Coping with Heterogeneity and Privacy in Communication-Efficient Federated Optimization

Yuejie Chi

Carnegie Mellon University

Lehigh University
October 2022

Acknowledgements

			
Zhize Li CMU	Boyue Li CMU	Haoyu Zhao Princeton	Peter Richtarik KAUST

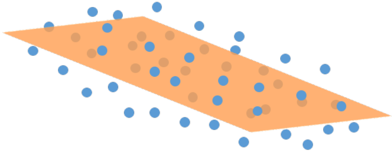
Empirical Risk Minimization (ERM)

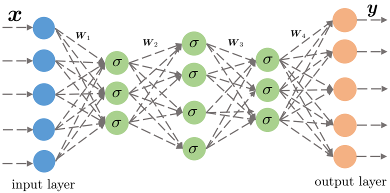
Given a set of data $\mathcal{M}$,

$$\text{minimize}_{\boldsymbol{x}} \quad f(\boldsymbol{x}) = \frac{1}{N} \sum_{\boldsymbol{z} \in \mathcal{M}} \ell(\boldsymbol{x}; \boldsymbol{z})$$

Here, $N =$ number of total samples.

- **convex:** least squares, logistic regression
- **non-convex:** PCA, training neural networks (focus of this talk)



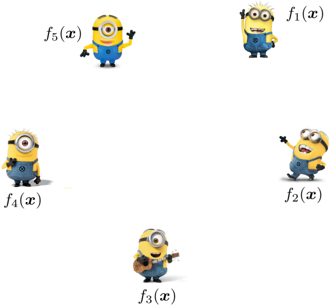


Distributed ERM

Distributed/Federated learning: due to privacy and scalability, data are distributed at multiple locations / workers / agents.

Let $\mathcal{M} = \cup_i \mathcal{M}_i$ be a data partition with equal splitting:

$$f(\boldsymbol{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x}), \quad \text{where} \quad f_i(\boldsymbol{x}) := \frac{1}{(N/n)} \sum_{\boldsymbol{z}\in\mathcal{M}_i} \ell(\boldsymbol{x};\boldsymbol{z}).$$

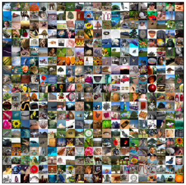


$n =$ number of agents

$\underbrace{N/n}_{m} =$ number of local samples

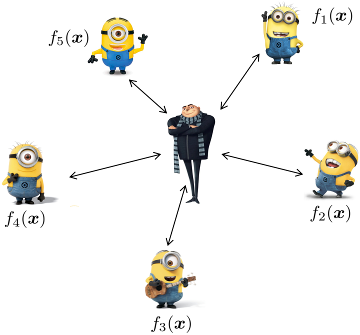
Challenges in federated/decentralized learning

- **Communication efficiency:** limited bandwidth, stragglers, ...
- **Heterogeneity:** non-iid data and systems across the agents
- **Privacy:** does not come for free without sharing data


Two distributed schemes

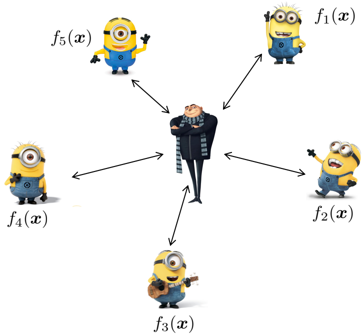
Two distributed schemes



Server/client model

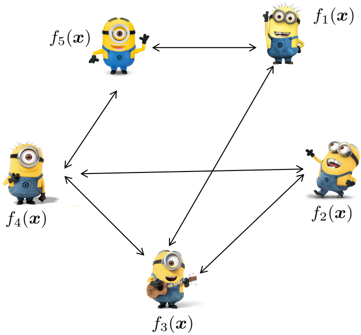
PS coordinates *global* information sharing

Two distributed schemes



Server/client model

PS coordinates *global* information sharing



Network/decentralized model

agents share *local* information over a graph topology

Communication efficiency

Communication cost $=$ Communication rounds $\times$ Cost per round

Communication efficiency

Communication cost = Communication rounds $\times$ Cost per round

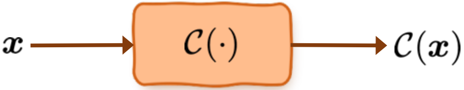
- **Local method:** perform more local computation to reduce communication rounds, e.g. FedAvg (McMahan et al., 2016).

Communication efficiency

Communication cost $=$ Communication rounds $\times$ Cost per round

- **Local method:** perform more local computation to reduce communication rounds, e.g. FedAvg (McMahan et al., 2016).

- **Communication compression:** compress the message into fewer bits, e.g. sparsification or quantization (Alistarh et al., 2017).

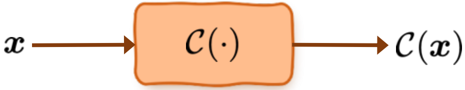


Communication efficiency

Communication cost $=$ Communication rounds $\times$ Cost per round

- **Local method:** perform more local computation to reduce communication rounds, e.g. FedAvg (McMahan et al., 2016).

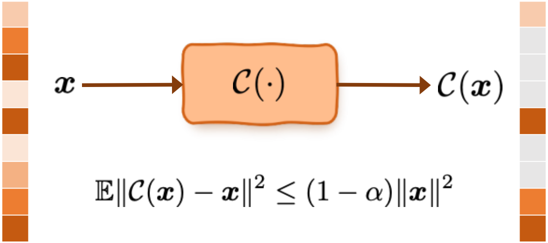
- **Communication compression:** compress the message into fewer bits, e.g. sparsification or quantization (Alistarh et al., 2017).



We will focus on the latter, which are particularly suitable for bandwidth-limited environments.

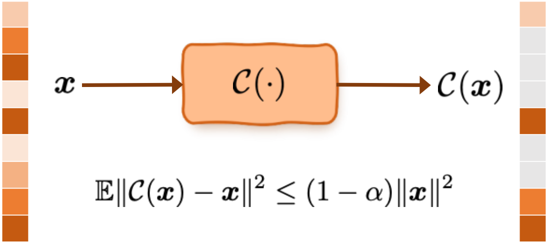
Communication compression

Communication compression is a popular approach to reduce communication cost (e.g., (Alistarh et al., 2017); (Koloskova et al., 2019)).



