchange model updates with a central orchestrating server. This can dramatically reduce the communication cost of the learning process and provides some degrees of data security, which can be further improved by incorporating mechanisms such as differential privacy and homomorphic encryption. Moreover, the distributed nature of FL allows to overcome the limited scalability of the standard centralized setting, in which all the training data must be gathered and processed at a single machine with enough computational power and storage resources. For these reasons FL is the tool of choice when the training data is naturally distributed in form of data islands, which often happens in networks of smartphones, IoT sensors or other devices.

In many relevant cases, such as simulation-based or black-box optimization, the derivatives of the objective functions may be expensive or infeasible to obtain [1]. Most of the existing federated learning algorithms, including the well-known FedAvg [2], are gradient-based and thus cannot be applied in such situations. A possible solution is offered by the class of zeroth-order (ZO) algorithms, that do not require any knowledge of the function derivatives. Rather, they only need the objective to be evaluated at certain query points, and they estimate derivatives by mean of finite-differences along a set of search directions. We address the reader to [3] for a survey on general zeroth-order optimization, and below we briefly review the ZO federated algorithms available in the literature: FedZO [4] is a zeroth-order version of FedAvg; ZONE-S [5] is a primal-dual algorithm in which at each iteration only one client is active, and the central server minimizes an augmented Lagrangian function; BAFFLE [6] uses a stochastic gradient estimator based on Stein's identity and focuses on the privacy aspect; AsyREVEL [7] addresses the vertical FL scenario, while this work concerns the horizontal FL setting.

Remarkably, none of the above algorithms considers the curvature of the objective function, missing out on the possibility to greatly improve the convergence rate. In fact, preconditioning with the Hessian matrix often leads to larger improvements per iteration and consequently much fewer iterations needed to converge. This is especially desirable in FL, where many communication rounds are generally needed, and could sensibly reduce bandwidth consumption and idle time. ZO-JADE [8], which is the only distributed zeroth-order algorithm to exploit the curvature information, estimates both the gradient and the diagonal of the Hessian matrix computing central-differences along the canonical basis. However, neglecting the off-diagonal elements of the Hessian may lead to suboptimal performance when the objective function is highly skewed. Moreover, ZO-JADE is designed for a general mesh network and does not take full advantage of the star topology of the federated setting.

Looking outside the zeroth-order literature, there are two second-order federated algorithms that provide superlinear convergence, namely FedNL [9] and SHED [10]. However both do not support approximate derivatives, preventing straightforward zeroth-order implementations where the exact gradient and Hessian are replaced with estimates.

[1] A. Maritan and L. Schenato are with the Department of Information Engineering, University of Padova, Italy. Email: alessio.maritan@phd.unipd.it, schenato@dei.unipd.it.

[2] Subhrakanti Dey is with the Department of Electrical Engineering, Uppsala University, Sweden. Email: subhrakanti.dey@angstrom.uu.se

Contributions: Motivated by the absence of federated zeroth-order algorithms which leverage the curvature information, in this paper we design a novel algorithm, named FedZeN (Zeroth-order Newton). We focus on convex optimization problems and aim to achieve superlinear convergence, which requires knowledge of the full Hessian matrix. For this reason, we extend the randomized incremental estimator proposed in [11] to make it suitable for federated zeroth-order implementation. In particular, we exploit synchronized pseudo-random number generators to sample a common set of r search direction at all the nodes. The clients query their local functions according to the search directions, compute a set of coefficients needed to build both the gradient and Hessian estimators, and send them to the central server. The latter updates the model parameters using a Newton-type method, which is known to be significantly faster than first-order methods. The approximation error due to the zeroth-order estimation is handled using either appropriate regularization or an eigenvalue clipping safeguarding.

Below we list the main novelties and distinguishing features of the proposed method. *(i)* We devise a federated incremental estimator of the full Hessian matrix, which enables tackling federated optimization problems using second-order methods even when the exact derivatives are not available. Our estimator is the distributed zeroth-order counterpart of the one proposed in [11], that converges almost surely to the true Hessian and whose squared error norm goes to zero linearly in expectation. We propose to generate the search directions needed by the estimator by uniformly sampling the Stiefel manifold, which empirically provides better accuracy and enables the use of an excellent gradient estimator. *(ii)* We design FedZeN, the first federated zeroth-order algorithm to estimate and exploit the Hessian of the global objective function. We provide a theoretical analysis of the algorithm, proving local quadratic convergence with high probability and global linear convergence up to zeroth-order precision. Our numerical simulations show that FedZeN outperforms the existing federated zeroth-order algorithms and exhibits superlinear convergence. *(iii)* The proposed distributed derivative estimation procedure naturally addresses some important concerns in federated learning. The first is non-identically distributed data: the algorithm can be applied to pools of clients with heterogeneous data distributions and is unaffected by client drift. The second is privacy: if the internal seed of the pseudo-random generators is kept private, the proposed procedure hides the estimated derivatives from potential external eavesdroppers. Regarding the computational and communication costs, at each iteration clients only need to evaluate their local function at $2r+1$ query points and transmit to the central server $d+r$ scalar values, where d is the dimension of the problem. The design parameter r is independent from the dimension of the problem, making the algorithm suitable for client devices with limited resources.

Notation: We denote with I_d the d-dimensional identity matrix and with $\mathbb{E}[\cdot]$ the expectation. Given a matrix, $\|\cdot\|$ is the spectral norm while $\|\cdot\|_F$ is the Frobenius norm. For brevity, we indicate with $[n]$ the set of integers $\{1,\dots,n\}$, and with $\mathcal{U}(\mathbb{S})$ the uniform distribution on the unit sphere $\mathbb{S}=\{z\in\mathbb{R}^d \text{ s.t. } \|z\|=1\}$.

