# FedNL: Making Newton-Type Methods Applicable to Federated Learning

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

[1]KAUST [2]MIPT

## The Problem and Assumptions

We want to solve the finite-sum optimization problem

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

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

**Assumptions:**

- $f$ is $\mu$-strongly convex;
- all $f_i$ have Lipschitz continuous Hessians with respect to spectral ($L_*$), Frobenius ($L_{\rm F}$) and infinity ($L_\infty$) norms;
- $x^*$ is the solution for Problem (1).

### Main goal

Our goal is to develop a communication efficient Newton-type method for federated learning.

## Newton's method

**Newton's step:** $x^{k+1} = x^k - \left(\nabla^2 f(x^k)\right)^{-1}\nabla f(x^k)$.

Pros:
- Fast **local quadratic** convergence rate
- Rate is **independent on the condition number**

Cons:
- Requires $\mathcal{O}(d^2)$ floats to be communicated by each worker to the server, where $d$ is typically very large

## Newton Star

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

Pros:
- Fast **local quadratic** convergence rate
- Rate is **independent on the condition number**
- Requires $\mathcal{O}(d)$ floats to be communicated by each worker;

Cons:
- Cannot be implemented in practice.

## Newton Zero

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

Pros:
- Fast **local linear** convergence rate
- Rate is **independent on the condition number**
- Requires $\mathcal{O}(d)$ floats to be communicated by each worker;

### Newton Triangle

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

## FedNL

**How to address the communication bottleneck?**
- 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 $k \ge 0$, with the goal of learning $\mathbf{H}_i(x^*)$ 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

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

## FedNL: Extensions

- **FedNL-PP:** FedNL with partial participation;
- **FedNL-LS:** FedNL with globalization via Line Search;
- **FedNL-CR:** FedNL with globalization via Cubic Regularization [2];
- **FedNL-BC:** FedNL with Bidirectional Compression.

## Compression operators

**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 $omega \ge 0$ satisfying

$$\mathbb{E}\mathcal{C}(M) = \mathbf{M}, \quad \mathbb{E}\|\mathcal{C}(\mathbf{M})-\mathbf{M}\|_{\rm F}^2 \le \omega\|\mathbf{M}\|_{\rm F}^2 \quad \mathbf{M}\in\mathbb{R}^{d\times d}.$$

**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

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

## Learning mechanism

### 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 $k$.

## Main features of a 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 a family of FedNL methods.

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

† Refer to the precise statements of the theorems in [3] for the exact values.

**Algorithm 1:** FedNL (Federated Newton Learn)

**Parameters:** Hessian learning rate $\alpha \ge 0$; 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 $l_i^k := \|\mathbf{H}_i^k - \nabla^2 f_i(x^k)\|_{\rm F}$ 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 $i\in[n]$