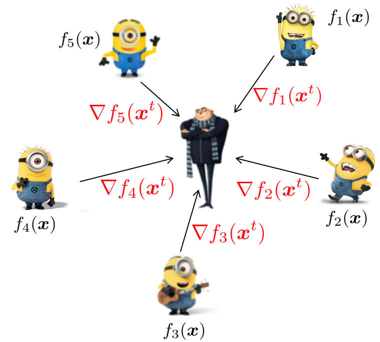
Communication compression

Communication compression is a popular approach to reduce communication cost (e.g., (Alistarh et al., 2017); (Koloskova et al., 2019)).

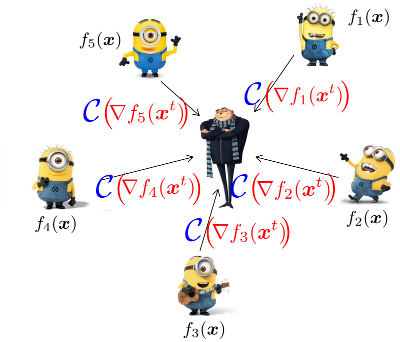


- **random sparsification:** $\alpha = k/d$ measures the compression ratio.
- Other examples: random quantization, top quantization, etc....

A prelude: what should we compress?



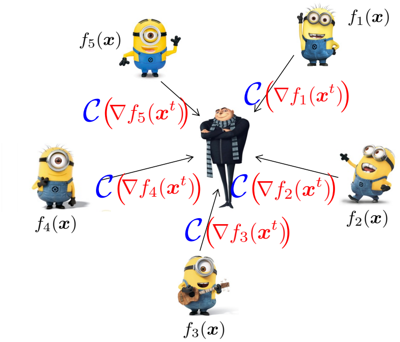
A prelude: what should we compress?



What about

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - \eta \frac{1}{n} \sum_{i=1}^{n} \mathcal{C}(\nabla f_i(\boldsymbol{x}^t))?$$

A prelude: what should we compress?



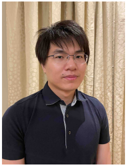
What about

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - \eta \frac{1}{n} \sum_{i=1}^{n} \mathcal{C}(\nabla f_i(\boldsymbol{x}^t))?$$

Somewhat surprisingly, *direct compression* may not work!

A counter-example

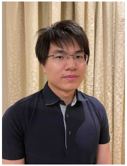
Consider $n = 3$ and let $f_i(x) = (\boldsymbol{a}_i^\top \boldsymbol{x})^2 + \frac{1}{2}\|\boldsymbol{x}\|^2$, where $\boldsymbol{a}_1 = (-4, 3, 3)^\top$, $\boldsymbol{a}_2 = (3, -4, 3)^\top$ and $\boldsymbol{a}_3 = (3, 3, -4)^\top$.



Zhize Li

A counter-example

Consider $n = 3$ and let $f_i(x) = (\boldsymbol{a}_i^\top \boldsymbol{x})^2 + \frac{1}{2}\|\boldsymbol{x}\|^2$, where $\boldsymbol{a}_1 = (-4, 3, 3)^\top$, $\boldsymbol{a}_2 = (3, -4, 3)^\top$ and $\boldsymbol{a}_3 = (3, 3, -4)^\top$.



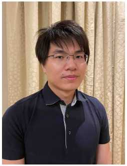
Zhize Li

- Let $\boldsymbol{x}^0 = (b, b, b)$, and the compressor be top$_1$,

$$\begin{aligned}
\nabla f_1(\boldsymbol{x}^0) &= b(-15, 13, 13)^\top & \longrightarrow \quad & \mathcal{C}(\nabla f_1(\boldsymbol{x}^0)) = b(-15, 0, 0)^\top \\
\nabla f_2(\boldsymbol{x}^0) &= b(13, -15, 13)^\top & \longrightarrow \quad & \mathcal{C}(\nabla f_2(\boldsymbol{x}^0)) = b(0, -15, 0)^\top \\
\nabla f_3(\boldsymbol{x}^0) &= b(13, 13, -15)^\top & \longrightarrow \quad & \mathcal{C}(\nabla f_3(\boldsymbol{x}^0)) = b(0, 0, -15)^\top
\end{aligned}$$

A counter-example

Consider $n = 3$ and let $f_i(x) = (\boldsymbol{a}_i^\top \boldsymbol{x})^2 + \frac{1}{2}\|\boldsymbol{x}\|^2$, where $\boldsymbol{a}_1 = (-4, 3, 3)^\top$, $\boldsymbol{a}_2 = (3, -4, 3)^\top$ and $\boldsymbol{a}_3 = (3, 3, -4)^\top$.



Zhize Li

- Let $\boldsymbol{x}^0 = (b, b, b)$, and the compressor be top$_1$,

$$\nabla f_1(\boldsymbol{x}^0) = b(-15, 13, 13)^\top \quad \longrightarrow \quad \mathcal{C}(\nabla f_1(\boldsymbol{x}^0)) = b(-15, 0, 0)^\top$$
$$\nabla f_2(\boldsymbol{x}^0) = b(13, -15, 13)^\top \quad \longrightarrow \quad \mathcal{C}(\nabla f_2(\boldsymbol{x}^0)) = b(0, -15, 0)^\top$$
$$\nabla f_3(\boldsymbol{x}^0) = b(13, 13, -15)^\top \quad \longrightarrow \quad \mathcal{C}(\nabla f_3(\boldsymbol{x}^0)) = b(0, 0, -15)^\top$$

- The next iteration

$$\boldsymbol{x}^1 = \boldsymbol{x}^0 - \eta \frac{1}{3}\sum_{i=1}^{3} \mathcal{C}(\nabla f_i(\boldsymbol{x}^0)) = (1 + 5\eta)\boldsymbol{x}^0,$$

and then $\boldsymbol{x}^t = (1 + 5\eta)^t \boldsymbol{x}^0$ diverges exponentially.

