Communication Efficient Distributed Learning

An half-day tutorial held in IJCAI 2021

Xiaorui Liu[1], Yao Li[2,3], Ming Yan[3,2], and Jiliang Tang[1]

Website: `https://lxiaorui.github.io/distopt/`

[1] Department of Computer Science and Engineering
[2] Department of Mathematics
[3] Department of Computational Mathematics, Science and Engineering
Michigan State University

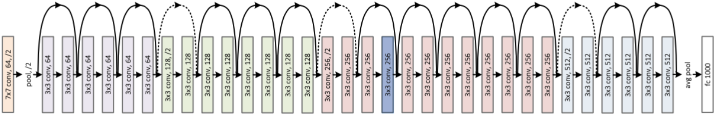
IJCAI Montreal, Aug. 20th, 2021

Outline

Table of Contents

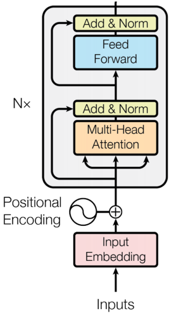
Machine Learning on Big Data

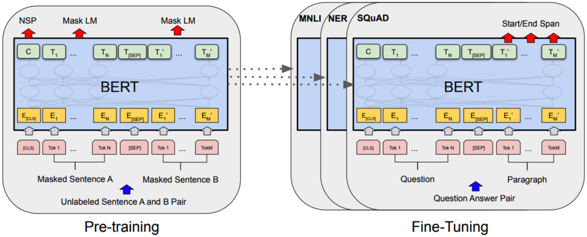
Image Recognition



Deep Residual Network

Language Model



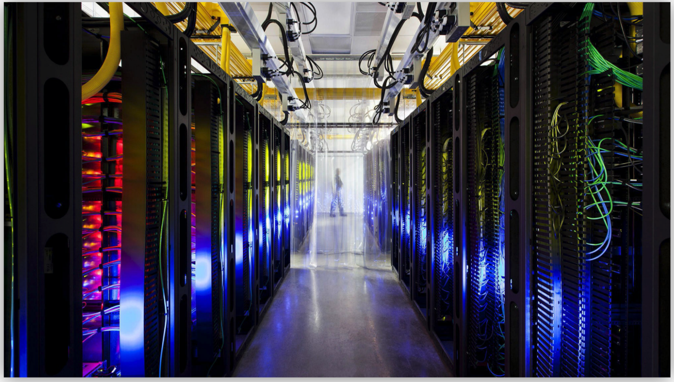


A multi-layer bidirectional Transformer

- Language modeling
- Reading comprehension
- Machine translation
- News article generation
- Question answering
- Grammar correction

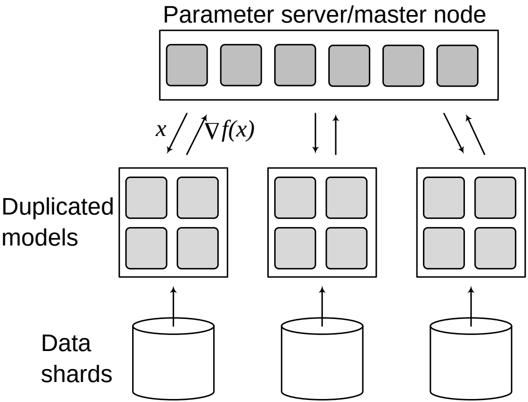
Large-scale learning problems: (big data + big model)

$$\underset{\mathbf{x}\in\mathbb{R}^p}{\text{minimize}} \;\; f(\mathbf{x}) + R(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\mathbb{E}_{\xi\sim\mathcal{D}_i}[\ell_i(\mathbf{x},\xi)]}_{:=f_i(\mathbf{x})} + R(\mathbf{x}).$$



Distributed computing for processing massive data and big models

Parallel SGD



Parallel SGD coordinated by the parameter server

Time cost per iteration: gradient computation + communication + model update

$$\text{Communication time} = \text{latency} + \frac{\text{model size}}{\text{network bandwidth}}$$

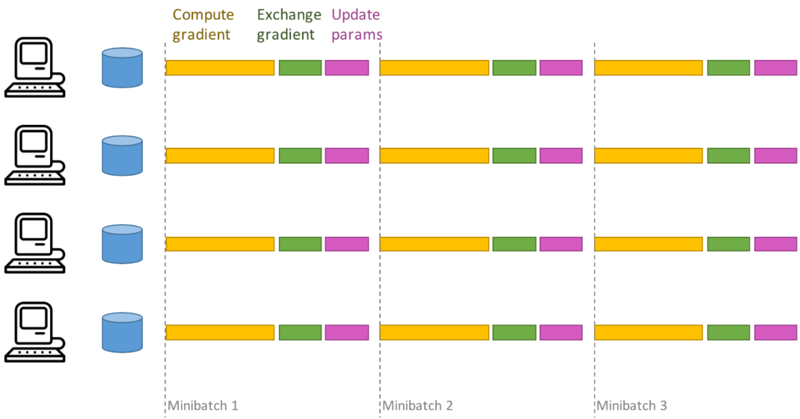


Image credit: Dan Alistarh

Ideally, communication is **fast** if model is small and bandwidth is large.

