Coping with Heterogeneity and Privacy in Communication-Efficient Federated Optimization

Yuejie Chi

Carnegie Mellon University

FedVision@CVPR
Jun. 2022

Acknowledgements

Zhize Li
CMU

Boyue Li
CMU

Haoyu Zhao
Princeton

Peter Richtarik
KAUST

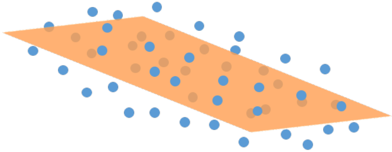
Empirical Risk Minimization (ERM)

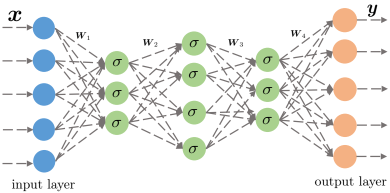
Given a set of data $\mathcal{M}$,

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

Here, N = number of total samples.

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



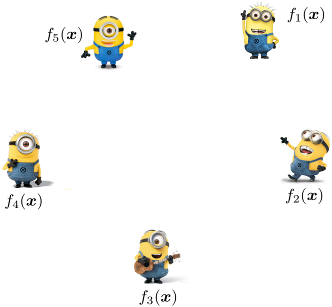


Distributed ERM

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:

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

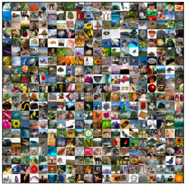


n = number of agents

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

Challenges in federated/decentralized learning

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


Communication efficiency

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

Communication efficiency

Communication cost = Communication rounds × Cost per round

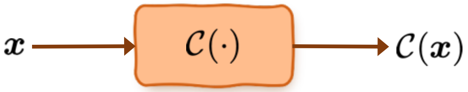
- **Local method:** perform more local computation to reduce communication rounds, e.g. FedAvg (McMahan et al., 2016).

Communication efficiency

Communication cost = Communication rounds × 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).

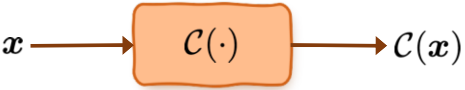


Communication efficiency

Communication cost = Communication rounds × Cost per round

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

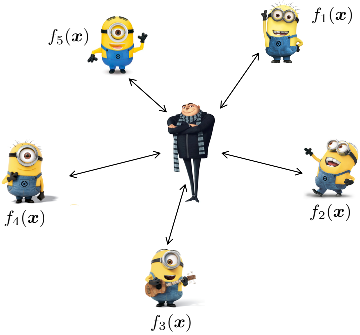
- **Communication compression:** compress the message into fewer bits, e.g. sparsification or quantization (Alistarh et al., 2017).



We will focus on communication compression methods.

Two distributed schemes

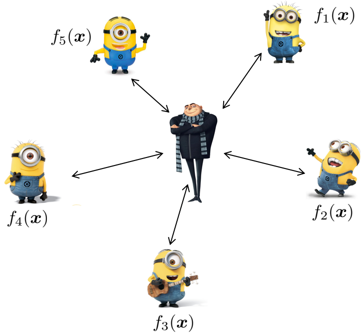
Two distributed schemes



Server/client model

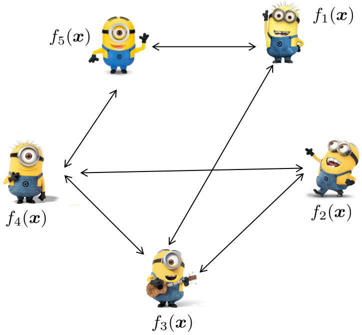
PS coordinates *global* information sharing

Two distributed schemes



Server/client model

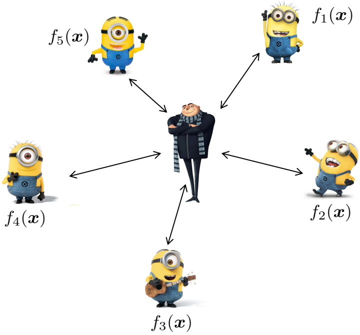
PS coordinates *global* information sharing



Network/decentralized model

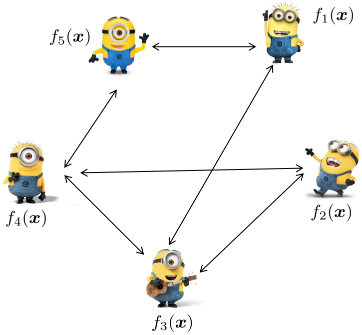
agents share *local* information over a graph topology

