FedNL: Making Newton-Type Methods Applicable to Federated Learning

Mher Safaryan[1] Rustem Islamov[1, 2] Xun Qian[1] Peter Richtárik[1]

[1]King Abdullah University of Science and Technology (KAUST) [2]Moscow Institute of Physics and Technology (MIPT)

The Problem and Assumptions

We want to solve the finite-sum optimization problem

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

(μ-strongly convex; # machines/devices; has Lipschitz Hessian; #model parameters; empirical loss/risk; local training data; local loss function $f_i(x) = \mathbb{E}_{\xi\sim\mathcal{D}_i}[f_\xi(x)]$)

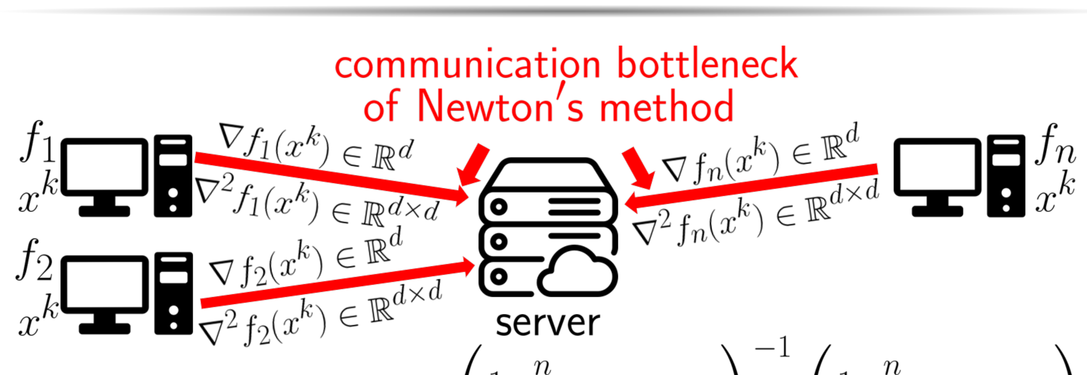
- Problem (1) has many applications in machine learning, data science and engineering.
- We focus on the regime when n **and** d **are very large**. This is typically the case in the big data settings (e.g., massively distributed and federated learning).

Notation: x^* is the solution of Problem (1).

Main goal

Our goal is to develop a **communication efficient** Newton-type method whose local convergence rate will be **independent of the condition number**, which will support **partial participation (PP)**, **bidirectional compression (BC)** and **globalization** techniques: **cubic regularization (CR)** and **line search (LS)**.

Communication bottleneck



$$x^{k+1} = x^k - \left(\frac{1}{n}\sum_{i=1}^{n}\nabla^2 f_i(x^k)\right)^{-1}\left(\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(x^k)\right)$$

Newton's Triangle

Newton: $x^{k+1} = x^k - (\nabla^2 f(x^k))^{-1}\nabla f(x^k)$.

Newton Star: $x^{k+1} = x^k - (\nabla^2 f(x^*))^{-1}\nabla f(x^k)$.

Newton Zero: $x^{k+1} = x^k - (\nabla^2 f(x^0))^{-1}\nabla f(x^k)$.