Parallel SGD

$$\text{Communication time} = \text{latency} + \frac{\text{model size}}{\text{network bandwidth}}$$

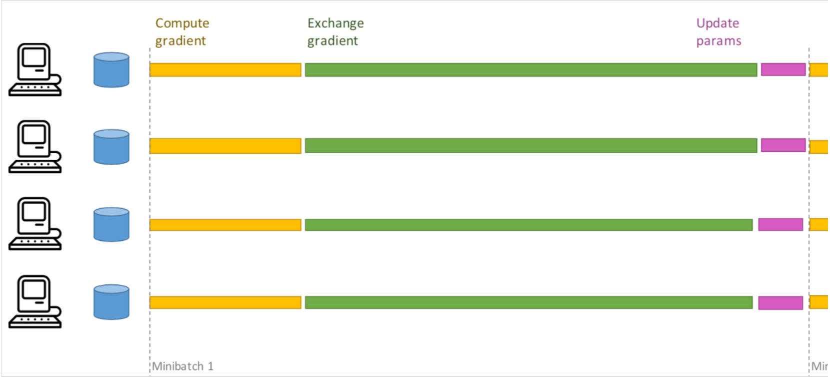


Image credit: Dan Alistarh

Practically, communication is **slow** if model is large and bandwidth is limited.

Scalability

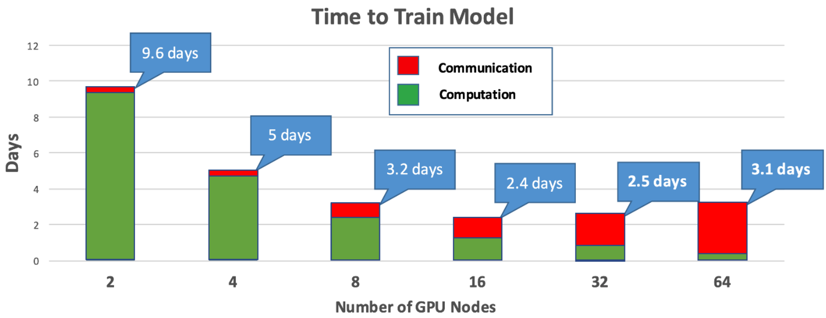
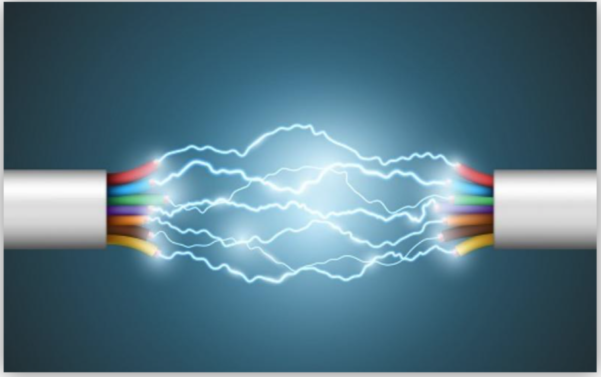


Image credit: Dan Alistarh

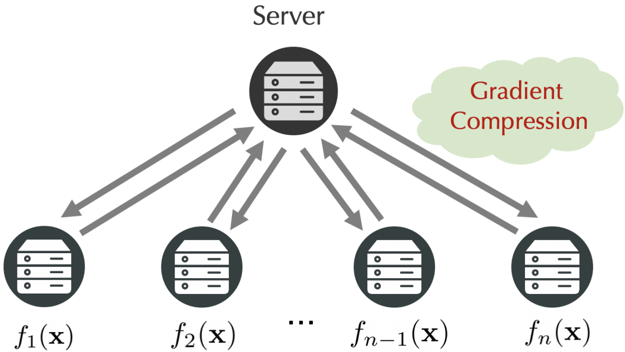
- When the number of nodes increases, the communication becomes even slower.
- The speedup becomes worse and we loose the benefit of distributed learning.