Two distributed schemes



Server/client model

PS coordinates *global* information sharing

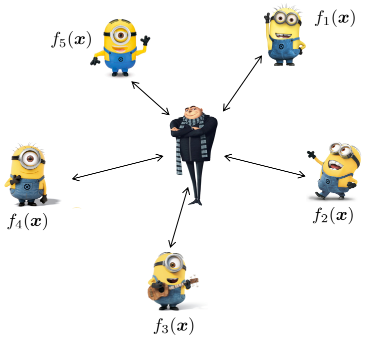


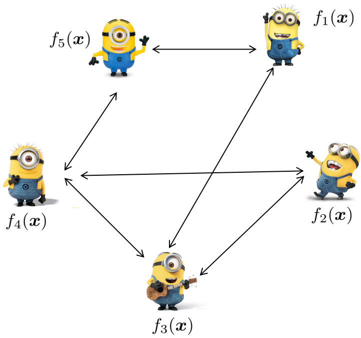
Network/decentralized model

agents share *local* information over a graph topology

Coping with heterogeneity

Two distributed schemes





Server/client model

PS coordinates *global* information sharing

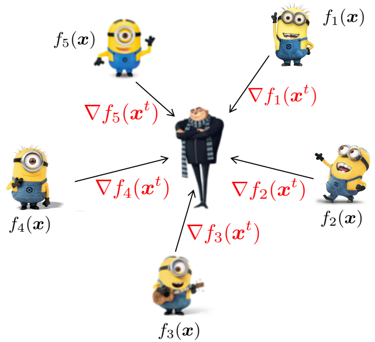
Coping with privacy

Network/decentralized model

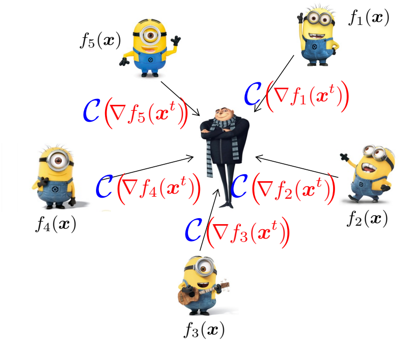
agents share *local* information over a graph topology

Coping with heterogeneity

A prelude: what should we compress?



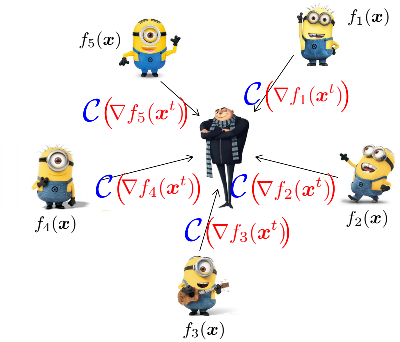
A prelude: what should we compress?



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?



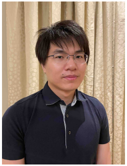
What about

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

Somewhat surprisingly, *direct compression* doesn't work!

A counter-example

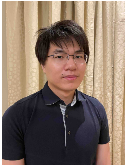
Consider $n = 3$ and let $f_i(x) = (\boldsymbol{a}_i^\top \boldsymbol{x})^2 + \frac{1}{2}\|\boldsymbol{x}\|^2$, where $\boldsymbol{a}_1 = (-4, 3, 3)^\top$, $\boldsymbol{a}_2 = (3, -4, 3)^\top$ and $\boldsymbol{a}_3 = (3, 3, -4)^\top$.



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 $\boldsymbol{a}_3 = (3, 3, -4)^\top$.



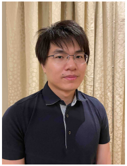
Zhize Li

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

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

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)

- Model update:

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

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

A better scheme: shift compression

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

- Model update:

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

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

- 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

A better scheme: shift compression

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

- Model update:

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

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

- 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

We'll consider algorithms using shift compression!

