

## The Problem

$$\min_{x \in \mathbb{R}^d} \left[ P(x) := f(x) + \frac{\lambda}{2} \|x\|^2 \right]. \quad (1)$$

Function  $f$  is convex, and has an “average of averages” structure:

$$f(x) := \frac{1}{n} \sum_{i=1}^n f_i(x), \quad f_i(x) := \frac{1}{m} \sum_{j=1}^m f_{ij}(x), \quad (2)$$

and  $\lambda \geq 0$  is a regularization parameter. Each  $f_{ij}$  is a function of the form:  $f_{ij}(x) := \varphi_{ij}(a_{ij}^\top x)$ . The Hessian of  $f_{ij}$  at point  $x$  is

$$\mathbf{H}_{ij}(x) := h_{ij}(x) a_{ij} a_{ij}^\top, \quad h_{ij}(x) := \varphi_{ij}''(a_{ij}^\top x). \quad (3)$$

The Hessian  $\mathbf{H}_i(x)$  of local functions  $f_i(x)$  and the Hessian  $\mathbf{H}(x)$  of  $f$  can be represented as linear combination of one-rank matrices.

## Assumptions

We assume that Problem (1) has at least one optimal solution  $x^*$ . For all  $i$  and  $j$ ,  $\varphi_{ij}''$  is  $\gamma$ -smooth, twice differentiable, and its second derivative  $\varphi_{ij}''$  is  $\nu$ -Lipschitz continuous.

### Main goal

Our goal is to develop a communication efficient Newton-type method for distributed optimization.

## Naive distributed implementation of Newton’s method

**Newton’s step:**  $x^{k+1} \stackrel{(1)}{=} x^k - (\mathbf{H}(x^k) + \lambda \mathbf{I})^{-1} \nabla P(x^k)$ .

**Each node:** computes the local Hessian  $\mathbf{H}_i(x^k)$  and gradient  $\nabla f_i(x^k)$ , then sends them to the server.

**Server:** averages the local Hessians and gradients to produce  $\mathbf{H}(x^k)$  and  $\nabla f(x^k)$ , respectively, adds  $\lambda \mathbf{I}$  to  $\mathbf{H}(x^k)$  and  $\lambda x^k$  to  $\nabla f(x^k)$ , then performs Newton step. Next, it sends  $x^{k+1}$  back to the nodes.

**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 (NS)

Assume that the server has access to coefficients  $h_{ij}(x^*)$  for all  $i$  and  $j$ , i.e access to the Hessian  $\mathbf{H}(x^*)$ .

**Step of NEWTON-STAR:**  $x^{k+1} = x^k - (\mathbf{H}(x^*) + \lambda \mathbf{I})^{-1} \nabla P(x^k)$ .

### Theorem 1 (Convergence of NS)

Assume that  $\mathbf{H}(x^*) \succeq \mu^* \mathbf{I}$  for some  $\mu^* \geq 0$  and that  $\mu^* + \lambda > 0$ . Then for any starting point  $x^0 \in \mathbb{R}^d$ , the iterates of NEWTON-STAR satisfy the following inequality:

$$\|x^{k+1} - x^*\| \leq \frac{\nu}{2(\mu^* + \lambda)} \cdot \left( \frac{1}{nm} \sum_{i=1}^n \sum_{j=1}^m \|a_{ij}\|^3 \right) \cdot \|x^k - x^*\|^2.$$

**Pros:** • Fast local quadratic convergence rate  
• Rate is independent on the condition number  
• Communication cost is  $\mathcal{O}(d)$  per-iteration

**Cons:** • Cannot be implemented in practice

## NEWTON-LEARN

**How to address the communication bottleneck?**

- Compressed communication
- Taking advantage of the structure of the problem

In NEWTON-LEARN we maintain a sequence of vectors

$$h_i^k = (h_{i1}^k, \dots, h_{im}^k) \in \mathbb{R}^m, \quad (4)$$

for all  $i = 1, \dots, n$  throughout the iterations  $k \geq 0$ , with the goal of learning the values  $h_{ij}(x^*)$  for all  $i, j$ :

$$h_{ij}(x^k) \rightarrow h_{ij}(x^*) \quad \text{as } k \rightarrow +\infty. \quad (5)$$