Communication Efficiency



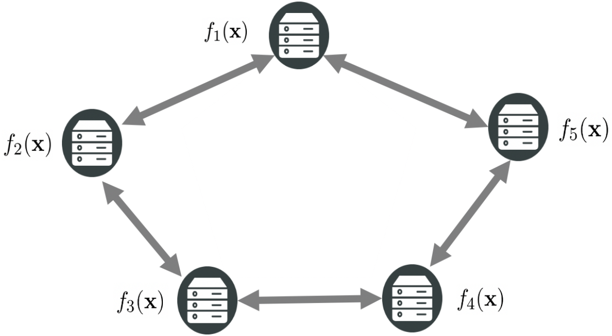
Can we design **communication efficient algorithms** to overcome the **communication bottleneck** for a better speedup and scalability?

Communication Compression

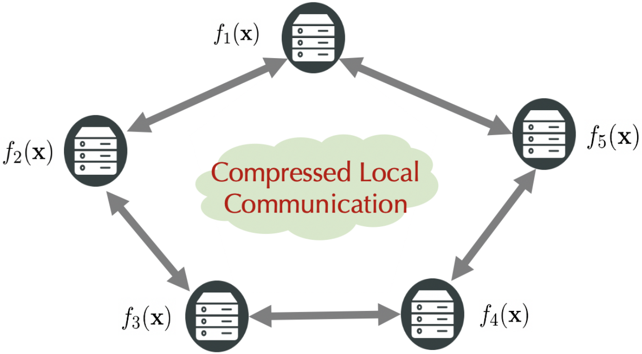


Compressing the synchronization information into low-bit representations

Decentralized Communication



Supporting general network topology by neighboring communication



Compressing the local communication between connected machines

Table of Contents

Optimization Problem

- Optimization problem

$$\underset{\mathbf{x}\in\mathbb{R}^p}{\text{minimize}}\ \mathbf{f}(\mathbf{x})+\mathbf{r}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\underbrace{\mathbb{E}_{\xi\sim\mathcal{D}_i}[f_i(\mathbf{x},\xi)]}_{:=f_i(\mathbf{x})}+\mathbf{r}(\mathbf{x}).$$

  We mainly focus on the case $\mathbf{r}(\mathbf{x}) = 0$ in this section.
  The function $f_i(\cdot,\xi)$ is differentiable.
  Examples: empirical risk minimization in statistical machine learning

- Two ways to measure the convergence of an optimization algorithm
  - How many iterations do we need to achieve some optimality precision ϵ?
  - What is the optimality precision ϵ can we achieve after K iteration?

- Convex: for $\alpha \in (0,1)$

$$f(\alpha \mathbf{x} + (1-\alpha)\mathbf{y})) \leq \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y})$$

- (μ-strongly) convex and differential: for any $\mathbf{y}$

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{\mu}{2} \|\mathbf{y} - \mathbf{x}\|^2$$

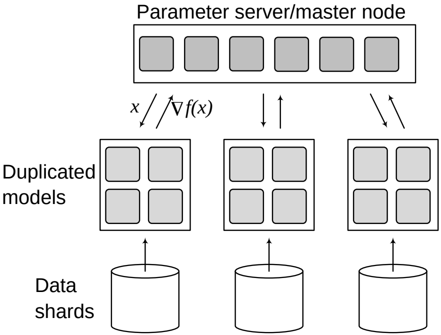
- L-smooth:

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{L}{2} \|\mathbf{y} - \mathbf{x}\|^2$$

- Proximal operator:

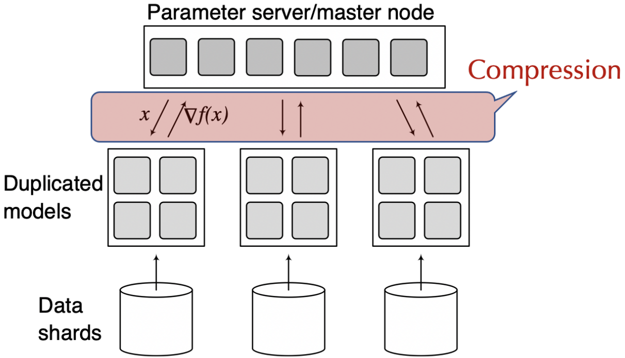
$$\mathbf{prox}_{\eta r}(\mathbf{x}) = \arg\min_{\mathbf{y}} \eta r(\mathbf{y}) + \frac{1}{2} \|\mathbf{y} - \mathbf{x}\|^2$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \eta \nabla \mathbf{f}(\mathbf{x}^k) = \mathbf{x}^k - \frac{\eta}{n} \sum_{i=1}^{n} \nabla f_i(\mathbf{x}^k)$$



- Data is partitioned at different nodes.
- Each worker node sends the gradient to the master node.
- The master node updates the model **x** and sends to the worker nodes.

