FedNS: A Fast Sketching Newton-Type Algorithm for Federated Learning

Jian Li[1], Yong Liu[2,*] and Weiping Wang[1]

[1] Institute of Information Engineering, Chinese Academy of Sciences

[2] Gaoling School of Artificial Intelligence, Renmin University of China

中国科学院信息工程研究所
INSTITUTE OF INFORMATION ENGINEERING,CAS

Backgrounds

Federated algorithms	Communication Rounds	Communication complexity
First-order methods (e.g. FedAvg & FedProx)	$\mathcal{O}(1/t)$	$\mathcal{O}(M)$
Distributed Newton's Methods	$\mathcal{O}(\log 1/t)$	$\mathcal{O}(M^2)$

Note: The computational complexities are computed in terms of regularized least squared loss to obtain a δ-accurate solution, i.e., $L(\boldsymbol{w}_t) - L(\boldsymbol{w}_{\mathcal{D},\lambda}) \leq \delta$. M is the number of model parameters.

- Existing federated learning algorithms:
  - First-order methods: Low communication burdens but slow convergence.
  - Second-order methods: Fast convergence but high communication burdens.
- Motivation: How to take advantage of both first-order and second-order algorithms, such that the federated algorithms can achieve low communication burdens and fast convergence.
- Contributions: We propose a federated Newton sketch method, named `FedNS`, which shares both first-order and second-order information across devices, i.e., local gradients and sketched square-root Hessian.
  - ☑ **1) On the algorithmic front.** FedNS: a simple federated method with Newton sketch. FedNDES: a communication-efficient Newton-type federated algorithm with line-search.
  - ☑ **2) On the statistical front.** Convergence analysis for two federated Newton sketching algorithms: `FedNS` and its line-search variant `FedNDES`. These methodologies delineate not only **super-linear convergence** rates but also entail a **small communication complexity**.

Preliminaries

$$\boldsymbol{w}_{\mathcal{D},\lambda} = \arg\min_{\boldsymbol{w}\in\mathcal{H}_K} \underbrace{\frac{1}{N}\sum_{i=1}^{N} \ell\left(f(\boldsymbol{x}_i), y_i\right)}_{L(\mathcal{D},\boldsymbol{w})} + \lambda\alpha(\boldsymbol{w}),$$

- Centralized Newton's Method

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \mu\mathcal{H}_{\mathcal{D},t}^{-1}\boldsymbol{g}_{\mathcal{D},t}, \quad \text{with}$$
$$\boldsymbol{g}_{\mathcal{D},t} := \nabla L(\mathcal{D}, \boldsymbol{w}_t) + \lambda\nabla\alpha(\boldsymbol{w}_t),$$
$$\mathcal{H}_{\mathcal{D},t} := \nabla^2 L(\mathcal{D}, \boldsymbol{w}_t) + \lambda\nabla^2\alpha(\boldsymbol{w}_t).$$