BEER: Fast Decentralized Nonconvex Optimization with Communication Compression

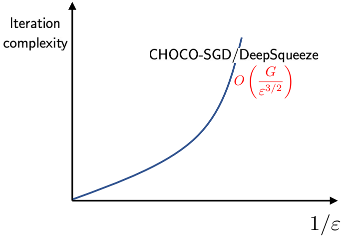
Haoyu Zhao
Princeton

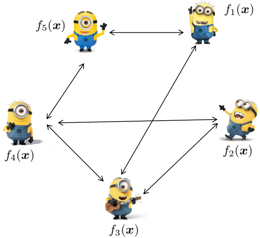
Boyue Li
CMU

Zhize Li
CMU

Peter Richtarik
KAUST

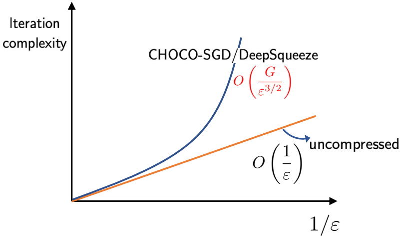
Prior art

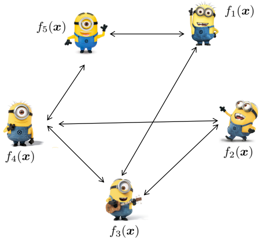




CHOCO-SGD (Koloskova et al., 2019) / DeepSqueeze (Tang et al., 2019):

Prior art



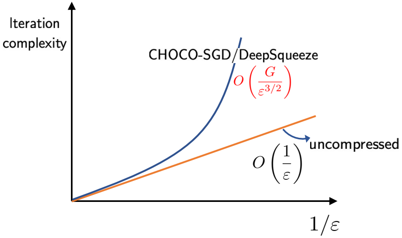


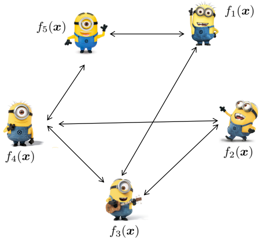
CHOCO-SGD (Koloskova et al., 2019) / DeepSqueeze (Tang et al., 2019):

- slow convergence rates (need more communication rounds) and
- Incompatible with heterogeneity: bounded gradient or dissimilarity

$$\mathbb{E}_{\xi_i \sim \mathcal{D}_i} \|\nabla f(\boldsymbol{x}; \xi_i)\| \leq G^2 \quad \text{or} \quad \mathbb{E}_i \|\nabla f_i(\boldsymbol{x}) - \nabla f(\boldsymbol{x})\| \leq G^2$$

Prior art





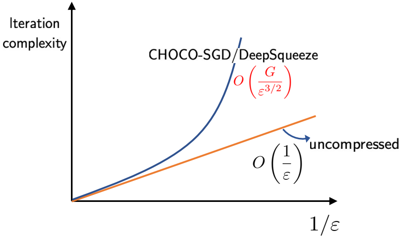
CHOCO-SGD (Koloskova et al., 2019) / DeepSqueeze (Tang et al., 2019):

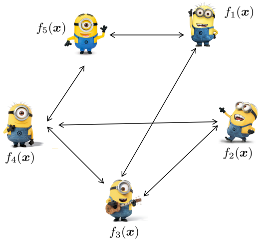
- slow convergence rates (need more communication rounds) and
- Incompatible with heterogeneity: bounded gradient or dissimilarity

$$\mathbb{E}_{\xi_i \sim \mathcal{D}_i} \|\nabla f(\boldsymbol{x}; \xi_i)\| \leq G^2 \quad \text{or} \quad \mathbb{E}_i \|\nabla f_i(\boldsymbol{x}) - \nabla f(\boldsymbol{x})\| \leq G^2$$

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: bounded gradient or dissimilarity

$$\mathbb{E}_{\xi_i\sim\mathcal{D}_i}\|\nabla f(\boldsymbol{x};\xi_i)\| \le G^2 \quad \text{or} \quad \mathbb{E}_i\|\nabla f_i(\boldsymbol{x}) - \nabla f(\boldsymbol{x})\| \le G^2$$

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

Yes, by using gradient tracking!

Decentralized gradient descent: a naive extension

Centralized Gradient Descent (GD):

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

Decentralized gradient descent: a naive extension

Centralized Gradient Descent (GD):

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

Decentralized Gradient Descent (DGD):

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

Decentralized gradient descent: a naive extension

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):

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

Decentralized gradient descent: a naive extension

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):

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

Constant step size, does not converge!