II. Problem Formulation

We consider the horizontal federated learning setting, where local datasets consist of samples with different IDs that belong to the same feature space. Data is not independently or identically distributed, i.e. the data distribution can vary across clients. We consider a federation of n clients wanting to collaboratively train a model parametrized by $x\in\mathbb{R}^d$ by solving the empirical risk minimization

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

Here $f_i(x):\mathbb{R}^d\to\mathbb{R}$ is the loss function of client i, and the global average $f(x)$ satisfies the following assumption, which is standard in convex optimization.

Assumption 1: The global cost is m-strongly convex and twice continuously differentiable with Lipschitz derivatives, i.e. there exist positive constants m, L_0, L_1, L_2 such that $\forall x,y\in\mathbb{R}^d$

$$\begin{aligned}\|f(x)-f(y)\|&\le L_0\|x-y\|,\\ \left\|\nabla^2 f(x)-\nabla^2 f(y)\right\|&\le L_2\|x-y\|,\\ mI_d\le\nabla^2 f(x)&\le L_1 I_d.\end{aligned}$$

In order to apply our distributed derivative estimation technique, we assume that all the clients and the central server own the same deterministic pseudo-random number generator (PRNG). This trick allows to generate common vectors at all the devices by just periodically sending seeds or internal states to ensure synchronization, greatly reducing the communication overhead of the algorithm. The presence of PRNGs is a mild requirement and is assumed also in other works, such as [6] and [12].

Assumption 2 (Synchronized PRNG): All the clients and the central server are equipped with the same pseudo-random number generator, whose output sequence can be determined a priori by having knowledge of the internal seed.

III. Zeroth-order Oracles

Zeroth-order estimators approximate derivatives by means of finite-differences between values taken by the objective function at given query points. The latter are chosen in a neighborhood of the current model parameters, fixing a set of search directions and a small scalar $\mu>0$. The value of the finite-difference granularity μ, also called discretization or smoothing parameter, is usually chosen based on the specific application and on the machine precision. When choosing the derivative estimator, one must consider the level of accuracy but also the associated computational cost. In fact, the common assumption in the ZO optimization field is that not only the exact derivative is inaccessible or prohibitive to obtain, but also that function evaluations are expensive and possibly budgeted in number.

A. Incremental randomized Hessian estimator

Most ZO algorithms avoid estimating the Hessian matrix, as this typically requires much more function evaluations than gradient estimation. For example, to approximate all the $d(d+1)/2$ distinct entries of the Hessian using forward finite-differences along the canonical basis $\{e_1, \dots, e_d\}$, one has to query the objective function at the points x, $\{x+\mu e_i\}$, $\{x+\mu e_i+\mu e_j\}$ with $i, j \in [d]$, for a total of $(d+1)(d/2+1)$ evaluations [13]. If instead the ij-th entry is estimated using a 4-points scheme, e.g. $\{x \circ \mu e_i \diamond \mu e_j\}$ for $\circ, \diamond \in \{+, -\}$, then the number of function evaluations grows to $2d(d+1)$.

To avoid necessarily computing $O(d^2)$ function values, one can resort to randomized estimation schemes, that allow to choose an arbitrary number r of search directions at the cost of a possibly larger approximation error. An example of randomized Hessian estimator is the one proposed in [14], which performs $4r^2$ function queries along orthogonal directions sampled from the Stiefel manifold. The error norm of this estimator is shown to decrease sublinearly with the number of search directions, and to suddenly drop only when $r = d$. Based on the second-order Stein's identity, [15] develops some unbiased estimators of the Hessian of a Gaussian-smoothed version of the objective function. However, as shown in our simulations, in practice these estimators still require too many function evaluations to provide an acceptable estimate. In fact, being sample averages over the set of search directions, by the law of large numbers their variance decreases with sublinear rate $1/r$.

In this work we employ an Hessian estimator based on a different principle. Given an initial symmetric matrix $H^0 \in \mathbb{R}^{d\times d}$, we apply r times the update proposed in [11]

$$H^k = H^{k-1} + (u^T \nabla^2 f(x) u - u^T H^{k-1} u) u u^T, \quad (2)$$

where $u \sim \mathcal{U}(\mathbb{S})$. The idea behind this iterative formula is to add a rank-one matrix such that the updated estimator matches the true Hessian along the direction u. The recursion (2) satisfies the linear convergence condition

$$\mathbb{E}\left[\left\|H^k - \nabla^2 f(x)\right\|_F^2\right] \leq \eta \left\|H^{k-1} - \nabla^2 f(x)\right\|_F^2, \quad (3)$$

where $\eta = 1-2/(d^2+2d)$, and asymptotically H^k converges almost surely to the exact Hessian [11]. Since the true Hessian is obviously not available, as also mentioned in [11] Hessian-vector products can be estimated using finite-differences, which makes the update (2) ideal for zeroth-order optimization. In particular, in FedZeN we approximate the directional curvature as

$$u^T \nabla^2 f(x) u \approx \frac{f(x+\mu u) - 2f(x) + f(x-\mu u)}{\mu^2},$$

and therefore computing the Hessian estimator H^r requires $2r+1$ function evaluations.

Differently from the other estimators available in the literature, (2) is an incremental formula. This lends itself to warm-start the estimator by initializing it with the estimate from the previous iteration. This is especially useful when the Hessian is constant or slowly changing and when approaching the global optimum, so that only few updates per iteration are needed. On the contrary, the estimators in [14] and [15] are designed to be reset at each iteration and not to exploit past estimates, and in this way they lose all previously collected information.