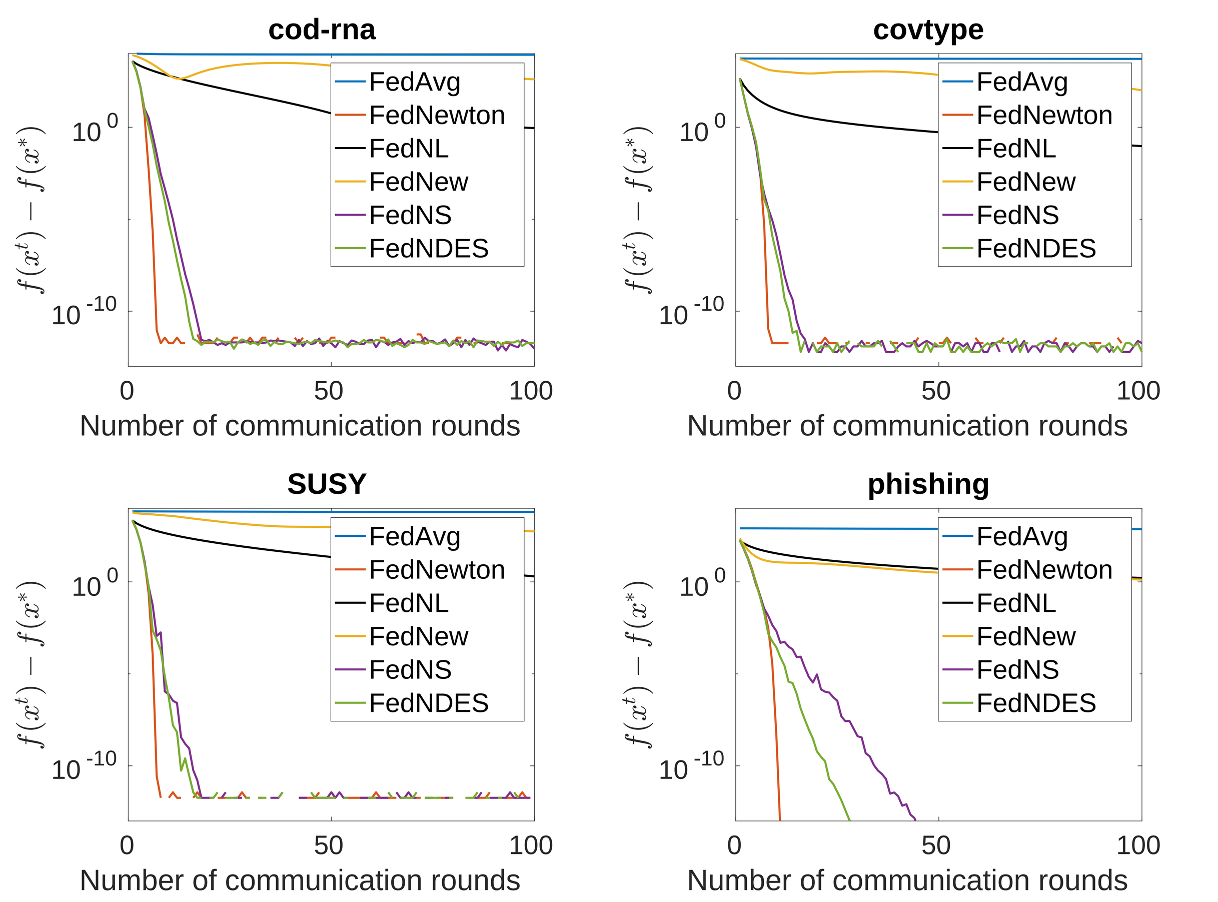
- Newton's Method with Partial Sketching

$$\nabla^2 L(\mathcal{D}, \boldsymbol{w}_t) = (\boldsymbol{S}\nabla^2 L(\mathcal{D}, \boldsymbol{w}_t)^{1/2})^\top(\boldsymbol{S}\nabla^2 L(\mathcal{D}, \boldsymbol{w}_t)^{1/2}).$$

- Federated Newton's Method (FedNewton)

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \mu\mathcal{H}_{\mathcal{D},t}^{-1}\boldsymbol{g}_{\mathcal{D},t} \qquad \text{with}$$
$$\mathcal{H}_{\mathcal{D},t} = \sum_{j=1}^{m}\frac{n_j}{N}\mathcal{H}_{\mathcal{D}_j,t}, \quad \boldsymbol{g}_{\mathcal{D},t} = \sum_{j=1}^{m}\frac{n_j}{N}\boldsymbol{g}_{\mathcal{D}_j,t}.$$

Experiments



- There is significant gaps between the convergence speeds of our proposed methods `FedNS`, `FedNDES` and the existing Newton-type FL methods.
- The proposed `FedNS` and `FedNDES` converge nearly to FedNewton, while FedNew and FedNL converge slowly closed to FedAvg.
- `FedNDES` converges faster than `FedNS` and the final predictive accuracies of `FedNDES` are higher.

FedNS

Algorithm 1: Federated Learning with Newton Sketch (`FedNS`)

Input: Feature mapping $\phi : \mathbb{R}^d \to \mathbb{R}^M$, start point $\boldsymbol{w}_0$, the termination iterations T and the step-size μ.
Output: The global estimator $\boldsymbol{w}_T$.