Decentralized gradient descent: a naive extension

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):

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

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 $\boldsymbol{s}_i^t$:

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

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

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

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

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

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

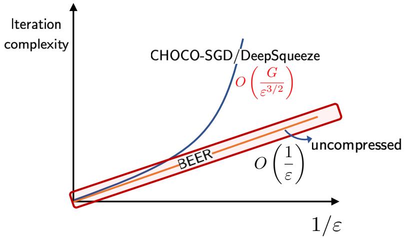
Theoretical convergence of BEER

Theorem (Zhao et al., 2022)

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

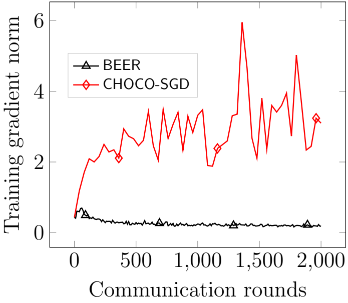
$$O\left(\frac{1}{\rho^3 \alpha \varepsilon}\right)$$

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



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

BEER vs CHOCO-SGD



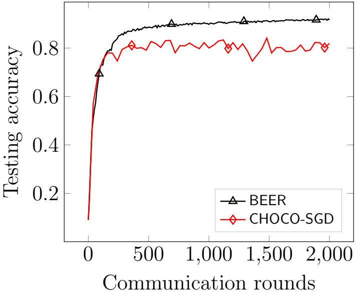


Figure: Training gradient norm and testing accuracy against communication rounds for classification on the *unshuffled* MNIST dataset using a simple neural network. Both BEER and CHOCO-SGD employ the biased gsgd_b compression with $b = 20$.

SoteriaFL: A Unified Framework for Private FL with Communication Compression



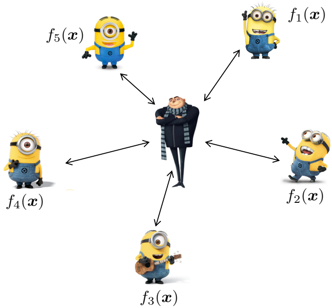
Zhize Li
CMU

Haoyu Zhao
Princeton

Boyue Li
CMU

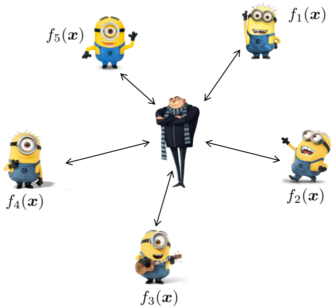
Motivation: a unified framework?

- **Privacy:** need to preserve the privacy of local data
- **Communication:** shift compression with many options, e.g. sparsification or quantization
- **Computation:** stochastic local gradient estimators with many options, e.g. SGD, SVRG or SAGA



Motivation: a unified framework?

- **Privacy:** need to preserve the privacy of local data
- **Communication:** 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

Highlights of SoteriaFL:

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

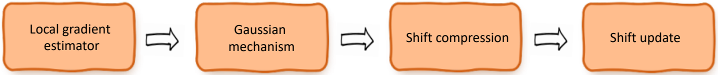
SoteriaFL: a unified framework for compressed private FL

SOTERIA
GODDESS OF SAFETY

Highlights of SoteriaFL:

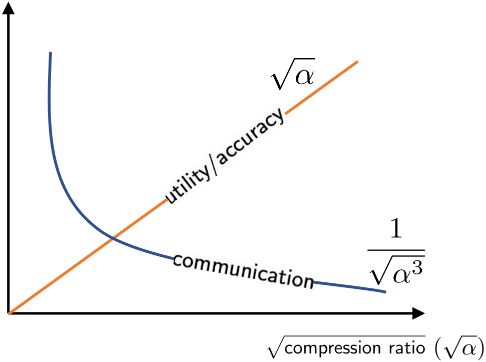
- Flexible local gradient estimators
- Protect local data privacy
- State-of-the-art shift compression scheme
- Privacy-utility-communication trade-offs

At each client:



At the server:

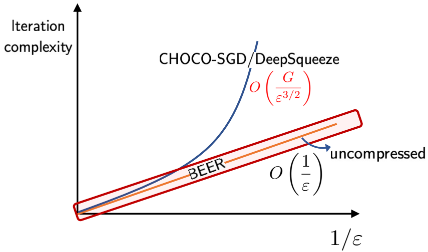
Privacy-utility-communication trade-off

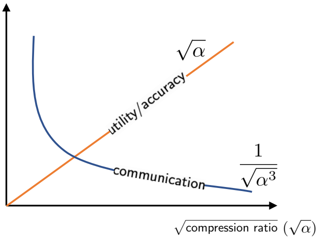


Under (ϵ, δ) local differential privacy:

- Utility/accuracy: $\frac{\sqrt{\alpha \log(1/\delta)}}{\epsilon}$
- Communication: $\frac{\epsilon}{\sqrt{\alpha^3 \log(1/\delta)}}$

Summary





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

Future work:

- 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.
2. SoteriaFL: A Unified Framework for Private Federated Learning with Communication Compression
   Z. Li, H. Zhao, B. Li, and Y. Chi, arXiv today.