B. Stiefel sampling

The first and fundamental step to build the randomized Hessian estimator is to choose a set of search directions $\{u_j \sim \mathcal{U}(\mathbb{S})\}$, $j \in [r]$. The standard way to generate these directions is to sample r vectors from $\mathcal{N}(0, I_d)$ and project them on the unit hypersphere by dividing by their norm. However, this sampling procedure is only asymptotically optimal, and for limited values of r may cause to oversample some regions of the space while barely exploring others. To address this problem, in FedZeN we generate a matrix uniformly sampled from the Stiefel manifold

$$V_{r,d} = \{U \in \mathbb{R}^{d\times r} \text{ such that } U^T U = I_r\}$$

and use its $r \leq d$ columns as search directions. Intuitively, since this set of vectors is orthogonal it should be more evenly spread in the search space, thus maximizing the information gain and reducing redundancy. Most importantly, the marginal distribution of these vectors is $\mathcal{U}(\mathbb{S})$, which is the one required by the Hessian estimator. An explanation of why this last fact holds is provided in [14], which first introduced Stiefel sampling for zeroth-order optimization. The procedure to uniformly sample from the Stiefel manifold is based on Theorem 2.2.1 of [16], stating that a matrix $U = [u_1, \dots, u_r]$ uniformly distributed on $V_{r,d}$ can be expressed as

$$U = X(X^T X)^{-1/2},\ X \in \mathbb{R}^{d\times r} \text{ s.t. } X_{ij} \overset{i.i.d.}{\sim} \mathcal{N}(0,1). \quad (4)$$

In our tests, generating the search directions according to (4) instead of using non-orthogonal directions considerably improves the accuracy of the Hessian estimator, especially for small values of r. In case $r > d$, we generate $\lceil r/d \rceil$ separate orthogonal matrices.

C. Zeroth-order gradient estimator

In this work we adopt the gradient estimator

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

where the orthonormal search directions $\{u_j \sim \mathcal{U}(\mathbb{S})\}$, $j \in [d]$ are a subset of the directions used to build the Hessian estimator. The choice of this gradient estimator is motivated by several reasons. First, as constructing the Hessian estimator H^r involves querying the objective function at the points $\{x \pm \mu u_j\}$, $j \in [r]$, it makes sense to reuse these function values to estimate also the gradient for free. Second, numerical simulations show that in practice r must be at least greater than the dimension of the problem to get a satisfactory approximation of the Hessian, which guarantees that d orthonormal search directions are always available. Finally, by estimating the gradient along an orthonormal basis we

can provide deterministic guarantees on the approximation error, as shown by the following Lemma.

Lemma 1 (Error of the gradient estimator): If the set of search directions used to build the gradient estimator (5) is an orthonormal basis $U = \{u_1, \dots, u_d\}$, then $\forall x \in \mathbb{R}^d$

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

Proof: Define $\Delta_i = \nabla^2 f(x + t\mu u_i) - \nabla^2 f(x - t\mu u_i)$. Since U is a basis of $\mathbb{R}^d$ and $\|u_i\| = 1\ \forall i \in [d]$, using Taylor expansion with integral remainder we get

$$\begin{aligned}
\|\nabla f(x) - g(x)\| &= \left\| \sum_{i=1}^{d} (\nabla f(x)^T u_i) u_i - g(x) \right\| \\
&\leq \sum_{i=1}^{d} \left\| \left(\nabla f(x)^T u_i - \frac{f(x + \mu u_i) - f(x - \mu u_i)}{2\mu} \right) u_i \right\| \\
&= \sum_{i=1}^{d} \left| \frac{\mu}{2} u_i^T \int_0^1 (1-t)\Delta_i dt\ u_i \right| \|u_i\| \\
&\leq \sum_{i=1}^{d} \left| \frac{\mu}{2} \|u_i\|^2 \int_0^1 (1-t) L_2 \|2t\mu u_i\|\, dt \right| = \frac{d\mu^2 L_2}{6}.
\end{aligned}$$

■

Other properties of the estimator (5) can be found in [14]. In comparison, commonly used randomized gradient estimators such as the ones employed in FedZO [4] and ZONE-S [5] do not search along orthogonal directions and are associated with larger variance and approximation errors [14] [17].

IV. Federated Hessian estimation

In this section we describe how to build the randomized estimators in a communication-efficient way by taking advantage of the star topology of the network, and we introduce the proposed federated zeroth-order algorithm. We use the subscript k where needed to denote the value of a variable at the k-th iteration of the algorithm.

According to Assumption 2, each client can access a pseudo-random number generator, and all the generators can be synchronized by making the central server broadcast a common internal seed at the first iteration. In the initialization step of the algorithm the master also chooses the initial Hessian estimator, which can be any symmetric matrix $H_1^0 \in \mathbb{R}^{d\times d}$, for example $H_1^0 = \beta I_d$ with $\beta > 0$. At each iteration, both the clients and the master use their PRNG to generate a common random matrix $X \in \mathbb{R}^{d\times r}$ such that $X_{ij} \overset{i.i.d.}{\sim} \mathcal{N}(0,1)$. From the latter, using Stiefel sampling (4) they compute the set of vectors $\{u_j \sim \mathcal{U}(\mathbb{S})\}$, $j \in [r]$, which is the same at all the nodes. The central server broadcasts the current decision vector x_k. Each client i evaluates its local function at the points x_k and $\{x_k \pm \mu u_j\}$, $j \in [r]$ to compute the d gradient coefficients

$$c_{ij} = \frac{f_i(x_k + \mu u_j) - f_i(x_k - \mu u_j)}{2\mu}, \tag{6}$$

and the r directional curvatures

$$b_{ij} = \frac{f_i(x_k + \mu u_j) - 2f_i(x_k) + f_i(x_k - \mu u_j)}{\mu^2}. \tag{7}$$

These $d + r$ scalars are sent to the master, where they are averaged over the set of clients. Using the fact that the search directions are the same for all nodes, the central server is able to build the derivative estimators, where the Hessian estimator is updated starting from $H_k^0 = H_{k-1}^r$ when $k > 1$.