1: **Local machines:** Compute the local feature mapping data matrix $\phi(\boldsymbol{X}_j)$.
2: **for** $t = 1$ to T **do**
3: **Local machines:** Sample the sketch matrix $\boldsymbol{S}_j^t \in \mathbb{R}^{k\times n_j}$ from the SRHT. Compute local sketch Hessian matrices $\boldsymbol{\Upsilon}_{\mathcal{D}_j,\lambda} = \boldsymbol{S}_j^t\nabla^2 L(\mathcal{D}_j, \boldsymbol{w}_t)^{1/2}$ and local gradients $\boldsymbol{g}_{\mathcal{D}_j,t}$. Upload them to the global server (↑).
4: **Global server:** Compute the global Hessian matrix $\widetilde{\boldsymbol{H}}_{\mathcal{D},t} = \sum_{j=1}^m \frac{n_j}{N}\boldsymbol{\Upsilon}_{\mathcal{D}_j,\lambda}^\top\boldsymbol{\Upsilon}_{\mathcal{D}_j,\lambda} + \lambda\nabla^2\alpha(\boldsymbol{w}_t)$ and the global gradient $\boldsymbol{g}_{\mathcal{D},t} = \sum_{j=1}^m \frac{n_j}{N}\boldsymbol{g}_{\mathcal{D}_j,t}$ and update the global estimator
$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} - \mu\widetilde{\boldsymbol{H}}_{\mathcal{D},t}^{-1}\boldsymbol{g}_{\mathcal{D},t}$$
and communicate it to local machines (↓).
5: **end for**

- Local sketch Hessian

$$\boldsymbol{\Upsilon}_{\mathcal{D}_j,\lambda} = \boldsymbol{S}_j\nabla^2 L(\mathcal{D}_j, \boldsymbol{w}_t)^{1/2}.$$

- Global sketch Hessian

$$\widetilde{\mathcal{H}}_{\mathcal{D},t} = \sum_{j=1}^{m}\frac{n_j}{N}\boldsymbol{\Upsilon}_{\mathcal{D}_j,\lambda}^\top\boldsymbol{\Upsilon}_{\mathcal{D}_j,\lambda} + \lambda\nabla^2\alpha(\boldsymbol{w}_t).$$

FedNDES

Algorithm 2: Dimension-efficient federated Newton (`FedNDES`)

Input: Feature mapping $\phi : \mathbb{R}^d \to \mathbb{R}^M$, start point $\boldsymbol{w}_0$, accuracy tolerance $\delta > 0$, line-search parameters (a, b), threshold sketch sizes $\overline{m}_1$ and $\overline{m}_2$, and the decrement parameter η.
Output: The global estimator $\boldsymbol{w}_T$.

1: **Local machines:** Compute the local feature mapping data matrix $\phi(\boldsymbol{X}_j)$. Initialize and $k_t = \overline{m}_1$.
2: **for** $t = 1, \cdots, T$ **do**
3: **Local machines:** Sample the sketch matrix $\boldsymbol{S}_j^t \in \mathbb{R}^{k\times n_j}$ from the SRHT. Compute local sketch square root Hessian $\boldsymbol{\Upsilon}_{\mathcal{D}_j,\lambda} = \boldsymbol{S}_j^t\nabla^2 L(\mathcal{D}_j, \boldsymbol{w}_t)^{1/2}$ and local gradients $\boldsymbol{g}_{\mathcal{D}_j,t}$. Upload them to the global server (↑).
4: **Global server:** Compute the global Hessian matrix $\widetilde{\boldsymbol{H}}_{\mathcal{D},t} = \sum_{j=1}^m \frac{n_j}{N}\boldsymbol{\Upsilon}_{\mathcal{D}_j,\lambda}^\top\boldsymbol{\Upsilon}_{\mathcal{D}_j,\lambda} + \lambda\nabla^2\alpha(\boldsymbol{w}_t)$, the global gradient $\boldsymbol{g}_{\mathcal{D},t} = \sum_{j=1}^m \frac{n_j}{N}\boldsymbol{g}_{\mathcal{D}_j,t}$ and the approximate Newton step $\Delta\boldsymbol{w}_t = -\widetilde{\boldsymbol{H}}_{\mathcal{D},t}^{-1}\boldsymbol{g}_{\mathcal{D},t}$. Compute the approximate Newton decrement
$$\tilde{\lambda}(\boldsymbol{w}_t) = \boldsymbol{g}_{\mathcal{D},t}^\top\Delta\boldsymbol{w}_t.$$
If $\tilde{\lambda}(\boldsymbol{w}_t)^2 \leq \frac{3}{4}\delta$ return the model $\boldsymbol{w}_t$. Otherwise send $\Delta\boldsymbol{w}_t$ and $\tilde{\lambda}(\boldsymbol{w}_t)$ to local workers.
5: **Local machines:** Line search from $\mu_j = 1$: **while** $L(\mathcal{D}_j, \boldsymbol{w}_t + \mu_j\Delta\boldsymbol{w}_t) > L(\mathcal{D}_j, \boldsymbol{w}_t) + a\mu_j\tilde{\lambda}(\boldsymbol{w}_t)$, **then** $\mu_j \leftarrow b\mu_j$. Send μ_j to the global server.
6: **Global server:** Let $\mu = \min_{j\in m}\mu_j$. Update the global estimator
$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \mu\Delta\boldsymbol{w}_t.$$
If $\tilde{\lambda}(\boldsymbol{w}_t) > \eta$, set $k = \overline{m}_1$. Otherwise, set $k = \overline{m}_2$. Communicate the global model $\boldsymbol{w}_t$ and the sketch size k to local machines (↓).
7: **end for**