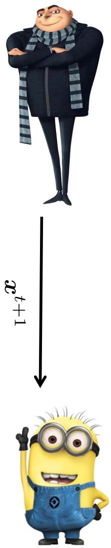
A better scheme: shift compression

(Stich et al., 2018; Richtárik et al., 2021)

- The PS updates the model:

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - \frac{\eta}{n} \sum_{i=1}^{n} \boldsymbol{g}_i^t$$

— $\boldsymbol{g}_i^t$ is the compressed surrogate of $\nabla f_i(\boldsymbol{x}^t)$



A better scheme: shift compression

(Stich et al., 2018; Richtárik et al., 2021)

- The PS updates the model:

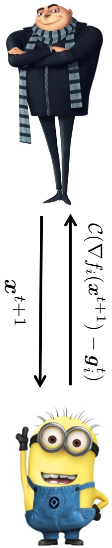
$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - \frac{\eta}{n}\sum_{i=1}^{n} \boldsymbol{g}_i^t$$

— $\boldsymbol{g}_i^t$ is the compressed surrogate of $\nabla f_i(\boldsymbol{x}^t)$

- Clients update $\boldsymbol{g}_i^t$ with a shift compression:

$$\boldsymbol{g}_i^{t+1} = \boldsymbol{g}_i^t + \underbrace{\mathcal{C}(\nabla f_i(\boldsymbol{x}^{t+1}) - \boldsymbol{g}_i^t)}_{\text{difference compression}}$$

— $\boldsymbol{g}_i^t$ is constructed accumulatively over time



Let's revisit the example

- Let $\boldsymbol{x}^0 = (b, b, b)$, and the compressor be top$_1$, $\boldsymbol{g}_i^0 = \mathcal{C}(\nabla f_i(\boldsymbol{x}^0))$, and the first iteration is still $\boldsymbol{x}^1 = (1 + 5\eta)\boldsymbol{x}^0$.

Let's revisit the example

- Let $\boldsymbol{x}^0 = (b, b, b)$, and the compressor be top$_1$, $\boldsymbol{g}_i^0 = \mathcal{C}(\nabla f_i(\boldsymbol{x}^0))$, and the first iteration is still $\boldsymbol{x}^1 = (1 + 5\eta)\boldsymbol{x}^0$.
- **Error feedback:**

$$\nabla f_1(\boldsymbol{x}^1) - \boldsymbol{g}_1^0 = b \begin{bmatrix} -75\eta \\ 13(1+5\eta) \\ 13(1+5\eta) \end{bmatrix}$$

Let's revisit the example

- Let $\boldsymbol{x}^0 = (b, b, b)$, and the compressor be top$_1$, $\boldsymbol{g}_i^0 = \mathcal{C}(\nabla f_i(\boldsymbol{x}^0))$, and the first iteration is still $\boldsymbol{x}^1 = (1 + 5\eta)\boldsymbol{x}^0$.
- **Error feedback:**

$$\nabla f_1(\boldsymbol{x}^1) - \boldsymbol{g}_1^0 = b \begin{bmatrix} -75\eta \\ 13(1+5\eta) \\ 13(1+5\eta) \end{bmatrix}$$

and as long as $\eta < 13/30$:

$$\mathcal{C}\left(\nabla f_1(\boldsymbol{x}^1) - \boldsymbol{g}_1^0\right) = b \begin{bmatrix} 0 \\ 13(1+5\eta) \\ 0 \end{bmatrix}$$

receiving information from coordinates other than the first one, leading to a better compressed gradient!

Let's revisit the example

- Let $\boldsymbol{x}^0 = (b, b, b)$, and the compressor be top$_1$, $\boldsymbol{g}_i^0 = \mathcal{C}(\nabla f_i(\boldsymbol{x}^0))$, and the first iteration is still $\boldsymbol{x}^1 = (1 + 5\eta)\boldsymbol{x}^0$.
- **Error feedback:**

$$\nabla f_1(\boldsymbol{x}^1) - \boldsymbol{g}_1^0 = b \begin{bmatrix} -75\eta \\ 13(1+5\eta) \\ 13(1+5\eta) \end{bmatrix}$$

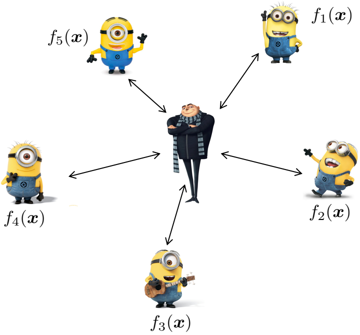
and as long as $\eta < 13/30$:

$$\mathcal{C}\left(\nabla f_1(\boldsymbol{x}^1) - \boldsymbol{g}_1^0\right) = b \begin{bmatrix} 0 \\ 13(1+5\eta) \\ 0 \end{bmatrix}$$

receiving information from coordinates other than the first one, leading to a better compressed gradient!

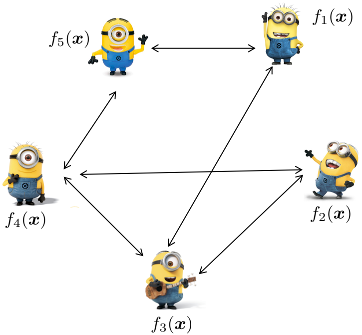
We'll consider algorithms using shift compression!

This talk: communication-compressed algorithms



Server/client model

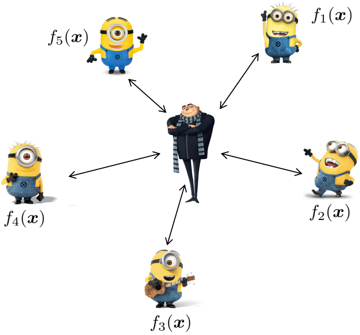
PS coordinates *global* information sharing



Network/decentralized model

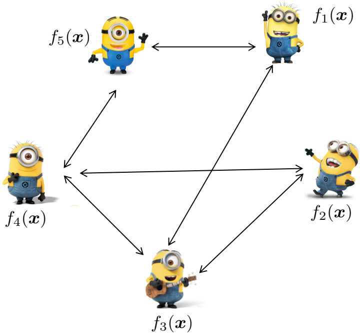
agents share *local* information over a graph topology

This talk: communication-compressed algorithms



Server/client model

PS coordinates *global* information sharing

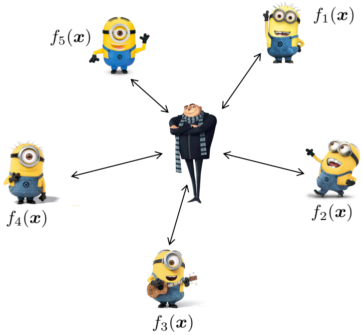


Network/decentralized model

agents share *local* information over a graph topology

Coping with heterogeneity

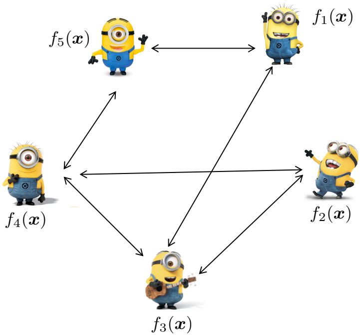
This talk: communication-compressed algorithms