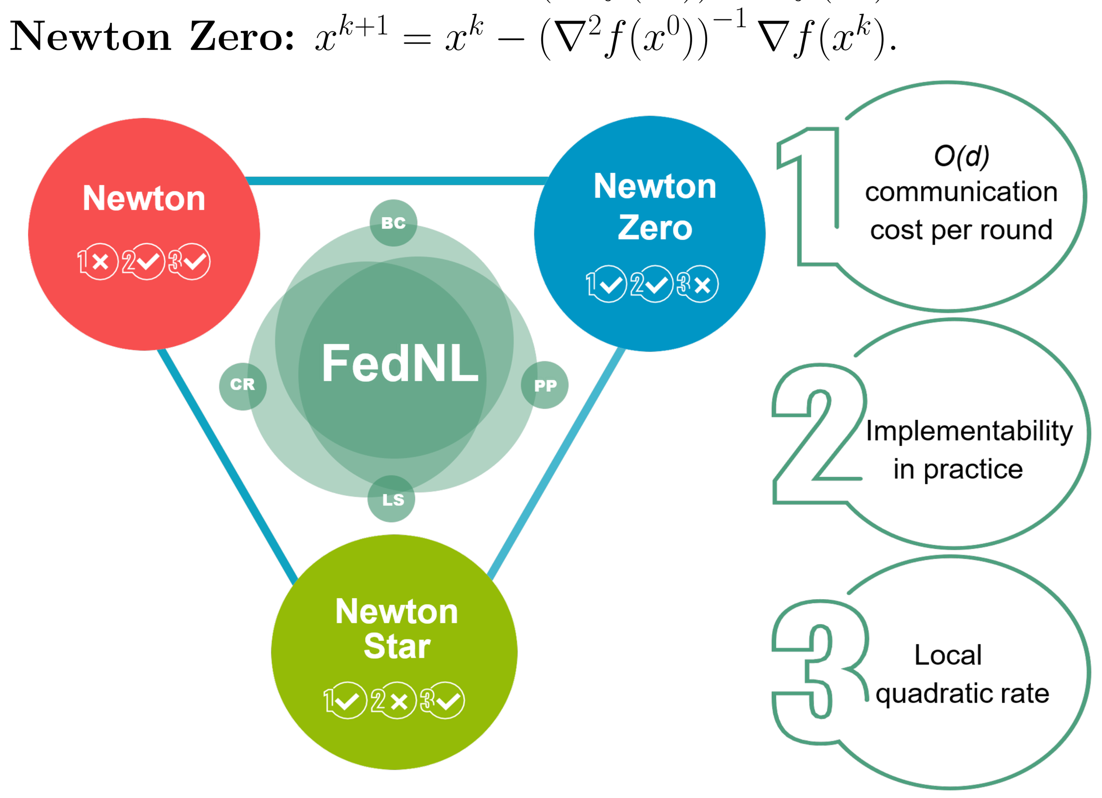


Figure 1: FedNL and its four extensions interpolates between these three special Newton-type methods — Newton (N), Newton Star (NS) and Newton Zero (N0).

FedNL

How to satisfy all goals?

- Learn the Hessian at the optimum, since NS already has all; properties;
- Compressed communication.

In FedNL we maintain a sequence of matrices $\mathbf{H}_i^k \in \mathbb{R}^{d\times d}$, for all $i = 1,\dots,n$ throughout the iterations $k \geq 0$, with the goal of learning $\nabla^2 f_i(x^*)$ for all i:

$$\mathbf{H}_i^k \to \nabla^2 f_i(x^*) \quad \text{as} \quad k\to+\infty.$$

Using $\mathbf{H}_i^k \approx \nabla^2 f_i(x^*)$, we can estimate the Hessian $\nabla^2 f(x^*)$ via

$$\nabla^2 f_i(x^*) \approx \mathbf{H}^k := \frac{1}{n}\sum_{i=1}^{n}\mathbf{H}_i^k.$$

Learning the matrices: the idea

We design a learning rule for matrices $\mathbf{H}_i^k$ via the **DIANA trick** [1] :

$$\mathbf{H}_i^{k+1} = \mathbf{H}_i^k + \alpha\mathcal{C}_i^k\left(\nabla^2 f_i(x^k) - \mathbf{H}_i^k\right),$$

where $\alpha > 0$ is a learning rate, and $\mathcal{C}_i^k$ is a freshly sampled compressor by node i at iteration k.