$$g_k = \sum_{j=1}^{d} \left(\frac{1}{n} \sum_{i=1}^{n} c_{ij} \right) u_j, \tag{8}$$

$$H_k^j = H_k^{j-1} + \left(\frac{1}{n} \sum_{i=1}^{n} b_{ij} - u_j^T H_k^{j-1} u_j \right) u_j u_j^T,\ j \in [r]. \tag{9}$$

To get the new model parameters, the central server performs a step of a Newton-type method, which requires H_k^r to be invertible and positive-definite. To ensure that this constraint is satisfied and improve the robustness of the algorithm with respect to estimation errors, we consider two possible safeguarding mechanisms. The first is a simple and computationally inexpensive regularization, where a scalar multiple of the identity matrix is added to H_k^r:

$$Z_k = (H_k^r + \rho I_d)^{-1}, \quad \rho > 0. \tag{10}$$

The second is eigenvalue clipping, based on the spectral decomposition $H_k^r = QDQ^T$, where D is the diagonal matrix whose entries are the eigenvalues of the estimator, and Q is orthonormal since the estimator is real and symmetric. Although spectral decomposition may be computationally demanding for high-dimensional problems, this operation is performed at the central server and allows to easily compute the inverse of the approximate Hessian as

$$Z_k = Q\bar{D}Q^T,\ \bar{D}_{ii} = 1/\max\left(\lambda_{\min}, \min(D_{ii}, \lambda_{\max})\right). \tag{11}$$

The above formula projects the eigenvalues in the interval $[\lambda_{\min}, \lambda_{\max}]$ before the inversion, where $0 < \lambda_{\min} < \lambda_{\max}$ are design parameters. Finally, the target variable is updated according to the approximate Newton step

$$x_{k+1} = x_k - \alpha Z_k g_k. \tag{12}$$

The learning rate α can be either a constant value or follow an increasing schedule. The latter option is preferable to promote algorithmic stability and prevent oscillations, as at the beginning the norm of the gradient is usually large and the Hessian approximation may not be sufficiently accurate. After a few damped iterations it is desirable to bring up the stepsize to $\alpha = 1$, which is the optimal value for the exact Newton method. This is justified by the fact that once the decision vector x_k begins to settle the Hessian becomes almost constant. As a consequence, in virtue of (3) the Hessian estimator converges linearly to the true Hessian, and one recovers an almost perfect Newton step. The pseudocode of the algorithm summarizes the main steps.

We now emphasize some of the strengths of FedZeN.

Algorithm 1 FedZeN

Initialization:
Central server (CS): Choose $x_1 \in \mathbb{R}^d$, $r > 0$, $H_1^0 \in \mathbb{R}^{d\times d}$ symmetric.
Clients: Choose $\mu \geq 0$.
CS: Broadcast r and a random seed for the PRNGs.
for each iteration $k = 1, 2, \dots$ **do**
All nodes: Generate $[u_1 \dots u_r]$ using (4).
CS: Broadcast x_k.
for each client $i \in [n]$ **do**
Compute $c_{ij}, b_{ij}\, j \in [r]$ using (6, 7).
end for
CS: If $k > 1$ set $H_k^0 = H_{k-1}^r$.
CS: Compute g_k, H_k^r using (8, 9).
CS: Compute Z_k using either (10) or (11).
CS: $x_{k+1} = x_k - \alpha Z_k g_k$.
end for

(i) The core of the algorithm is the distributed estimation of the full Hessian matrix of the global objective function, which is used for preconditioning. By considering also the off-diagonal entries of the Hessian, the algorithm can preserve fast convergence also in case of highly skewed objectives. The estimation of both gradient and Hessian only requires to evaluate the local functions at an arbitrary number of points, making the algorithm suited for black-box optimization problems in which the exact derivatives are not available.

(ii) The algorithm is conceptually simple and straightforward to be implemented, as it is self-contained and differently from [5] does not involve solving auxiliary subproblems. Moreover, it requires a small amount of parameter tuning, as the only design parameters are r, α and the ones required for robust matrix inversion, i.e. either ρ or the pair $\lambda_{\min}, \lambda_{\max}$. In particular, r determines both the number of function queries and the number of scalars transmitted by each client, allowing to adapt the computational and communication cost of the algorithm to the available resources.

(iii) Differently from other works which require statistical similarity between the local functions, here we do not make any assumption about the relationship between the data distributions of the clients. Indeed, by estimating the derivatives of the global objective function, FedZeN naturally handles data heterogeneity between clients. Moreover, since it does not perform multiple local iterations, it does not suffer from client drift.

(iv) Since g_k and H_k^r cannot be obtained without knowing the vectors $\{u_j\}$, which in turn require knowing the seed of the PRNG, our method offers an additional level of privacy provided that the initial seed is transmitted on a secure channel. In this case, a possible eavesdropper on the communication channel between the participants and the central server would not be able to obtain the estimated derivatives, as the coefficients c_{ij} and b_{ij} are useless by themselves. This is very important for data security, since in some cases it is possible to reconstruct raw data samples from shared gradients [18].

V. Convergence analysis

In this section we derive theoretical guarantees on the performance of the proposed algorithm. Our analysis is inspired by the one in [19], which deals with subsampled Newton methods.

Remark 1: Below we derive rates of convergence up to zeroth-order precision, which is the smallest theoretically achievable accuracy. Once the zeroth-order estimation error becomes dominant, one can either reduce the finite-difference granularity μ or terminate the algorithm. We recall that the design parameter μ can be chosen arbitrarily small according to the available hardware, and for $\mu \to 0$ we have exact convergence.

Our first result, Theorem 1, concerns the improvement of the function value and shows that the algorithm enjoys global linear convergence up to zeroth-order precision.