Server/client model

PS coordinates *global* information sharing

Coping with privacy



Network/decentralized model

agents share *local* information over a graph topology

Coping with heterogeneity

BEER: Fast Decentralized Nonconvex Optimization with Communication Compression



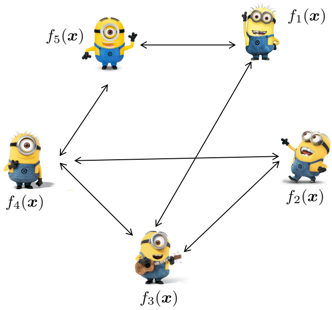
Haoyu Zhao
Princeton

Boyue Li
CMU

Zhize Li
CMU

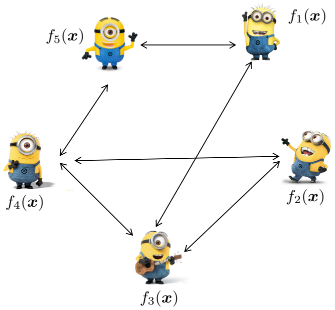
Peter Richtarik
KAUST

Decentralized nonconvex opt with compressed comm



- The mixing of information is characterized by a mixing matrix $\boldsymbol{W} = [w_{ij}] \in \mathbb{R}^{n\times n}$ aligned with the network topology.

Decentralized nonconvex opt with compressed comm



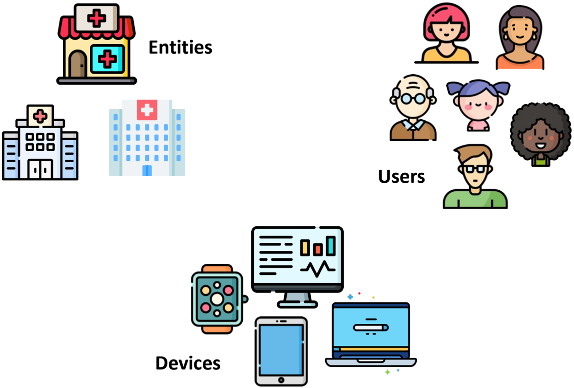
- The mixing of information is characterized by a mixing matrix $\boldsymbol{W} = [w_{ij}] \in \mathbb{R}^{n \times n}$ aligned with the network topology.
- The spectral quantity, which we call the *spectral gap*,

$$\rho \triangleq 1 - |\lambda_2(\boldsymbol{W})| \in (0, 1]$$

captures how fast information mixes over the network.

Goal: design fast-converging algorithms with communication compression

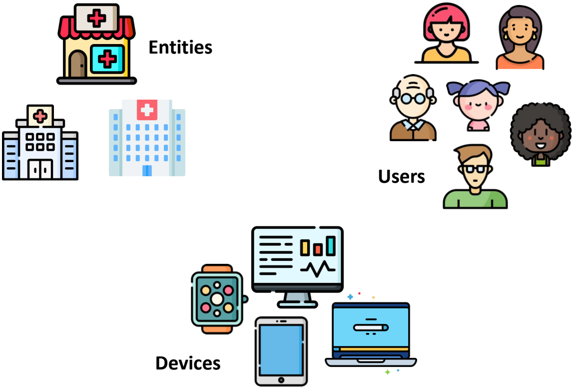
Data heterogeneity



Heterogeneity measure

$$\mathbb{E}_i \| \underbrace{\nabla f_i(\boldsymbol{x})}_{\textit{local obj.}} - \underbrace{\nabla f(\boldsymbol{x})}_{\textit{global obj.}} \|^2 \leq G^2$$

Data heterogeneity

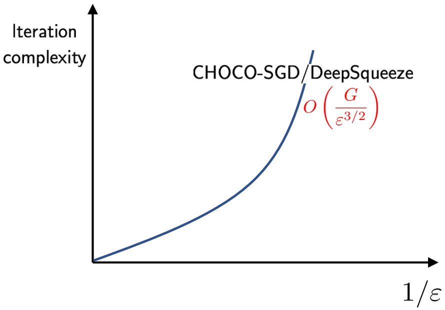


Heterogeneity measure

$$\mathbb{E}_i \| \underbrace{\nabla f_i(\boldsymbol{x})}_{\textit{local obj.}} - \underbrace{\nabla f(\boldsymbol{x})}_{\textit{global obj.}} \|^2 \leq G^2$$

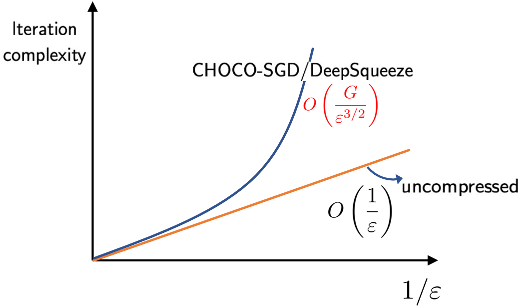
— G *can be unbounded!*

Prior art



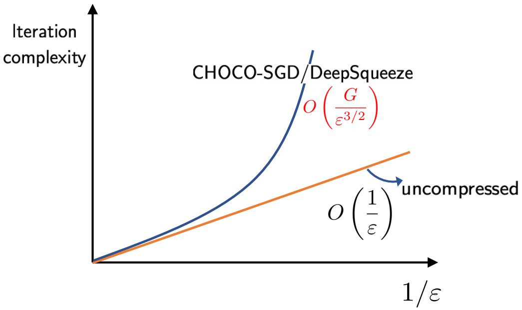
CHOCO-SGD (Koloskova et al., 2019) / DeepSqueeze (Tang et al., 2019):

Prior art



CHOCO-SGD (Koloskova et al., 2019) / DeepSqueeze (Tang et al., 2019):

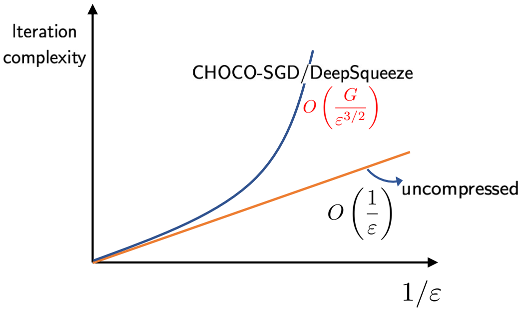
Prior art



CHOCO-SGD (Koloskova et al., 2019) / DeepSqueeze (Tang et al., 2019):

- slow convergence rates (need more communication rounds) and
- Incompatible with heterogeneity

Prior art

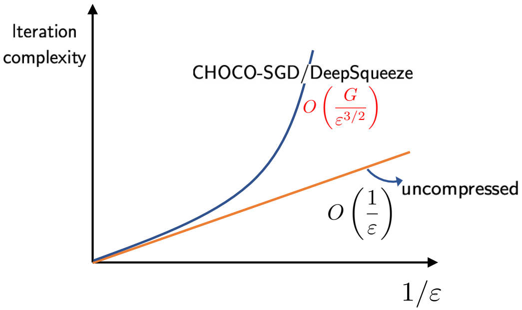


CHOCO-SGD (Koloskova et al., 2019) / DeepSqueeze (Tang et al., 2019):

- slow convergence rates (need more communication rounds) and
- Incompatible with heterogeneity

Can we converge at the rate $O\left(\frac{1}{\varepsilon}\right)$ under arbitrary heterogeneity?