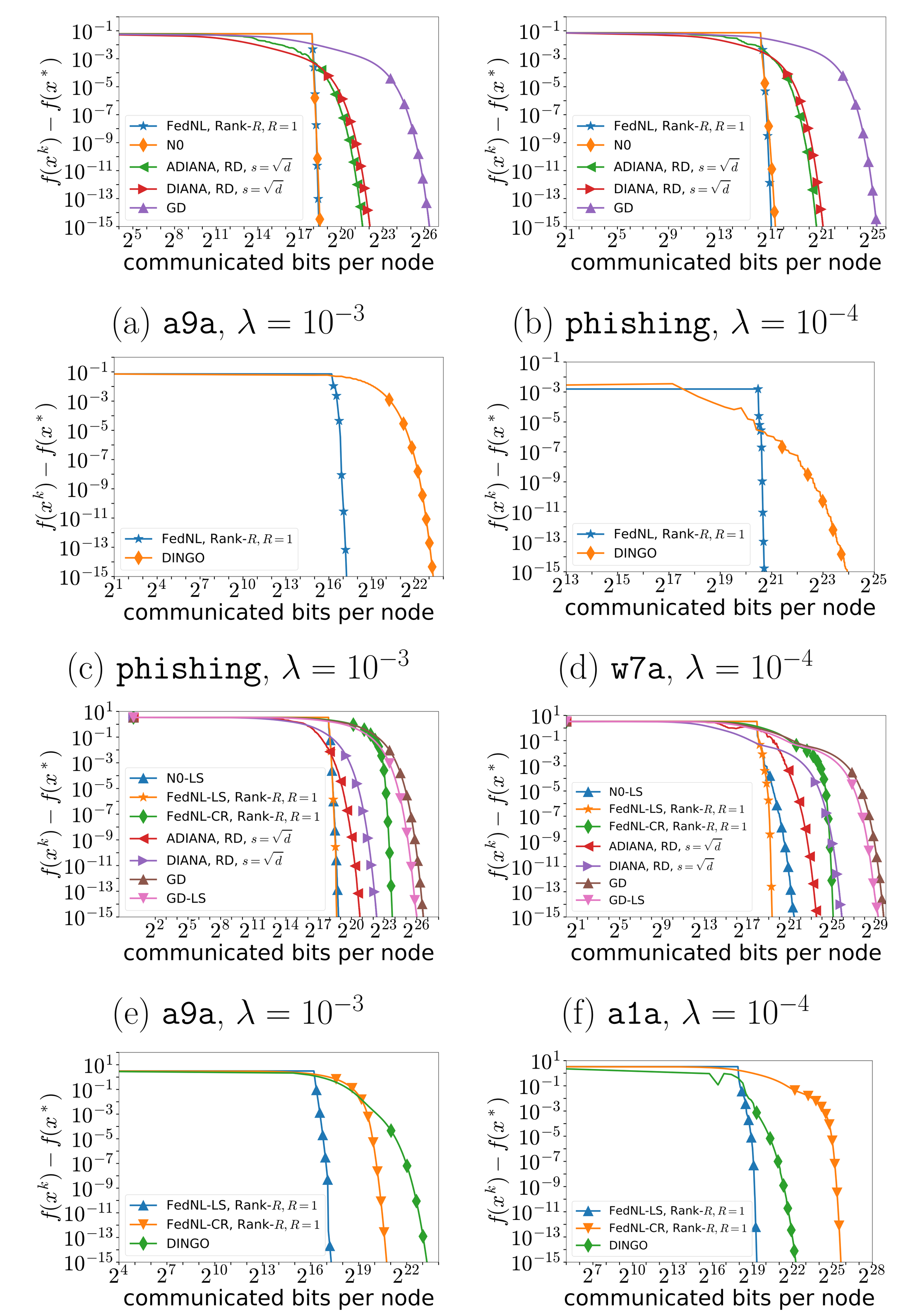
$\nabla f(x^k) = \frac{1}{n}\sum_{i=1}^n \nabla f_i(x^k)$, $\mathbf{S}^k = \frac{1}{n}\sum_{i=1}^n \mathbf{S}_i^k$

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

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

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

## Experiments




(a) a9a, $\lambda = 10^{-3}$ (b) phishing, $\lambda = 10^{-4}$ (c) phishing, $\lambda = 10^{-3}$ (d) w7a, $\lambda = 10^{-4}$ (e) a9a, $\lambda = 10^{-3}$ (f) a1a, $\lambda = 10^{-4}$ (g) phishing, $\lambda = 10^{-3}$ (h) a1a, $\lambda = 10^{-4}$

Figure 1: **First row:** Local comparison of FedNL and N0 with ADIANA, DIANA, and GD in terms of communication complexity. **Second row:** Local comparison of FedNL with DINGO (second row) in terms of communication complexity. **Third row:** Global comparison of FedNL-LS, N0-LS, and FedNL-CR with ADIANA, DIANA, GD, and GD-LS in terms of communication complexity. **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] Yurii Nesterov and Boris T. Polyak. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 108(1) : 177 − 205, 2006.

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