Theorem 1 (Global linear convergence): Let Z_k be computed using the eigenvalue clipping formula (11), let $\alpha \leq 2\lambda_{\min}/L_1$ and define $\gamma = \frac{\alpha 2m}{\lambda_{\max}}\left(1 - \frac{L_1\alpha}{2\lambda_{\min}}\right)$. Then at each iteration of FedZeN the bound

$$f(x_{k+1}) - f(x^\star) \leq (1-\gamma)\left(f(x_k) - f(x^\star)\right) + O(\mu^2)$$

holds with probability 1, i.e. for each realization of the derivative estimators. The stepsize that maximizes γ is $\alpha^\star = \lambda_{\min}/L_1$, for which $\gamma(\alpha^\star) = (m\lambda_{\min})/(L_1\lambda_{\max})$.

Proof: We list the main steps: ① Taylor's expansion for functions with bounded Hessian, ② add and subtract $\nabla f(x_k)$, ③ Cauchy-Schwarz, ④ Lemma 1, the bound $\nabla f(x) \leq L_0\ \forall x \in \mathbb{R}^d$ provided by Assumption 1, and the fact that by construction $1/\lambda_{\max} \leq \|Z_k\| \leq 1/\lambda_{\min}$, ⑤ the assumption on α.

$$\begin{aligned}
f(x_{k+1}) &\overset{①}{\leq} f(x_k) + \nabla f(x_k)^T(-\alpha Z_k g_k) + \frac{L_1}{2}\left\|\alpha Z_k g_k\right\|^2 \\
&\overset{②}{=} f(x_k) - \alpha\nabla f(x_k)^T Z_k(g_k - \nabla f(x_k) + \nabla f(x_k)) \\
&\quad + \frac{L_1\alpha^2}{2}\left\|Z_k(g_k - \nabla f(x_k) + \nabla f(x_k))\right\|^2 \\
&\overset{③}{\leq} f(x_k) - \alpha\nabla f(x_k)^T\left(Z_k - \frac{L_1\alpha}{2}Z_k^2\right)\nabla f(x_k) \\
&\quad + \alpha\left\|\nabla f(x_k)\right\|\left\|Z_k\right\|\left\|\nabla f(x_k) - g_k\right\| + \frac{L_1\alpha^2}{2}\left\|Z_k^2\right\| \\
&\quad \times \left\|g_k - \nabla f(x_k)\right\|\left(\left\|g_k - \nabla f(x_k)\right\| + 2\left\|\nabla f(x_k)\right\|\right) \\
&\overset{④}{\leq} f(x_k) - \alpha\nabla f(x_k)^T Z_k^{1/2}\left(I - \frac{L_1\alpha}{2}Z_k\right)Z_k^{1/2}\nabla f(x_k) \\
&\quad + \frac{\alpha L_0}{\lambda_{\min}}\frac{dL_2\mu^2}{6} + \frac{L_1\alpha^2}{2\lambda_{\min}^2}\frac{dL_2\mu^2}{6}\left(\frac{dL_2\mu^2}{6} + 2L_0\right) \\
&\overset{⑤}{\leq} f(x_k) - \alpha\left\|\nabla f(x_k)\right\|^2\left(1 - \frac{L_1\alpha}{2\lambda_{\min}}\right)\frac{1}{\lambda_{\max}} + O(\mu^2).
\end{aligned}$$

Recalling that m-strong convexity implies $\|\nabla f(x)\|^2 \geq 2m(f(x) - f(x^\star))$ and subtracting $f(x^\star)$ from both sides the proof is concluded. ■

As usually done for Newton-type methods, we inspect the behaviour of the algorithm in a neighborhood of the optimal solution to derive a faster convergence rate.

Theorem 2 (Linear-quadratic local bound): Consider the generic k-th iteration of FedZeN where Z_k is computed using eigenvalue clipping (11) and the stepsize is $\alpha = 1$. Then the improvement towards the global minimum $x^\star$ satisfies

$$\|x_{k+1} - x^\star\| \leq \frac{L_2}{2\lambda_{\min}} \|x_k - x^\star\|^2 + \frac{dL_2\mu^2}{6\lambda_{\min}} + \frac{\left\|\nabla^2 f(x_k) - Z_k^{-1}\right\|}{\lambda_{\min}} \|x_k - x^\star\|.$$

Proof: We first isolate the contribution due to the approximation error of the derivative estimators.

$$\begin{aligned}
&\|x_{k+1} - x^\star\| = \|x_k - x^\star - Z_k g_k\| \\
&= \left\|Z_k \left(Z_k^{-1}(x_k - x^\star) - g_k + \nabla f(x_k) - \nabla f(x_k)\right)\right\| \\
&\leq \frac{1}{\lambda_{\min}} \left[\left\|Z_k^{-1}(x_k - x^\star) - \nabla f(x_k)\right\| + \|\nabla f(x_k) - g_k\|\right] \\
&\leq \frac{1}{\lambda_{\min}} \left[\left\|(Z_k^{-1} - \nabla^2 f(x_k))(x_k - x^\star)\right\| + \|\nabla f(x_k) - g_k\|\right] \\
&\quad + \frac{1}{\lambda_{\min}} \left\|\nabla^2 f(x_k)(x_k - x^\star) - \nabla f(x_k)\right\|.
\end{aligned}$$