Communication Compression



- Reduce the number of bits in communication
- Examples: 1Bit SGD, QSGD, Terngrad, signSGD, ECQ-SGD, DIANA, MEM-SGD, DoubleSqueeze, DORE, etc.

- Define $Q(\mathbf{x})$ as the compressed version of $\mathbf{x}$, and it can be encoded with fewer bits
- $Q(\mathbf{x})$ can be deterministic or stochastic
- $Q(\mathbf{x})$ can be biased ($\mathbb{E}Q(\mathbf{x}) = \mathbf{x}$) or unbiased ($\mathbb{E}Q(\mathbf{x}) \neq \mathbf{x}$)
- C-contracted operator:

$$\mathbb{E}\|\mathbf{x} - Q(\mathbf{x})\|^2 \leq C\|\mathbf{x}\|^2$$

  C controls the compression error.

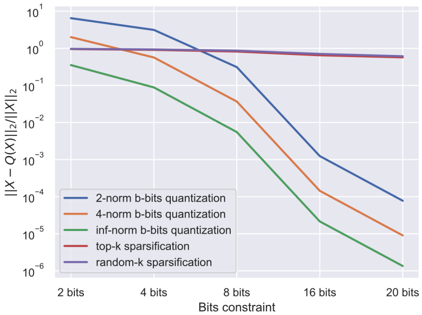
Compression Operator: Example

- No compression: $C = 0$
- Top-K sparsification: select top K elements according to the magnitudes (biased, $C < 1$)
- Rand-K sparsification: randomly select K elements and rescale it (unbiased, $C < 1$)
- p-norm b-bit quantization: (unbiased, $C > 0$)

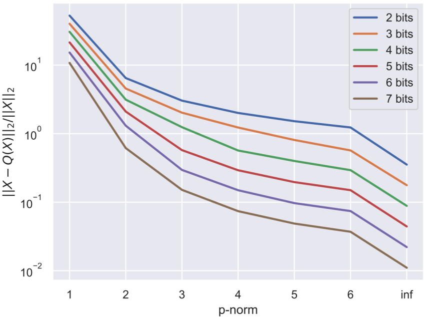
$$Q_\infty(\mathbf{x}) := \left(\|\mathbf{x}\|_p 2^{-(b-1)} \mathrm{sign}(\mathbf{x})\right) \cdot \left\lfloor \frac{2^{(b-1)}|\mathbf{x}|}{\|\mathbf{x}\|_p} + \mathbf{u} \right\rfloor \quad (1)$$

  Example: $[1.2, -0.1] \Rightarrow ([1, 0], \|[1.2, -0.1\|_p)$
  It can also be done separately in each data block to reduce compression error [Mishchenko et al. '19].
- Others

- Evaluated on 100 uniformly generated random vectors in $\mathbb{R}^{10000}$ [Liu et al. '21]



Comparison of compression error between different compression operators



Comparison of compression error for p-norm b-bit quantization

Centralized Learning

- Non-compressed version:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \eta \nabla \mathbf{f}(\mathbf{x}^k) = \mathbf{x}^k - \frac{\eta}{n} \sum_{i=1}^{n} \nabla f_i(\mathbf{x}^k)$$

- Compressed framework:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \eta \nabla \mathbf{f}(\mathbf{x}^k) = \mathbf{x}^k - \frac{\eta}{n} \sum_{i=1}^{n} \mathbf{g}_i^k,$$

where $\mathbf{g}_i^k$ is an approximation of $\nabla f_i(\mathbf{x}^k)$.

Compression Technique: Direct Compression (Q-SGD)

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{\eta}{n}\sum_{i=1}^{n} Q(\nabla f_i(\mathbf{x}^k))$$

- Assume that all nodes have the true solution $\mathbf{x}^*$. Then one step of parallel (exact) gradient descent with compression will leave $\mathbf{x}^*$ in general.