Compressing matrices

Unbiased Compressors. By $\mathbb{B}(\omega)$ we denote the class of (possibly randomized) unbiased compression operators $\mathcal{C}:\mathbb{R}^{d\times d}\to\mathbb{R}^{d\times d}$ with variance parameter $\omega\geq 0$ satisfying

$$\mathbb{E}\left[\mathcal{C}(\mathbf{M})\right] = \mathbf{M}, \quad \mathbb{E}\left[\|\mathcal{C}(\mathbf{M})-\mathbf{M}\|_{\mathrm{F}}^2\right] \leq \omega\|\mathbf{M}\|_{\mathrm{F}}^2 \quad \forall\, \mathbf{M}\in\mathbb{R}^{d\times d}.$$

Example: For arbitrary matrix $\mathbf{M}$ we choose a set $\mathcal{S}_K$ of indices (i,j) of cardinality K uniformly at random, then Rand-K compressor can be defined via

$$\mathcal{C}(\mathbf{M})_{ij} = \begin{cases} \frac{d^2}{K}\mathbf{M}_{ij} & \text{if } (i,j)\in\mathcal{S}_K, \\ 0 & \text{otherwise.}\end{cases}$$

Rand-K belongs to $\mathbb{B}(\omega)$ with $\omega = \frac{d^2}{K} - 1$.

Contractive Compressors. By $\mathbb{C}(\delta)$ we denote the class of deterministic contractive compression operators $\mathcal{C}:\mathbb{R}^{d\times d}\to\mathbb{R}^{d\times d}$ with contraction parameter $\delta\in[0,1]$ satisfying

$$\|\mathcal{C}(\mathbf{M})\|_{\mathrm{F}} \leq \|\mathbf{M}\|_{\mathrm{F}}, \quad \|\mathcal{C}(\mathbf{M})-\mathbf{M}\|_{\mathrm{F}}^2 \leq (1-\delta)\|\mathbf{M}\|_{\mathrm{F}}^2, \quad \forall\, \mathbf{M}\in\mathbb{R}^{d\times d}.$$

Example: For arbitrary matrix $\mathbf{M}$ we choose a set $\mathcal{G}_K$ of indices (i,j) of cardinality K related to K maximum elements of $\mathbf{M}$ by magnitude, then Top-K compressor can be defined via

$$\mathcal{C}(\mathbf{M})_{ij} = \begin{cases} \mathbf{M}_{ij} & \text{if } (i,j)\in\mathcal{G}_K, \\ 0 & \text{otherwise.}\end{cases}$$

Top-K belongs to $\mathbb{C}(\delta)$ with $\delta = \frac{K}{d^2}$.

Main features of the family of FedNL methods important for Federated Learning

- supports **heterogeneous data** setting
- uses **adaptive stepsizes**
- supports **unbiased Hessian compression** (e.g., Rand-K)
- fast local rate: **independent of the condition number**
- has global convergence guarantees via **line search**
- supports smart **uplink gradient compression** at the devices
- applies to general **finite-sum problems**
- **privacy is enhanced** (training data is not sent to the server)
- supports **contractive Hessian compression** (e.g., Top-K)
- supports **partial participation**
- has global convergence guarantees via **cubic regularization**
- supports smart **downlink model compression** by the server

Table: Convergence results for the family of FedNL methods.

Method	Convergence result †	type	rate	Rate independent of the condition number
N0	$r_k \leq \frac{1}{2^k}r_0$	local	linear	✓
NS	$r_{k+1} \leq c r_k^2$	local	quadratic	✓
FedNL	$r_k \leq \frac{1}{2^k}r_0$	local	linear	✓
	$\Phi_1^k \leq \theta^k\Phi_1^0$	local	linear	✓
	$r_{k+1} \leq c\theta^k r_k$	local	superlinear	✓
FedNL-PP[1]	$\mathcal{W}^k \leq \theta^k\mathcal{W}^0$	local	linear	✓
	$\Phi_2^k \leq \theta^k\Phi_2^0$	local	linear	✓
	$r_{k+1} \leq c\theta^k\mathcal{W}_k$	local	linear	✓
FedNL-LS[2]	$\Delta_k \leq \theta^k\Delta_0$	global	linear	✗
FedNL-CR[3]	$\Delta_k \leq c/k$	global	sublinear	✗
	$\Delta_k \leq \theta^k\Delta_0$	global	linear	✗
	$\Phi_1^k \leq \theta^k\Phi_1^0$	local	linear	✓
	$r_{k+1} \leq c\theta^k r_k$	local	superlinear	✓
FedNL-BC[4]	$\Phi_3^k \leq \theta^k\Phi_3^0$	local	linear	✓

† Refer to the precise statements of the theorems in [4].
[1]FedNL with partial participation; [2]FedNL with line search; [3]FedNL with cubic regularization; [4]FedNL with bidirectional compression.