Using  $h_{ij}^k \approx h_{ij}(x^*)$ , we can estimate the Hessian  $\mathbf{H}(x^*)$  via

$$\mathbf{H}(x^*) \approx \mathbf{H}^k := \frac{1}{n} \sum_{i=1}^n \mathbf{H}_i^k, \quad \mathbf{H}_i^k := \frac{1}{m} \sum_{j=1}^m h_{ij}^k a_{ij} a_{ij}^\top. \quad (6)$$

## Compressed learning

**Compression operator:** A randomized map  $\mathcal{C} : \mathbb{R}^m \rightarrow \mathbb{R}^m$  is a *compression operator* (*compressor*) if there exists a constant  $\omega \geq 0$  such that for all  $x \in \mathbb{R}^m$

$$\mathbb{E}[\mathcal{C}(x)] = x, \quad \mathbb{E}[\|\mathcal{C}(x)\|^2] \leq (\omega + 1) \|x\|^2. \quad (7)$$

**Random sparsification (random- $r$ )** [1]: Compressor defined as

$$\mathcal{C}(x) := \frac{m}{r} \cdot \xi \circ x, \quad (8)$$

where  $\xi \in \mathbb{R}^m$  is a random vector distributed uniformly at random on the discrete set  $\{y \in \{0, 1\}^m : \|y\|_0 = r\}$ . The variance parameter associated with this compressor is  $\omega = \frac{m}{r} - 1$ .

## NEWTON-LEARN: NL1

**Assumption:** We assume that each  $\varphi_{ij}(x)$  is convex, and  $\lambda > 0$ .

### Learning the coefficients: the idea

We design a learning rule for vectors  $h_i^k$  via the **DIANA trick** [2]:

$$h_i^{k+1} = \left[ h_i^k + \eta \mathcal{C}_i^k (h_i(x^k) - h_i^k) \right]_+, \quad (9)$$

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

**Main properties:** •  $h_{ij}^k \geq 0$  for all  $i, j$   
• update is sparse:  $\|h_i^{k+1} - h_i^k\|_0 \leq s$ , where  $s = \mathcal{O}(1)$   
•  $\mathbf{H}^k \succeq 0$

**Each node:** Computes update  $h_i^{k+1} = \left[ h_i^k + \eta \mathcal{C}_i^k (h_i(x^k) - h_i^k) \right]_+$  and gradient  $\nabla f_i(x^k)$ . Then the node broadcasts the gradient, update  $h_i^{k+1} - h_i^k$  and data points  $a_{ij}$  for which  $h_{ij}^{k+1} - h_{ij}^k \neq 0$ .

**Server:** averages the local gradients to produce  $\nabla f(x^k)$  and constructs  $\mathbf{H}^k$  via (6). Then it performs a Newton-like step:

$$x^{k+1} = x^k - (\mathbf{H}^k + \lambda \mathbf{I})^{-1} (\nabla f(x^k) + \lambda x^k), \quad (10)$$

and finally broadcasts  $x^{k+1}$  back to the nodes.

**Pros** • Local linear and superlinear rates  
• Rates are **independent on the condition number**  
• Communication cost  $\mathcal{O}(d)$  per iteration

### Algorithm 1 NL1: NEWTON-LEARN ( $\lambda > 0$ case)

**Parameters:** learning rate  $\eta > 0$

**Initialization:**  $x^0 \in \mathbb{R}^d$ ,  $h_1^0, \dots, h_n^0 \in \mathbb{R}_+^m$ ;

$\mathbf{H}^0 = \frac{1}{nm} \sum_{i=1}^n \sum_{j=1}^m h_{ij}^0 a_{ij} a_{ij}^\top \in \mathbb{R}^{d \times d}$

**for**  $k = 0, 1, \dots$  **do**

Broadcast  $x^k$  to all workers

**for each node**  $i = 1, \dots, n$  **do**

Compute local gradient  $\nabla f_i(x^k)$

$h_i^{k+1} = [h_i^k + \eta \mathcal{C}_i^k (h_i(x^k) - h_i^k)]_+$

Send  $\nabla f_i(x^k)$ ,  $h_i^{k+1} - h_i^k$  and corresponding  $a_{ij}$  to server