Compression Technique: Direct Compression (Q-SGD)

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{\eta}{n}\sum_{i=1}^{n} Q(\nabla f_i(\mathbf{x}^k))$$

- Assume that all nodes have the true solution $\mathbf{x}^*$. Then one step of parallel (exact) gradient descent with compression will leave $\mathbf{x}^*$ in general.
- That is

$$\begin{aligned}\mathbf{x} = \ & \mathbf{x}^* - \frac{\eta}{n}\sum_{i=1}^{n} Q(\nabla f_i(\mathbf{x}^*)) \\ = \ & \mathbf{x}^* - \frac{\eta}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x}^*) + \frac{\eta}{n}\sum_{i=1}^{n}(\nabla f_i(\mathbf{x}^*) - Q(\nabla f_i(\mathbf{x}^*))).\end{aligned}$$

The only ways to have $\mathbf{x} = \mathbf{x}^*$ are that the sum of the compression error is $\mathbf{0}$ or $\eta = 0$.

Compression Technique: Error Compensation (EF-SGD)

$$\mathbf{x}^{k+1} = \mathbf{x}^{k} - \frac{\eta}{n}\sum_{i=1}^{n} Q(\nabla f_i(\mathbf{x}^{k}) + \mathbf{e}_i^{k-1})$$

$$\mathbf{e}_i^{k} = \nabla f_i(\mathbf{x}^{k}) + \mathbf{e}_i^{k-1} - Q(\nabla f_i(\mathbf{x}^{k}) + \mathbf{e}_i^{k-1})$$

- Main idea: add the compression error to the next variable to be compressed [Seide et al. '14, Wu et al. '18, Stich et al. '18, Karimireddy et al. '19]

Compression Technique: Error Compensation (EF-SGD)

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{\eta}{n}\sum_{i=1}^{n} Q(\nabla f_i(\mathbf{x}^k) + \mathbf{e}_i^{k-1})$$
$$\mathbf{e}_i^k = \nabla f_i(\mathbf{x}^k) + \mathbf{e}_i^{k-1} - Q(\nabla f_i(\mathbf{x}^k) + \mathbf{e}_i^{k-1})$$

- Main idea: add the compression error to the next variable to be compressed [Seide et al. '14, Wu et al. '18, Stich et al. '18, Karimireddy et al. '19]
- Assume that all nodes have the true solution $\mathbf{x}^*$. In general, we have.

$$\begin{aligned}\mathbf{x} = & \ \mathbf{x}^* - \frac{\eta}{n}\sum_{i=1}^{n} Q(\nabla f_i(\mathbf{x}^*) + \mathbf{e}_i) \\ = & \ \mathbf{x}^* - \frac{\eta}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x}^*) \\ & + \frac{\eta}{n}\sum_{i=1}^{n}(\nabla f_i(\mathbf{x}^*) + \mathbf{e}_i - Q(\nabla f_i(\mathbf{x}^*) + \mathbf{e}_i)) - \frac{\eta}{n}\sum_{i=1}^{n}\mathbf{e}_i.\end{aligned}$$

The only ways to have $\mathbf{x} = \mathbf{x}^*$ are that the sum of the compression error does not change or $\eta = 0$.

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{\eta}{n}\sum_{i=1}^{n} Q(\nabla f_i(\mathbf{x}^k) + \mathbf{e}_i^{k-1})$$

$$\mathbf{e}_i^k = \nabla f_i(\mathbf{x}^k) + \mathbf{e}_i^{k-1} - Q(\nabla f_i(\mathbf{x}^k) + \mathbf{e}_i^{k-1})$$

- Main idea: add the compression error to the next variable to be compressed [Seide et al. '14, Wu et al. '18, Stich et al. '18, Karimireddy et al. '19]
- Assume that all nodes have the true solution $\mathbf{x}^*$. In general, we have.

$$\begin{aligned}\mathbf{x} = {} & \mathbf{x}^* - \frac{\eta}{n}\sum_{i=1}^{n} Q(\nabla f_i(\mathbf{x}^*) + \mathbf{e}_i) \\ = {} & \mathbf{x}^* - \frac{\eta}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x}^*) \\ & + \frac{\eta}{n}\sum_{i=1}^{n}(\nabla f_i(\mathbf{x}^*) + \mathbf{e}_i - Q(\nabla f_i(\mathbf{x}^*) + \mathbf{e}_i)) - \frac{\eta}{n}\sum_{i=1}^{n}\mathbf{e}_i.\end{aligned}$$

  The only ways to have $\mathbf{x} = \mathbf{x}^*$ are that the sum of the compression error does not change or $\eta = 0$.
- It convergences with a diminishing stepsize also, and it requires the boundedness of the stochastic gradient.