Algorithm 1: FedNL (Federated Newton Learn)

Parameters: Hessian learning rate $\alpha\geq 0$; compression operators $\{\mathcal{C}_1^k,\dots,\mathcal{C}_n^k\}$

Initialization: $x^0\in\mathbb{R}^d$; $\mathbf{H}_1^0,\dots,\mathbf{H}_n^0\in\mathbb{R}^{d\times d}$ and $\mathbf{H}^0 := \frac{1}{n}\sum_{i=1}^n\mathbf{H}_i^0$

for *each device* $i = 1,\dots,n$ *in parallel* **do**

  Get x^k from the server and compute local gradient $\nabla f_i(x^k)$ and local Hessian $\nabla^2 f_i(x^k)$

  Send $\nabla f_i(x^k)$, $\mathbf{S}_i^k := \mathcal{C}_i^k(\nabla^2 f_i(x^k) - \mathbf{H}_i^k)$ and $l_i^k := \|\mathbf{H}_i^k - \nabla^2 f_i(x^k)\|_{\mathrm{F}}$ to the server

  Update local Hessian shift to $\mathbf{H}_i^{k+1} = \mathbf{H}_i^k + \alpha\mathbf{S}_i^k$

end

on server

  Get $\nabla f_i(x^k)$, $\mathbf{S}_i^k$ and l_i^k from each node $i\in[n]$

  $\nabla f(x^k) = \frac{1}{n}\sum_{i=1}^n\nabla f_i(x^k)$, $\mathbf{S}^k = \frac{1}{n}\sum_{i=1}^n\mathbf{S}_i^k$

  $l^k = \frac{1}{n}\sum_{i=1}^n l_i^k$, $\mathbf{H}^{k+1} = \mathbf{H}^k + \alpha\mathbf{S}^k$

  Option 1: $x^{k+1} = x^k - \left[\mathbf{H}^k\right]_\mu^{-1}\nabla f(x^k)$

  Option 2: $x^{k+1} = x^k - \left[\mathbf{H}^k + l^k\mathbf{I}\right]^{-1}\nabla f(x^k)$

$[\cdot]_\mu$ denotes the projection onto the cone of positive definite matrices with constant μ.

Experiments

We consider L2 regularized logistic regression problem:

$$\min_{x\in\mathbb{R}^d}\left\{\frac{1}{n}\sum_{i=1}^{n} f_i(x) + \frac{\lambda}{2}\|x\|^2\right\}, \quad f_i(x) = \frac{1}{m}\sum_{j=1}^{m}\log\left(1+e^{-b_{ij}a_{ij}^\top x}\right).$$