Prior art



CHOCO-SGD (Koloskova et al., 2019) / DeepSqueeze (Tang et al., 2019):

- slow convergence rates (need more communication rounds) and
- Incompatible with heterogeneity

Can we converge at the rate $O\left(\frac{1}{\varepsilon}\right)$ under arbitrary heterogeneity?

Yes, by using gradient tracking!

Detour: DGD with gradient tracking

Centralized Gradient Descent (GD):

$$\boldsymbol{x}^t = \boldsymbol{x}^{t-1} - \eta \nabla f(\boldsymbol{x}^{t-1})$$

Detour: DGD with gradient tracking

Centralized Gradient Descent (GD):

$$\boldsymbol{x}^t = \boldsymbol{x}^{t-1} - \eta \nabla f(\boldsymbol{x}^{t-1})$$

Decentralized Gradient Descent (DGD):

$$\boldsymbol{x}_i^t = \underbrace{\sum\nolimits_j w_{ij} \boldsymbol{x}_j^{t-1}}_{\text{mixing}} - \eta \underbrace{\nabla f_i(\boldsymbol{x}_i^{t-1})}_{\text{local gradient}}$$

Detour: DGD with gradient tracking

Centralized Gradient Descent (GD):

$$\boldsymbol{x}^t = \boldsymbol{x}^{t-1} - \eta \nabla f(\boldsymbol{x}^{t-1})$$

Constant step size, linear convergence for strongly convex problems.

Decentralized Gradient Descent (DGD):

$$\boldsymbol{x}_i^t = \underbrace{\sum\nolimits_j w_{ij} \boldsymbol{x}_j^{t-1}}_{\text{mixing}} - \eta \underbrace{\nabla f_i(\boldsymbol{x}_i^{t-1})}_{\text{local gradient}}$$

Detour: DGD with gradient tracking

Centralized Gradient Descent (GD):

$$\boldsymbol{x}^t = \boldsymbol{x}^{t-1} - \eta \nabla f(\boldsymbol{x}^{t-1})$$

Constant step size, linear convergence for strongly convex problems.

Decentralized Gradient Descent (DGD):

$$\boldsymbol{x}_i^t = \underbrace{\sum\nolimits_j w_{ij} \boldsymbol{x}_j^{t-1}}_{\text{mixing}} - \underbrace{\eta \nabla f_i(\boldsymbol{x}_i^{t-1})}_{\text{local gradient}}$$

Constant step size, does not converge!

Detour: DGD with gradient tracking

Centralized Gradient Descent (GD):

$$\boldsymbol{x}^t = \boldsymbol{x}^{t-1} - \eta \nabla f(\boldsymbol{x}^{t-1})$$

Constant step size, linear convergence for strongly convex problems.

Decentralized Gradient Descent (DGD):

$$\boldsymbol{x}_i^t = \underbrace{\sum\nolimits_j w_{ij} \boldsymbol{x}_j^{t-1}}_{\text{mixing}} - \underbrace{\eta \nabla f_i(\boldsymbol{x}_i^{t-1})}_{\text{local gradient}}$$

Constant step size, does not converge!

At optimal point $\boldsymbol{x}^\star$: $\nabla f(\boldsymbol{x}^\star) = \mathbf{0}$, but $\nabla f_i(\boldsymbol{x}^\star) \neq \mathbf{0}$

How do we fix this?

DGD with gradient tracking

Use dynamic average consensus (Zhu and Martinez, 2010) to track the global gradient $\boldsymbol{s}_i^t$:

$$\boldsymbol{x}_i^t = \underbrace{\sum\nolimits_j w_{ij} \boldsymbol{x}_j^{t-1}}_{\text{mixing}} - \eta \boldsymbol{s}_i^t$$

$$\boldsymbol{s}_i^t = \underbrace{\sum\nolimits_j w_{ij} \boldsymbol{s}_i^{t-1}}_{\text{mixing}} + \underbrace{\nabla f_i(\boldsymbol{x}_i^t) - \nabla f_i(\boldsymbol{x}_i^{t-1})}_{\text{gradient tracking}}$$

DGD with gradient tracking

Use dynamic average consensus (Zhu and Martinez, 2010) to track the global gradient $\boldsymbol{s}_i^t$:

$$\boldsymbol{x}_i^t = \underbrace{\sum\nolimits_j w_{ij}\boldsymbol{x}_j^{t-1}}_{\text{mixing}} - \eta \boldsymbol{s}_i^t$$

$$\boldsymbol{s}_i^t = \underbrace{\sum\nolimits_j w_{ij}\boldsymbol{s}_i^{t-1}}_{\text{mixing}} + \underbrace{\nabla f_i(\boldsymbol{x}_i^t) - \nabla f_i(\boldsymbol{x}_i^{t-1})}_{\text{gradient tracking}}$$

This trick, and other alternatives, have been used extensively to fix the non-convergence issue in decentralized optimization.

DGD with gradient tracking

Use dynamic average consensus (Zhu and Martinez, 2010) to track the global gradient $\boldsymbol{s}_i^t$:

$$\boldsymbol{x}_i^t = \underbrace{\sum_j w_{ij} \boldsymbol{x}_j^{t-1}}_{\text{mixing}} - \eta \boldsymbol{s}_i^t$$

$$\boldsymbol{s}_i^t = \underbrace{\sum_j w_{ij} \boldsymbol{s}_i^{t-1}}_{\text{mixing}} + \underbrace{\nabla f_i(\boldsymbol{x}_i^t) - \nabla f_i(\boldsymbol{x}_i^{t-1})}_{\text{gradient tracking}}$$

This trick, and other alternatives, have been used extensively to fix the non-convergence issue in decentralized optimization.