Recalling that $\nabla f(x^\star) = 0$ and using the fundamental theorem of calculus and the Lipschitz property of the Hessian, we can bound the norm in the last term as

$$\begin{aligned}
&\|\nabla^2 f(x_k)(x_k - x^\star) + \nabla f(x^\star) - \nabla f(x_k)\| \\
&\leq \left\|\nabla^2 f(x_k)(x_k - x^\star) + \int_0^1 \frac{d}{dt} \nabla f(x_k + t(x^\star - x_k))\, dt\right\| \\
&= \left\|\int_0^1 \left[\nabla^2 f(x_k) - \nabla^2 f(x_k + t(x^\star - x_k))\right](x_k - x^\star)\, dt\right\| \\
&\leq \|x_k - x^\star\| \int_0^1 \left\|\nabla^2 f(x_k) - \nabla^2 f(x_k + t(x^\star - x_k))\right\|\, dt \\
&\leq \|x_k - x^\star\| \int_0^1 L_2 \|t(x_k - x^\star)\|\, dt \\
&= \frac{L_2 \|x_k - x^\star\|^2}{2}.
\end{aligned}$$

Using the above result and Lemma 1 we get the bound to be proved. ∎

The bound provided by Theorem 2 can be used to prove local quadratic convergence up to zeroth-order precision with high probability. To do so, from now on we allow a variable number of search directions at each iteration. To simplify the analysis, we bound with high probability the approximation error of the Hessian estimator by means of the following assumption.

Assumption 3: Fix a symmetric $H_k^0 \in \mathbb{R}^{d\times d}$, an accuracy $\epsilon_k > 0$ and a failure probability $\delta \in (0,1)$. Update the initial estimate H_k^0 along r search directions to obtain H_k^r, and compute Z_k using eigenvalue clipping (11) with $0 < \lambda_{\min} \leq m$ and $\lambda_{\max} \geq L_1$. We assume that there exist a finite-difference precision $\bar{\mu}(\epsilon_k, \delta)$ and a minimum number of search directions $\bar{r}(\epsilon_k, \delta, \mu)$ such that if $\mu \leq \bar{\mu}(\epsilon_k, \delta)$ and $r \geq \bar{r}(\epsilon_k, \delta, \mu)$ it holds $\mathbb{P}\left(\left\|\nabla^2 f(x_k) - Z_k^{-1}\right\| \geq \epsilon_k\right) \leq \delta$.

Remark 2: In the interest of space, we do not provide an explicit formula for $\bar{\mu}(\epsilon_k, \delta)$ and $\bar{r}(\epsilon_k, \delta, \mu)$ and leave it for future work. In particular, we plan to consider the estimation error $\left\|\nabla^2 f(x_k) - H_k^r\right\|$ for any value of $(\epsilon, \delta, \mu, H^0)$, obtaining stronger results that do not rely on the choice of the eigenvalue clipping parameters. This will involve quantifying the zeroth-order error and analysing how the latter builds up over the course of the estimator updates, allowing to replace Assumption 3 with rigorous bounds and provable conditions. Intuitively, by shrinking μ one can make the zeroth-order error negligible and approximate arbitrarily well the update (2). The latter is known to converge almost surely to the true Hessian, and the existence of $\bar{r}$ is guaranteed by the convergence rate (3).

Theorem 3 (Local quadratic convergence w.h.p.): Consider the assumptions of Theorem 2 and let Assumption 3 be satisfied. At each iteration k of FedZeN:

- If $\|g_k\| > \frac{dL_2\mu^2}{6}$, according to Assumption 3 choose $\delta \in (0,1)$, $\epsilon_k \leq \frac{1}{L_1}\left(\|g_k\| - \frac{dL_2\mu^2}{6}\right)$, and let $\bar{r}(\epsilon_k, \delta, \mu)$ be the corresponding minimum number of search directions. If $r > \bar{r}(\epsilon_k, \delta, \mu)$, then with probability $(1-\delta)$ the local convergence rate up to zeroth-order precision is quadratic:
$$\|x_{k+1} - x^\star\| \leq \frac{L_2 + 2}{2\lambda_{\min}} \|x_k - x^\star\|^2 + O(\mu^2).$$
- If $\|g_k\| \leq \frac{dL_2\mu^2}{6}$, then the suboptimality gap satisfies
$$\|x_k - x^\star\| \leq \frac{dL_2\mu^2}{3m} = O(\mu^2)$$
and one can use this as a stopping criterion.

Proof: In the first case, Assumption 3 guarantees that with probability $(1-\delta)$ it holds

$$\begin{aligned}
&\left\|\nabla^2 f(x_k) - Z_k^{-1}\right\| \leq \epsilon_k \leq \frac{\|g_k\| - \frac{dL_2\mu^2}{6}}{L_1} \\
&\leq \frac{\|\|g_k\| - \|\nabla f(x_k) - g_k\|\|}{L_1} \leq \frac{\|\nabla f(x_k)\|}{L_1} \leq \|x_k - x^\star\|.
\end{aligned}$$

Combining the above inequality with Theorem 2 we obtain the quadratic rate to be proved. In the second case, we have

$$\begin{aligned}
\|x_k - x^\star\| &\leq \frac{\|\nabla f(x_k)\|}{m} \leq \frac{\|g_k\| + \|\nabla f(x_k) - g_k\|}{m} \\
&\leq \frac{\|g_k\| + \frac{dL_2\mu^2}{6}}{m} \leq \frac{2}{m}\frac{dL_2\mu^2}{6}.
\end{aligned}$$

∎

Remark 3: The condition on ϵ_k used in Theorem 3 is implementable in practice by following these steps: *(i)* search along d orthonormal directions and build g_k, *(ii)* use the latter to compute the upper bound on ϵ_k, *(iii)* choose $r \geq \bar{r}(\epsilon_k, \delta, \mu)$, *(iv)* evaluate the function along the remaining $r - d$ directions to build H_k^r.

VI. Numerical results

A. Testing the distributed Hessian estimator

We start the numerical analysis by comparing the proposed distributed zeroth-order Hessian estimator to the main

Hessian estimators available in the literature. We consider the following competitors: *(i)* the identity matrix, which is implicitly used by the methods that only use gradient estimates, *(ii)* the Jacobi estimator employed in [8], which approximates the diagonal of the Hessian matrix, *(iii)* a distributed version of the randomized estimator based on the second-order Stein's identity proposed in [15], and *(iv)* a distributed version of the one based on Stiefel sampling, introduced by [14]. Differently from our Hessian estimator, all the aforementioned ones are thought to be reset at each iteration. To allow a complete comparison, we also implement incremental versions of the last two estimators where the latest estimate is used as starting point.