Compression Technique: Difference Compression (DIANA)

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{\eta}{n} \sum_{i=1}^{n} \mathbf{h}_i^k + Q(\nabla f_i(\mathbf{x}^k) - \mathbf{h}_i^k)$$

- Main idea: quantize the difference between the gradient and an estimation [Mishchenko et al. '19, Liu et al. '20]

$$\mathbf{x}^{k+1} = \mathbf{x}^{k} - \frac{\eta}{n}\sum_{i=1}^{n} \mathbf{h}_i^k + Q(\nabla f_i(\mathbf{x}^k) - \mathbf{h}_i^k)$$

- Main idea: quantize the difference between the gradient and an estimation [Mishchenko et al. '19, Liu et al. '20]
- Assume that all nodes have the true solution $\mathbf{x}^*$. In general, we have.

$$\begin{aligned}\mathbf{x} = \; & \mathbf{x}^* - \frac{\eta}{n}\sum_{i=1}^{n} \mathbf{h}_i + Q(\nabla f_i(\mathbf{x}^*) - \mathbf{h}_i) \\ = \; & \mathbf{x}^* - \frac{\eta}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x}^*) + \frac{\eta}{n}\sum_{i=1}^{n}(\nabla f_i(\mathbf{x}^*) - \mathbf{h}_i - Q(\nabla f_i(\mathbf{x}^*) - \mathbf{h}_i))\end{aligned}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{\eta}{n}\sum_{i=1}^{n} \mathbf{h}_i^k + Q(\nabla f_i(\mathbf{x}^k) - \mathbf{h}_i^k)$$

- Main idea: quantize the difference between the gradient and an estimation [Mishchenko et al. '19, Liu et al. '20]
- Assume that all nodes have the true solution $\mathbf{x}^*$. In general, we have.

$$\begin{aligned} \mathbf{x} = &\ \mathbf{x}^* - \frac{\eta}{n}\sum_{i=1}^{n} \mathbf{h}_i + Q(\nabla f_i(\mathbf{x}^*) - \mathbf{h}_i) \\ = &\ \mathbf{x}^* - \frac{\eta}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x}^*) + \frac{\eta}{n}\sum_{i=1}^{n}(\nabla f_i(\mathbf{x}^*) - \mathbf{h}_i - Q(\nabla f_i(\mathbf{x}^*) - \mathbf{h}_i)) \end{aligned}$$

- For a positive η, if we can have $\mathbf{h}_i = \nabla f_i(\mathbf{x}^*)$, then we have $\mathbf{x} = \mathbf{x}^*$.

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{\eta}{n}\sum_{i=1}^{n} \mathbf{h}_i^k + Q(\nabla f_i(\mathbf{x}^k) - \mathbf{h}_i^k)$$

- Main idea: quantize the difference between the gradient and an estimation [Mishchenko et al. '19, Liu et al. '20]
- Assume that all nodes have the true solution $\mathbf{x}^*$. In general, we have.

$$\begin{aligned}\mathbf{x} =& \ \mathbf{x}^* - \frac{\eta}{n}\sum_{i=1}^{n} \mathbf{h}_i + Q(\nabla f_i(\mathbf{x}^*) - \mathbf{h}_i) \\ =& \ \mathbf{x}^* - \frac{\eta}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x}^*) + \frac{\eta}{n}\sum_{i=1}^{n}(\nabla f_i(\mathbf{x}^*) - \mathbf{h}_i - Q(\nabla f_i(\mathbf{x}^*) - \mathbf{h}_i))\end{aligned}$$

- For a positive η, if we can have $\mathbf{h}_i = \nabla f_i(\mathbf{x}^*)$, then we have $\mathbf{x} = \mathbf{x}^*$.
- Question: How to update $\mathbf{h}_i$?

$$\mathbf{h}_i^{k+1} = \mathbf{h}_i^k + \alpha Q(\nabla f_i(\mathbf{x}^k) - \mathbf{h}_i^k)$$

Comparison of Three Compression Techniques

- Direct compression: large variance, works when the variance converges to zero
- Error compensation: smaller variance compared to the direct compression, also works when the variance converges to zero
- Difference compression: the variance converges to zero.