- EXTRA (Shi, Ling, Wu and Yin, 2015); NEXT (Di Lorenzo and Scutari, 2016); NIDS (Li, Shi, Yan, 2017); ADD-OPT (Xi, Xin, and Khan, 2017); DIGING (Nedic, Olshevsky, and Shi, 2017); DGD (Qu and Li, 2018);
- many, many more...

BEER: gradient tracking + shift compression

$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n]$: local models.
$\nabla F(\boldsymbol{X}) = [\nabla f_1(\boldsymbol{x}_1), \nabla f_2(\boldsymbol{x}_2), \cdots, \nabla f_n(\boldsymbol{x}_n)]$: local gradients.

BEER: gradient tracking + shift compression

$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n]$: local models.
$\nabla F(\boldsymbol{X}) = [\nabla f_1(\boldsymbol{x}_1), \nabla f_2(\boldsymbol{x}_2), \cdots, \nabla f_n(\boldsymbol{x}_n)]$: local gradients.

- **model update:**

$$\boldsymbol{X}^{t+1} = \boldsymbol{X}^t + \gamma \underbrace{\boldsymbol{H}^t(\boldsymbol{W} - \boldsymbol{I})}_{\text{mixing}} - \eta \underbrace{\boldsymbol{V}^t}_{\text{gradient}}$$

  where $\boldsymbol{H}^t$ is the accumulated compressed surrogate of $\boldsymbol{X}^t$, and $\boldsymbol{V}^t$ is the global gradient estimates across the agents.

BEER: gradient tracking + shift compression

$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n]$: local models.
$\nabla F(\boldsymbol{X}) = [\nabla f_1(\boldsymbol{x}_1), \nabla f_2(\boldsymbol{x}_2), \cdots, \nabla f_n(\boldsymbol{x}_n)]$: local gradients.

- **model update:**

$$\boldsymbol{X}^{t+1} = \boldsymbol{X}^t + \gamma \underbrace{\boldsymbol{H}^t(\boldsymbol{W} - \boldsymbol{I})}_{\text{mixing}} - \eta \underbrace{\boldsymbol{V}^t}_{\text{gradient}}$$

  where $\boldsymbol{H}^t$ is the accumulated compressed surrogate of $\boldsymbol{X}^t$, and $\boldsymbol{V}^t$ is the global gradient estimates across the agents.

- **gradient tracking:**

$$\boldsymbol{V}^{t+1} = \boldsymbol{V}^t + \gamma \underbrace{\boldsymbol{G}^t(\boldsymbol{W} - \boldsymbol{I})}_{\text{mixing}} + \underbrace{\nabla F(\boldsymbol{X}^{t+1}) - \nabla F(\boldsymbol{X}^t)}_{\text{gradient tracking}},$$

  where $\boldsymbol{G}^t$ is the accumulated compressed surrogate of $\boldsymbol{V}^t$.

BEER: gradient tracking + shift compression

$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n]$: local models.
$\nabla F(\boldsymbol{X}) = [\nabla f_1(\boldsymbol{x}_1), \nabla f_2(\boldsymbol{x}_2), \cdots, \nabla f_n(\boldsymbol{x}_n)]$: local gradients.

- **model update:**

$$\boldsymbol{X}^{t+1} = \boldsymbol{X}^t + \gamma \underbrace{\boldsymbol{H}^t(\boldsymbol{W} - \boldsymbol{I})}_{\text{mixing}} - \eta \underbrace{\boldsymbol{V}^t}_{\text{gradient}}$$

  where $\boldsymbol{H}^t$ is the accumulated compressed surrogate of $\boldsymbol{X}^t$, and $\boldsymbol{V}^t$ is the global gradient estimates across the agents.

- **gradient tracking:**

$$\boldsymbol{V}^{t+1} = \boldsymbol{V}^t + \gamma \underbrace{\boldsymbol{G}^t(\boldsymbol{W} - \boldsymbol{I})}_{\text{mixing}} + \underbrace{\nabla F(\boldsymbol{X}^{t+1}) - \nabla F(\boldsymbol{X}^t)}_{\text{gradient tracking}},$$

  where $\boldsymbol{G}^t$ is the accumulated compressed surrogate of $\boldsymbol{V}^t$.

- Both $\boldsymbol{H}^t$ and $\boldsymbol{G}^t$ are updated using **shift compression**.

Theoretical convergence of BEER

Theorem (Zhao et al., 2022)

To achieve $\mathbb{E}\|\nabla f(\boldsymbol{x}^{\text{output}})\|^2 \leq \varepsilon$, *BEER requires at most*

$$O\left(\frac{1}{\rho^3 \alpha \varepsilon}\right)$$

communication rounds, without the bounded heterogeneity assumption. Here, α *is the compression ratio,* β *is the spectral gap of the network.*

Theoretical convergence of BEER

Theorem (Zhao et al., 2022)

To achieve $\mathbb{E}\|\nabla f(\boldsymbol{x}^{\text{output}})\|^2 \le \varepsilon$, *BEER requires at most*

$$O\left(\frac{1}{\rho^3 \alpha \varepsilon}\right)$$

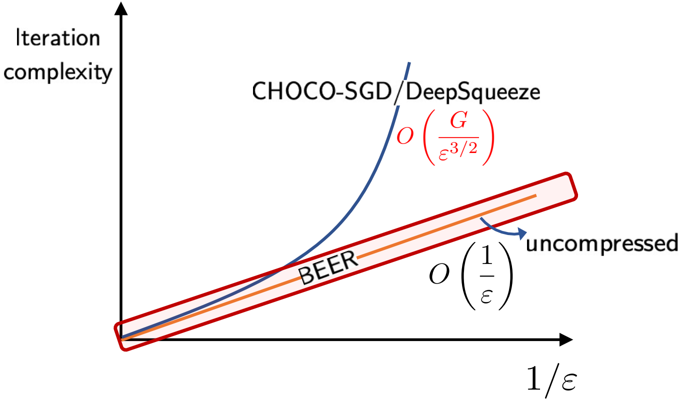
communication rounds, without the bounded heterogeneity assumption. Here, α *is the compression ratio,* β *is the spectral gap of the network.*

- Assuming constant α and ρ, the convergence rate of BEER is

$$O\left(\frac{1}{\varepsilon}\right).$$

- Our result can also be extended to using stochastic gradients.

Theoretical convergence of BEER



BEER converges at the rate $O\left(\frac{1}{\varepsilon}\right)$ under arbitrary heterogeneity!