Figure 1 displays the evolution of the approximation error in case of constant Hessian, showing that our distributed zeroth-order version of (2) outperforms all the other estimators, including the incremental versions of the competitors. This happens because while the estimators [14] and [15] are sample averages, the update (2) imposes the correct curvature along each search direction. The plot shows the average errors over 100 random Hessian matrices. All algorithms perform the same number of function evaluations per iteration, namely $2d+1$, which is the amount of queries required by the deterministic Jacobi estimator [8]. Since the Hessian is known to be constant, the incremental versions of [14] and [15] compute the mean of all the past estimates, e.g. $H_k^{\text{inc}} = (H_k + (k-1)H_{k-1})/k$.

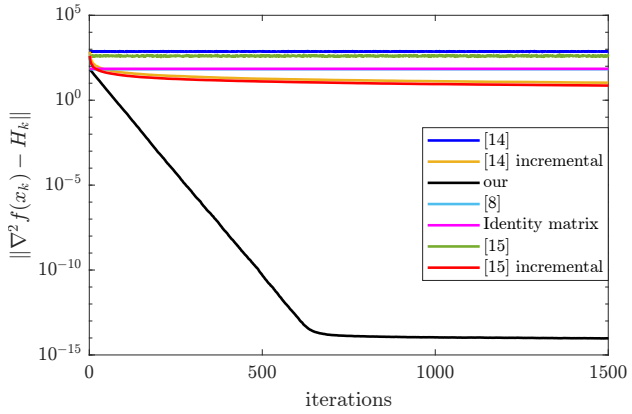

Fig. 1: Approximation error of various Hessian estimators in case of constant Hessian, averaging over 100 matrices.

B. Test of FedZeN

We empirically test the efficiency of the proposed FedZeN and compare it to the federated zeroth-order algorithms FedZO [4] and ZONE-S [5] and to a federated version of ZO-JADE [8]. Below we briefly introduce these competitors and describe the chosen hyperparameter configurations, that are the ones leading to the best performances in our test. *(i)* FedZO [4] is essentially a zeroth-order version of FedAvg, and we set learning rate $\eta = 0.1$, $H = 10$ local epochs and $b_2 = d$ perturbation directions. *(ii)* ZONE-S [5] selects only one active client at each iteration and minimizes an augmented Lagrangian function at the server. In our implementation we use use Nesterov accelerated gradient to minimize the Lagrangian, and in Figure 2 we show the average number of function queries per client. *(iii)* ZO-JADE [8] is designed for general mesh networks of agents and uses an estimate of the diagonal of the Hessian for preconditioning. In our federated implementation we set learning rate $\epsilon = 0.2$. *(iv)* For the proposed FedZeN we set $\lambda_{\min} = 10^{-3}$, $\lambda_{\max} = 10^4$, $\rho = 10^{-2}$ and $r = d$ search directions. The learning rate follows a simple increasing schedule: we start with a conservative $\alpha = 0.3$ and after 30 iterations we switch to $\alpha = 1$.

The test consists in binary classification via logistic regression, where two classes of the dataset Covertype [20] are evenly split over a pool of $n = 100$ clients. The local objectives are the regularized log-losses

$$f_i(x, \mathcal{D}_i) = \frac{1}{|\mathcal{D}_i|} \sum_{k=1}^{|\mathcal{D}_i|} \log\left(1 + \exp\left(-l_k [s_k^T \ 1] x\right)\right) + \frac{w}{2} \|x\|^2,$$

where $l_k \in \{-1, 1\}$ is the label associated to the sample $s_k \in \mathbb{R}^{d-1}$, $d = 55$ and $w > 0$. All algorithms are started from the same initial x_1. The normalized training loss shown in the plots is $(f(x) - f(x^\star))/|f(x^\star)|$, where $x^\star$ is the global minimum.

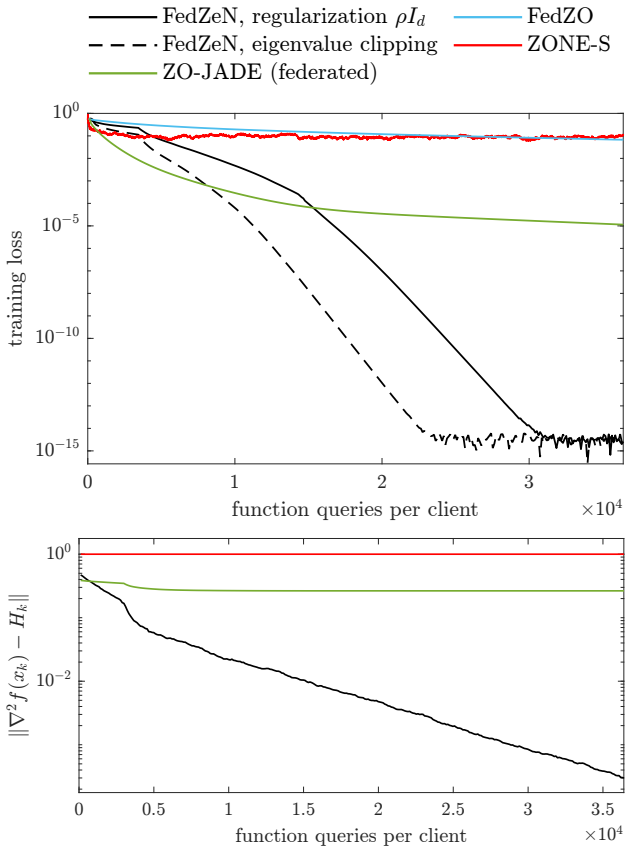

Fig. 2: Logistic regression using the dataset Covertype [20]. FedZeN (our) versus other federated zeroth-order algorithms.