Input: Stepsize $\alpha, \beta, \gamma, \eta$, initialize $\mathbf{h}^0 = \mathbf{h}_i^0 = \mathbf{0}^d$, $\hat{\mathbf{x}}_i^0 = \hat{\mathbf{x}}^0$, $\forall i \in \{1, \ldots, n\}$.
for $k = 1, 2, \cdots, K-1$ **do**

For each worker $\{i = 1, 2, \cdots, n\}$:

Gradient residual: $\Delta_i^k = \nabla f_i(\hat{\mathbf{x}}_i^k) - \mathbf{h}_i^k$
Compression: $\hat{\Delta}_i^k = Q(\Delta_i^k)$
$\mathbf{h}_i^{k+1} = \mathbf{h}_i^k + \alpha \hat{\Delta}_i^k$
Sent $\hat{\Delta}_i^k$ to the master
Receive $\hat{\mathbf{q}}^k$ from the master
$\hat{\mathbf{x}}_i^{k+1} = \hat{\mathbf{x}}_i^k + \beta \hat{\mathbf{q}}^k$

For the master:

Receive $\hat{\Delta}_i^k$s from workers
$\hat{\Delta}^k = 1/n \sum_i^n \hat{\Delta}_i^k$
$\hat{\mathbf{g}}^k = \mathbf{h}^k + \hat{\Delta}^k$
$\mathbf{h}^{k+1} = \mathbf{h}^k + \alpha \hat{\Delta}^k \{= \frac{1}{n} \sum_{i=1}^n \mathbf{h}_i^{k+1}\}$
$\mathbf{q}^k = -\eta \hat{\mathbf{g}}^k + \gamma \mathbf{e}^k$
Compression: $\hat{\mathbf{q}}^k = Q(\mathbf{q}^k)$
$\mathbf{e}^{k+1} = \mathbf{q}^k - \hat{\mathbf{q}}^k$
Broadcast $\hat{\mathbf{q}}^k$ to workers

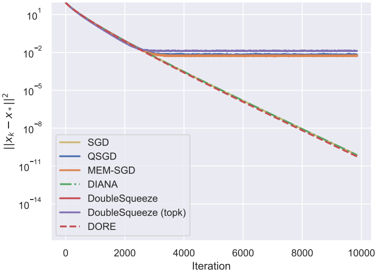
end for
Output: any $\hat{\mathbf{x}}_i^K$

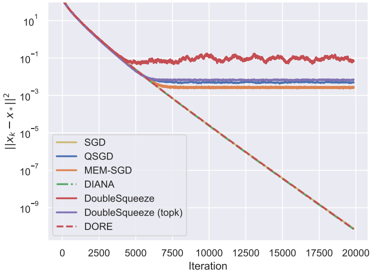
- The worker nodes compress the residual.
- The master node uses error compensation and broadcasts the compressed update (which converges to $\mathbf{0}$).

Algorithm Comparison

Algorithm	Compression	Compress. Model	Linear	Nonconvex Rate
SGD	No	No	✓	$\frac{1}{\sqrt{Kn}} + \frac{1}{K}$
QSGD	Grad	2-norm	N/A	$\frac{1}{K} + B$
MEM-SGD	Grad	k-contraction	N/A	N/A
DIANA	Grad	p-norm	✓	$\frac{1}{\sqrt{Kn}} + \frac{1}{K}$
DoubleSqueeze	Grad+Model	Bdd Variance	N/A	$\frac{1}{\sqrt{Kn}} + \frac{1}{K^{2/3}} + \frac{1}{K}$
DORE	Grad+Model	Assum. 1	✓	$\frac{1}{\sqrt{Kn}} + \frac{1}{K}$

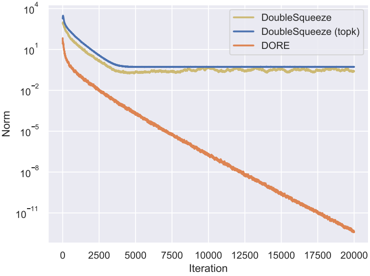
Convergence complexity comparison

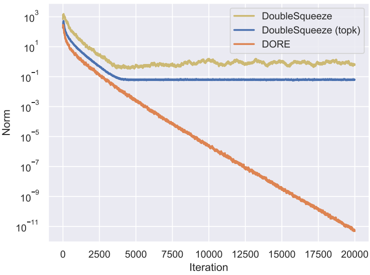




- $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 + \lambda\|\mathbf{x}\|^2$ with $\mathbf{A} \in \mathbb{R}^{1200\times 500}$. The rows of $\mathbf{A}$ are allocated evenly to 20 worker nodes. We take the exact gradient in each node to exclude the effect of the gradient variance (i.e., $\sigma = 0$).
  Left: $\gamma = 0.05$, Right: $\gamma = 0.025$.
- Linear convergence: DORE, SGD, DIANA; Not converge: QSGD, MEM-SGD, DoubleSqueeze (diverges in Left figure), DoubleSqueeze (topk)

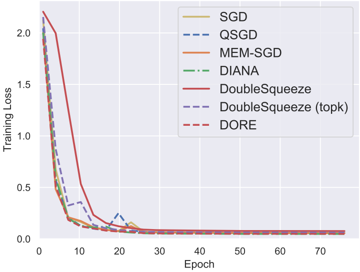
Linear Regression: Data to be Compressed

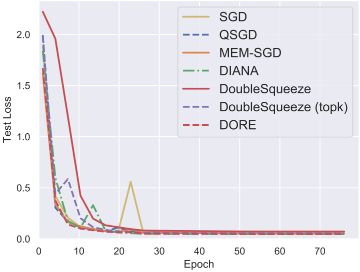




- The norm of the variable to be compressed in all algorithms.
  Left: the worker node; Right: the master node.
- The norm of the variable decreases exponentially for DORE, while that of DoubleSqueeze does not decrease after certain iterations.