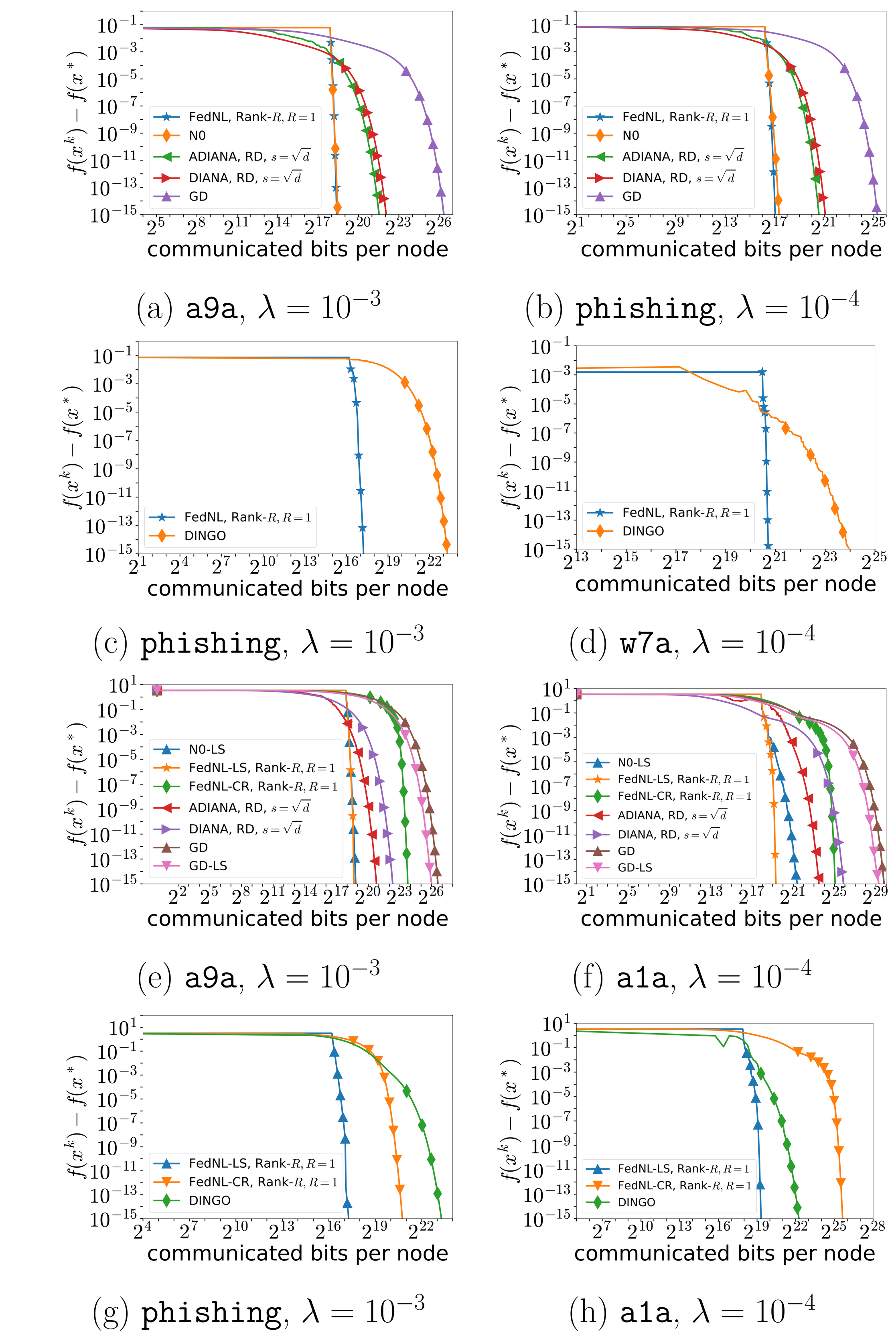


Figure 2: **First row:** Local comparison of FedNL and N0 with ADIANA, DIANA, and GD; **Second row:** Local comparison of FedNL with DINGO (second row); **Third row:** Global comparison of FedNL-LS, N0-LS, and FedNL-CR with ADIANA, DIANA, GD, and GD-LS; **Fourth row:** Global comparison of FedNL-LS and FedNL-CR with DINGO; in terms of communication complexity.

References

[1] Konstantin Mishchenko, Eduard Gorbunov, Martin Takáč, and Peter Richtárik. Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*, 2019.

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

[3] Filip Hanzely, Nikita Doikov, Yurii Nesterov, and Peter Richtárik. Stochastic subspace cubic Newton method. *In International Conference on Machine Learning*, pages 4027–4038. PMLR, 2020.

[4] Mher Safaryan, Rustem Islamov, Xun Qian, and Peter Richtárik. FedNL: Making Newton-Type Methods Applicable to Federated Learning. *arXiv preprint arXiv:2106.02969*, 2021.