Figure 2 clearly shows that FedZeN outperforms the other zeroth-order algorithms. In particular, ZONE-S settles

far from the optimal solution and FedZO, that relies only on gradient estimates, converges much more slowly than the algorithms that also exploit the Hessian. The federated version of ZO-JADE is initially in the lead, as it has an accurate estimate of the diagonal of the Hessian available from the first iteration. However, ZO-JADE is soon surpassed by our FedZeN, whose winning strategy is to invest the early function evaluations to estimate the full Hessian and then capitalize it. The speed of convergence is expressed in terms of number of function evaluations, which in the ZO optimization field are assumed to be the most expensive computations. Plotting the training loss against the number of iterations or the the number of scalars transmitted and received by each client, one obtains figures identical to the one shown. The plot on the bottom confirms the effectiveness of the incremental Hessian estimator employed in FedZeN when the target Hessian changes over time. Since methods that estimate only the gradient can be thought to use the identity matrix as Hessian estimator, in ZONE-S and FedZO we set $H_k = I_d \ \forall k$ just for reference.

VII. Conclusions

We have introduced a general procedure to estimate the global Hessian matrix in the federated learning setting when exact derivatives are not available and functions are only accessible through point evaluations. Under the mild assumption that all nodes own a pseudo-random number generator, we generate a common set of search directions at all the nodes, sampling from the Stiefel manifold for greater estimation accuracy. This allows to greatly reduce the communication complexity and conceal the estimated derivatives from external eavesdroppers. Since the Hessian estimator is incremental and builds upon past estimates, few function evaluations per iteration are required. This distributed estimation technique is the foundation of the proposed FedZeN, a zeroth-order algorithm for federated learning which is the first to approximate and leverage the Hessian matrix. FedZeN allows to tailor both the communication and the computational costs to the capabilities of the clients by selecting an appropriate number of search directions. Moreover, the algorithm is suited for federations of clients with heterogeneous data distributions. FedZeN comes with theoretical guarantees of global linear and local quadratic convergence up to zeroth-order precision. Numerical simulations confirm that FedZeN converges superlinearly, outperforming the main federated zeroth-order algorithms.

References

[1] C. Audet and W. Hare, *Derivative-Free and Blackbox Optimization*. Springer, 01 2017.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr 2017, p. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[3] S. Liu, P.-Y. Chen, B. Kailkhura, G. Zhang, A. O. Hero III, and P. K. Varshney, "A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications," *IEEE Signal Processing Magazine*, vol. 37, no. 5, pp. 43–54, 2020.

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

[5] D. Hajinezhad, M. Hong, and A. Garcia, "Zeroth order nonconvex multi-agent optimization over networks," no. arXiv:1710.09997, Feb 2019, arXiv:1710.09997 [math, stat]. [Online]. Available: http://arxiv.org/abs/1710.09997

[6] H. Feng, T. Pang, C. Du, W. Chen, S. Yan, and M. Lin, "Does federated learning really need backpropagation?" no. arXiv:2301.12195, May 2023, arXiv:2301.12195 [cs]. [Online]. Available: http://arxiv.org/abs/2301.12195

[7] Q. Zhang, B. Gu, Z. Dang, C. Deng, and H. Huang, "Desirable companion for vertical federated learning: New zeroth-order gradient based algorithm," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, ser. CIKM '21. New York, NY, USA: Association for Computing Machinery, Oct 2021, p. 2598–2607. [Online]. Available: https://dl.acm.org/doi/10.1145/3459637.3482249

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

[9] M. Safaryan, R. Islamov, X. Qian, and P. Richtárik, "FedNL: Making newton-type methods applicable to federated learning," no. arXiv:2106.02969, May 2022, arXiv:2106.02969 [cs, math]. [Online]. Available: http://arxiv.org/abs/2106.02969

[10] N. D. Fabbro, S. Dey, M. Rossi, and L. Schenato, "Shed: A newton-type algorithm for federated learning based on incremental hessian eigenvector sharing," no. arXiv:2202.05800, Sep 2022, arXiv:2202.05800 [cs, math]. [Online]. Available: http://arxiv.org/abs/2202.05800

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

[12] A. Agafonov, D. Kamzolov, R. Tappenden, A. Gasnikov, and M. Takáč, "FLECS: A federated learning second-order framework via compression and sketching," no. arXiv:2206.02009, Jun 2022, arXiv:2206.02009 [math]. [Online]. Available: http://arxiv.org/abs/2206.02009

[13] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.

[14] Y. Feng and T. Wang, "Stochastic zeroth-order gradient and hessian estimators: variance reduction and refined bias bounds," *Information and Inference: A Journal of the IMA*, vol. 12, no. 3, p. iaad014, Sep 2023.

[15] K. Balasubramanian and S. Ghadimi, "Zeroth-order nonconvex stochastic optimization: Handling constraints, high dimensionality, and saddle points," *Foundations of Computational Mathematics*, vol. 22, no. 1, p. 35–76, Feb 2022.

[16] Y. Chikuse and Y. Chikuse, *Statistics on special manifolds*. Springer, 2003, vol. 1.

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

[18] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 937–16 947, 2020.

[19] R. Bollapragada, R. H. Byrd, and J. Nocedal, "Exact and inexact subsampled newton methods for optimization," *IMA Journal of Numerical Analysis*, vol. 39, no. 2, p. 545–578, Apr 2019.

[20] J. Blackard, "Covertype," UCI Machine Learning Repository, 1998, DOI: https://doi.org/10.24432/C50K5N.