**end**

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

$$\mathbf{H}^{k+1} = \mathbf{H}^k + \frac{1}{nm} \sum_{i=1}^n \sum_{j=1}^m (h_{ij}^{k+1} - h_{ij}^k) a_{ij} a_{ij}^\top$$

**end**

## Convergence theory

The analysis relies on the Lyapunov function

$$\Phi_1^k = \|x^k - x^*\|^2 + \frac{1}{\eta n m \nu^2 R^2} \mathcal{H}^k, \quad \mathcal{H}^k = \sum_{i=1}^n \|h_i^k - h_i(x^*)\|^2,$$

where  $R = \max_{i,j} \|a_{ij}\|$ .

### Theorem 2 (convergence of NL1)

**Theorem 2.** Let each  $\varphi_{ij}$  is convex,  $\lambda > 0$ , and  $\eta \leq \frac{1}{\omega+1}$ . Assume that  $\|x^k - x^*\|^2 \leq \frac{\lambda^2}{12\nu^2 R^6}$  for all  $k \geq 0$ . Then for Algorithm 1 we have the inequalities

$$\begin{aligned} \mathbb{E}[\Phi_1^k] &\leq \theta_1^k \Phi_1^0, \\ \mathbb{E} \left[ \frac{\|x^{k+1} - x^*\|^2}{\|x^k - x^*\|^2} \right] &\leq \theta_1^k \left( 6\eta + \frac{1}{2} \right) \frac{\nu^2 R^6}{\lambda^2} \Phi_1^0, \end{aligned}$$

where  $\theta_1 = 1 - \min \left\{ \frac{\eta}{2}, \frac{5}{8} \right\}$ , **which is independent on the condition number.**

Assumption on  $\|x^k - x^*\|$  can be relaxed using the following lemma:

### Lemma 1

Assume  $h_{ij}^k$  is a convex combination of  $\{h_{ij}(x^0), \dots, h_{ij}(x^k)\}$  for all  $i, j$  and  $k$ . Assume  $\|x^0 - x^*\|^2 \leq \frac{\lambda}{12\nu^2 R^6}$ . Then

$$\|x^k - x^*\|^2 \leq \frac{\lambda^2}{12\nu^2 R^6} \text{ for all } k > 0.$$

It is easy to verify that if we choose  $h_{ij}^0 = h_{ij}(x^0)$ , use the random sparsification compressor (8) and  $\eta \leq \frac{1}{\omega+1}$ , then  $h_{ij}^k$  is always a convex combination of  $\{h_{ij}(x^0), \dots, h_{ij}(x^k)\}$  for  $k > 0$ .

## NEWTON-LEARN: NL2

We additionally develop a modified method (NL2) which handles the case where  $P$  is  $\mu$ -strongly convex,  $|h_{ij}^k| \leq \gamma$ , and  $\lambda \geq 0$ .

**Properties:** • Local linear and superlinear rates  
• Rates are **independent on condition number**  
•  $\mathcal{O}(d)$  bits are communicated per iteration

## CUBIC-NEWTON-LEARN

We also constructed a method (CNL) with global convergence guarantees using cubic regularization [3].

**Properties:** • Local linear and superlinear rates

- Global linear rate in the strongly convex case and global sublinear rate in the convex case
- Rates are **independent on condition number**
- $\mathcal{O}(d)$  bits are communicated per iteration