BEER vs CHOCO-SGD

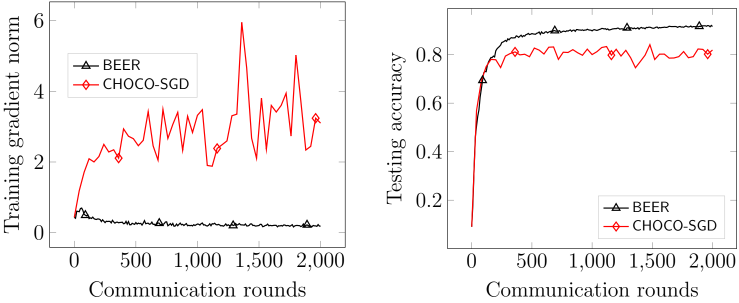


Figure: Training gradient norm and testing accuracy against communication rounds for classification on the *unshuffled* MNIST dataset using a simple neural network. Both BEER and CHOCO-SGD employ the biased gsgd_b compression with $b = 20$.

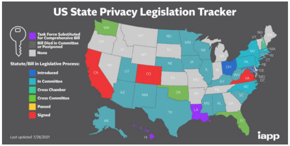
SoteriaFL: A Unified Framework for Private FL with Communication Compression

Zhize Li
CMU

Haoyu Zhao
Princeton

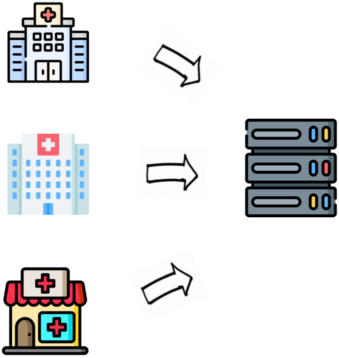
Boyue Li
CMU

A little privacy, please



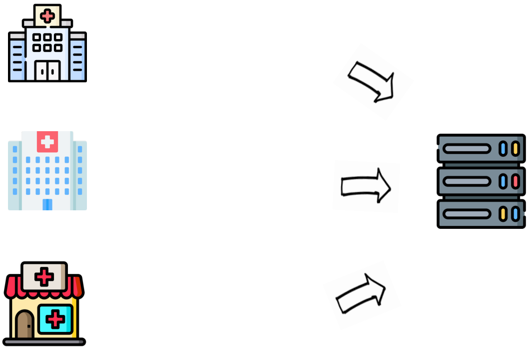
Privacy guarantees are becoming increasingly critical!

Protecting local privacy via differential privacy



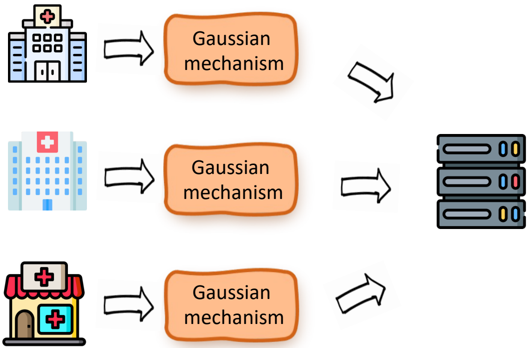
Introducing local differential privacy to guarantee the client privacy

Protecting local privacy via differential privacy



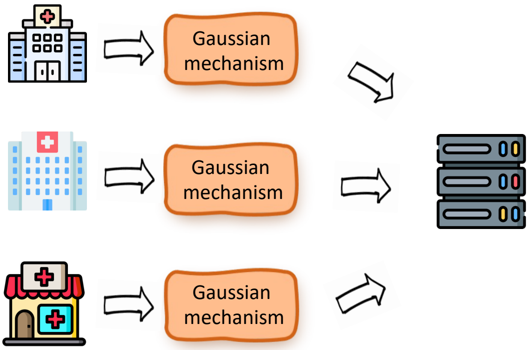
Introducing local differential privacy to guarantee the client privacy

Protecting local privacy via differential privacy



Introducing local differential privacy to guarantee the client privacy

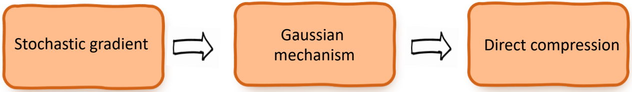
Protecting local privacy via differential privacy



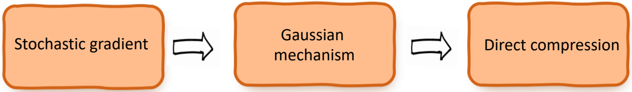
Introducing local differential privacy to guarantee the client privacy

— used by Google, Apple, etc in products

Warm-up: a direct compression approach (CDP-SGD)



Warm-up: a direct compression approach (CDP-SGD)



Theorem (Li et al., 2022)

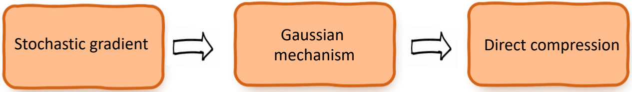
CDP-SGD achieves (ϵ, δ)-LDP, and the utility

$$\mathbb{E}\|\nabla f(\boldsymbol{x}^{\text{output}})\|^2 \lesssim \frac{1}{m\epsilon}\sqrt{\frac{d\log(1/\delta)}{\alpha n}}$$

within communication complexity on the order of

$$n^{3/2}\alpha^{3/2}m\epsilon\sqrt{\frac{d}{\log(1/\delta)}} + \frac{\alpha n m^2 \epsilon^2}{\log(1/\delta)}.$$

Warm-up: a direct compression approach (CDP-SGD)



Theorem (Li et al., 2022)

CDP-SGD achieves (ϵ, δ)-LDP, and the utility

$$\mathbb{E}\|\nabla f(\boldsymbol{x}^{\text{output}})\|^2 \lesssim \frac{1}{m\epsilon}\sqrt{\frac{d\log(1/\delta)}{\alpha n}}$$

within communication complexity on the order of

$$n^{3/2}\alpha^{3/2}m\epsilon\sqrt{\frac{d}{\log(1/\delta)}} + \frac{\alpha n m^2 \epsilon^2}{\log(1/\delta)}.$$

- Larger $\frac{\sqrt{\log(1/\delta)}}{\epsilon}$ gives stronger privacy, worse accuracy, fewer communication.