Comparison

Related Work	Heterogeneous setting	Sketch size k	Iterations t	Communication per round	Total communication complexity
FedAvg (Li et al. 2020c; Su, Xu, and Yang 2021)	√	–	$\mathcal{O}\left(\frac{1}{\delta}\right)$	$\mathcal{O}(M)$	$\mathcal{O}(\frac{M}{\delta})$
FedProx (Li et al. 2020a; Su, Xu, and Yang 2021)	√	–	$\mathcal{O}\left(\frac{1}{\delta}\right)$	$\mathcal{O}(M)$	$\mathcal{O}(\frac{M}{\delta})$
DistributedNewton (Ghosh, Maity, and Mazumdar 2020)	×	–	$\mathcal{O}\left(\log\frac{1}{\delta}\right)$	$\mathcal{O}(M)$	$\mathcal{O}(M\log\frac{1}{\delta})$
LocalNewton (Gupta et al. 2021)	×	–	$\mathcal{O}\left(\log\frac{1}{\delta}\right)$	$\mathcal{O}(M)$	$\mathcal{O}(M\log\frac{1}{\delta})$
FedNL (Safaryan et al. 2022)	√	–	$\mathcal{O}\left(\log\frac{1}{\delta}\right)$	$\mathcal{O}(M)$	$\mathcal{O}(M\log\frac{1}{\delta})$
SHED (Fabbro et al. 2022)	√	–	$\mathcal{O}(\log\frac{1}{\delta})$	–	$\mathcal{O}(M^2)$
FedNewton	√	–	$\mathcal{O}\left(\log\log\frac{1}{\delta}\right)$	$\mathcal{O}(M^2)$	$\mathcal{O}(M^2\log\log\frac{1}{\delta})$
`FedNS` (Algorithm 1)	√	M	$\mathcal{O}\left(\log\log\frac{1}{\delta}\right)$	$\mathcal{O}(kM)$	$\mathcal{O}(kM\log\log\frac{1}{\delta})$
`FedNDES` (Algorithm 2)	√	d_λ	$\mathcal{O}\left(\log\log\frac{1}{\delta}\right)$	$\mathcal{O}(kM)$	$\mathcal{O}(kM\log\log\frac{1}{\delta})$

Note: The computational complexities are computed in terms of regularized least squared loss to obtain a δ-accurate solution, i.e., $L(\boldsymbol{w}_t) - L(\boldsymbol{w}_{\mathcal{D},\lambda}) \leq \delta$. The convergence analysis for FedAvg is provided in (Li et al. 2020c; Su, Xu, and Yang 2021) and that for FedProx is provided in (Li et al. 2020a; Su, Xu, and Yang 2021).

- The proposed algorithms achieve faster convergence with relatively small communication burdens.