## Experiments

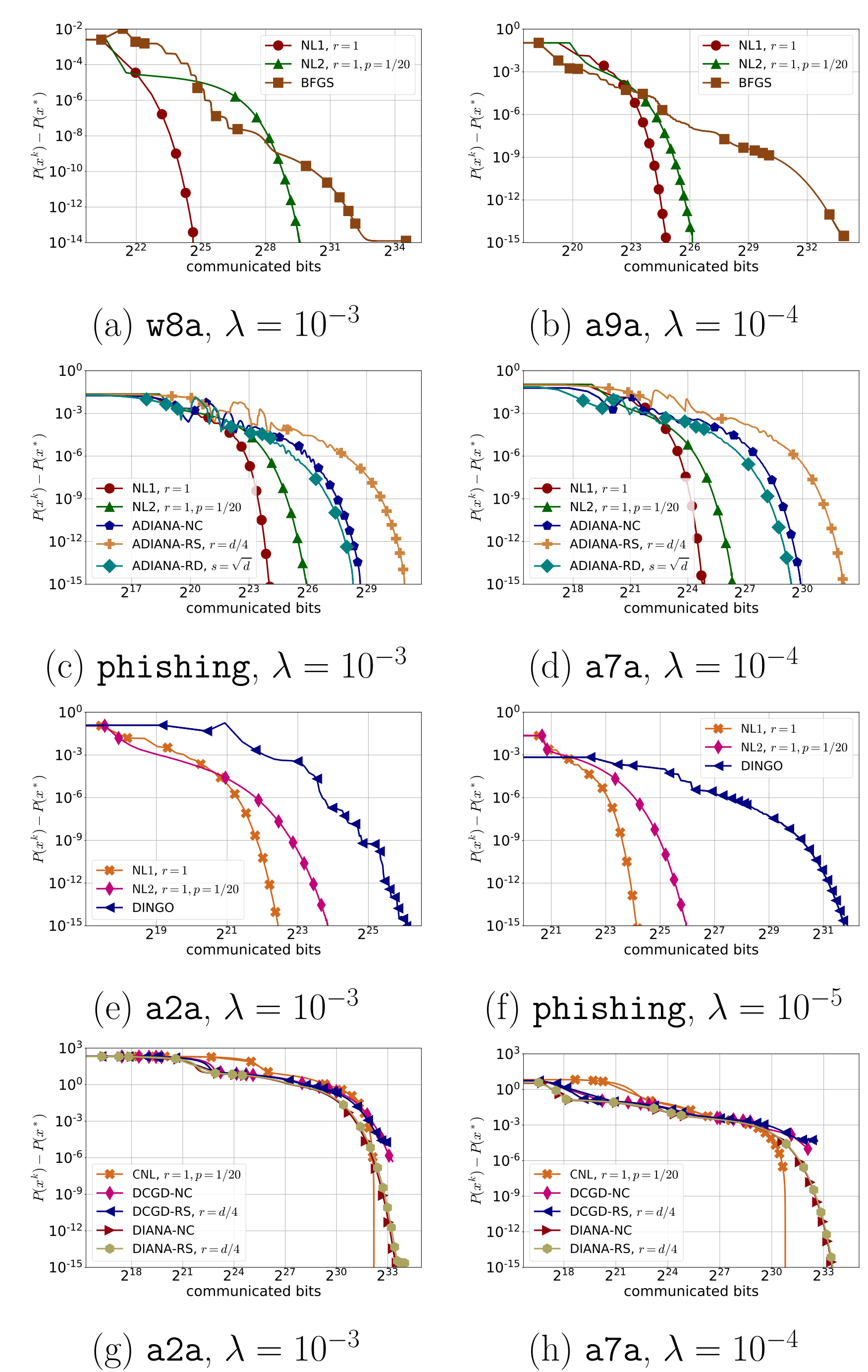


Figure 1: Comparison of NL1, NL2 with (a), (b) BFGS; (c), (d) ADIANA; (e), (f) DINGO in terms of communication complexity. Comparison of CNL with (g), (h) DIANA and DCGD in terms of communication complexity.

## References

- [1] Sebastian U. Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems*, pages 4447 – 4458, 2018.
- [2] Konstantin Mishchenko, Eduard Gorbunov, Martin Takáč, and Peter Richtárik. Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*, 2019.
- [3] Yurii Nesterov and Boris T. Polyak. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 108(1) : 177 – 205, 2006.
- [4] Rustem Islamov, Xun Qian, and Peter Richtárik. Distributed Second Order Methods with Fast Rates and Compressed Communication. *arXiv preprint arXiv:2102.07158*, 2021.