Warm-up: a direct compression approach (CDP-SGD)

Stochastic gradient $\Rightarrow$ Gaussian mechanism $\Rightarrow$ Direct compression

Theorem (Li et al., 2022)

CDP-SGD achieves (ϵ, δ)*-LDP, and the utility*

$$\mathbb{E}\|\nabla f(\boldsymbol{x}^{\text{output}})\|^2 \lesssim \frac{1}{m\epsilon}\sqrt{\frac{d\log(1/\delta)}{\alpha n}}$$

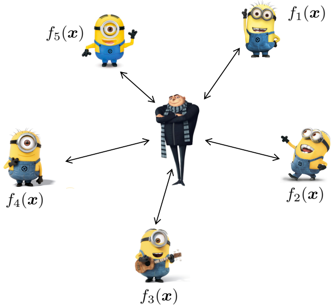
within communication complexity on the order of

$$n^{3/2}\alpha^{3/2}m\epsilon\sqrt{\frac{d}{\log(1/\delta)}} + \frac{\alpha n m^2 \epsilon^2}{\log(1/\delta)}.$$

- Larger $\frac{\sqrt{\log(1/\delta)}}{\epsilon}$ gives stronger privacy, worse accuracy, fewer communication.
- Caveat: the communication complexity is $O(m^2)$ when the local data size m is dominating.

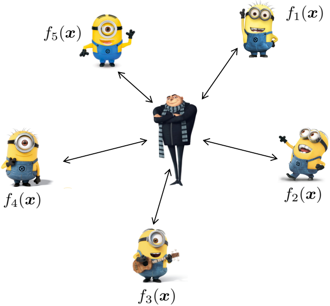
Better compression and compute: a unified framework?

- **Compression:** shift compression with many options, e.g. sparsification or quantization
- **Computation:** stochastic local gradient estimators with many options, e.g. SGD, SVRG or SAGA



Better compression and compute: a unified framework?

- **Compression:** shift compression with many options, e.g. sparsification or quantization
- **Computation:** stochastic local gradient estimators with many options, e.g. SGD, SVRG or SAGA



Can we develop a unified framework for private FL with compression, with a characterization of the privacy-utility-communication trade-off?

SoteriaFL: a unified framework for compressed private FL

Local gradient estimator $\Rightarrow$ Gaussian mechanism $\Rightarrow$ Shift compression $\Rightarrow$ Shift update

SOTERIA
GODDESS OF SAFETY

Highlights of SoteriaFL:

- Flexible local gradient estimators
- Protect local data privacy
- State-of-the-art shift compression scheme
- Privacy-utility-communication trade-offs

Performance of SoteriaFL

Theorem (Li et al., 2022)

When $n \geq 1/\alpha^3$, *SoteriaFL—with SGD, GD, SVRG, SAGA—achieves* (ϵ, δ)*-LDP, and the utility*

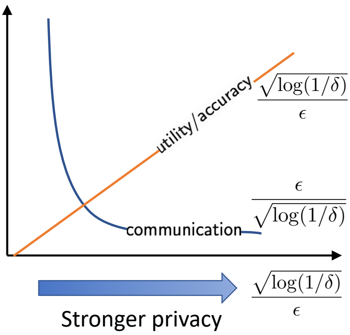
$$\mathbb{E}\|\nabla f(\boldsymbol{x}^{\text{output}})\|^2 \lesssim \frac{1}{m\epsilon}\sqrt{\frac{d\log(1/\delta)}{\alpha n}}$$

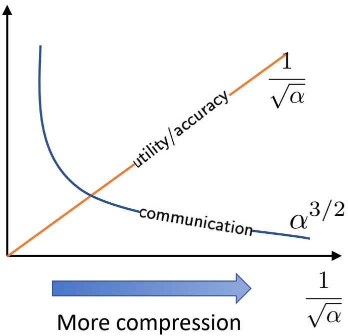
with communication complexity on the order of

$$n^{3/2}\alpha^{3/2}m\epsilon\sqrt{\frac{d}{\log(1/\delta)}}.$$

- Communication complexity is linear in m, better than CDP-SGD!
- Our analysis applies to unbiased compressions, and adapts to other gradient estimators too.

Privacy-utility-communication trade-off





- Stronger privacy, worse accuracy, fewer communication
- More compression, worse accuracy, fewer communication

Numerical experiments

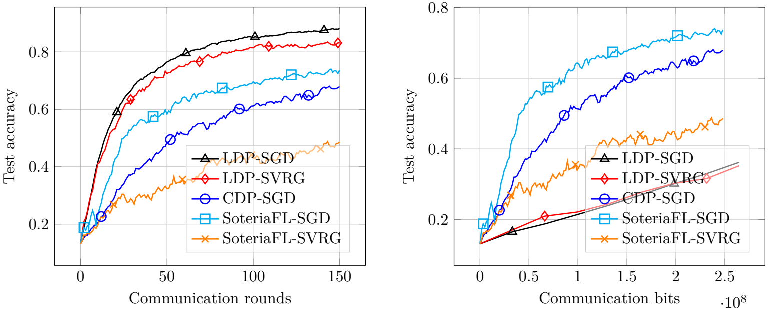
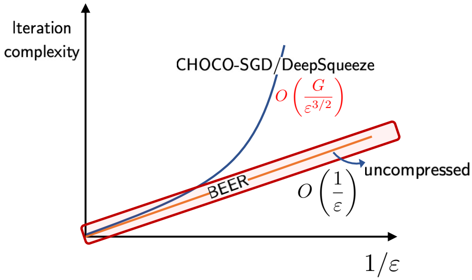
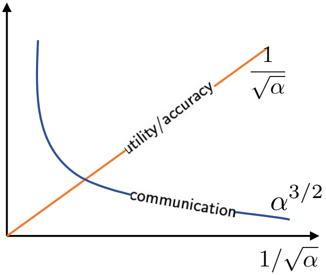


Figure: Shallow NN training on the MNIST dataset under $(1, 10^{-3})$-LDP.

Summary





Provably efficient communication-compressed FL algorithms for heterogeneous and private data!

Future work:

- Client-adaptive privacy-preserving decentralized algorithms under data heterogeneity.

Thank you!

1. BEER: Fast $O(1/T)$ Rate for Decentralized Nonconvex Optimization with Communication Compression
   H. Zhao, B. Li, Z. Li, P. Richtarik, and Y. Chi, arXiv:2201.13320, NeurIPS 2022.
2. SoteriaFL: A Unified Framework for Private Federated Learning with Communication Compression
   Z. Li, H. Zhao, B. Li, and Y. Chi, arXiv:2206.09888, NeurIPS 2022.