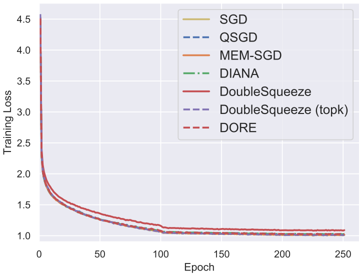
LeNet on MINST

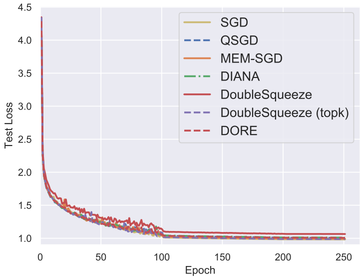




- We use 1 parameter server and 10 worker nodes, each of which is equipped with an NVIDIA Tesla K80 GPU. The batch size for each worker node is 256. Learning rate is 0.1 and decreases by a factor of 0.1 after every 25 epochs.

Resnet18 Trained on CIFAR10





- We use 1 parameter server and 10 worker nodes, each of which is equipped with an NVIDIA Tesla K80 GPU. The batch size for each worker node is 256. Learning rate is 0.01 and decreases by a factor of 0.1 after every 100 epochs.

Stochastic Algorithms

- Stochastic gradient descent: replace $\nabla f_i(\mathbf{x}^k)$ by a stochastic approximation $\mathbf{g}_i^k$. It is unbiased $\mathbb{E}\mathbf{g}_i^k = \nabla f_i(\mathbf{x}^k)$ and has positive variance.
- Variance reduction techniques: SVRG, SAGA, SARAH.
- Momentum method: momentum SGD, STORM, ROOT-SGD, IGT.

- For stochastic gradient without variance reduction, error compensation may be good enough and the compression error will not converge to zero.
- However, for stochastic with momentum, error compensation can not remove previous errors.

Momentum SGD

Momentum update

$$\mathbf{v}^k = (1-\alpha_k)\mathbf{v}^{k-1} + \alpha_k \mathbf{g}^k, \quad \mathbf{x}^{k+1} = \mathbf{x}^k - \eta \mathbf{v}^k,$$

where $\mathbf{g}^k$ is an estimation of the gradient at $\mathbf{x}^k$.

	α_k	$\mathbf{g}^k$
SGD	1	$\nabla f_i(\mathbf{x}^k; \xi_k)$
Momentum SGD	α	$\nabla f_i(\mathbf{x}^k; \xi_k)$
STORM	α	$\frac{1}{\alpha_k}\left(\nabla f_i(\mathbf{x}^k; \xi_k) - (1-\alpha_k)\nabla f_i(\mathbf{x}^{k-1}; \xi_k)\right)$
ROOT-SGD	$1/k$	$\frac{1}{\alpha_k}\left(\nabla f_i(\mathbf{x}^k; \xi_k) - (1-\alpha_k)\nabla f_i(\mathbf{x}^{k-1}; \xi_k)\right)$
IGT	α	$\nabla f_i\left(\mathbf{x}^k + \frac{1-\alpha_k}{\alpha_k}(\mathbf{x}^k - \mathbf{x}^{k-1}); \xi_k\right)$

Standard error compensation without momentum:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \eta Q(\mathbf{g}^k + \mathbf{e}^{k-1}), \quad \mathbf{e}^k = \mathbf{g}^k + \mathbf{e}^{k-1} - Q(\mathbf{g}^k + \mathbf{e}^{k-1}).$$

Error Accumulation (Let $\alpha_k = \alpha$ and $\mathbf{v}^{-1} = \mathbf{0}$)

No compression:

$$\mathbf{x}^T = \mathbf{x}^0 - \eta \sum_{t=0}^{T-1} \sum_{s=0}^{t} \alpha(1-\alpha)^{t-s} \mathbf{g}^s = \mathbf{x}^0 - \eta \sum_{t=0}^{T-1} \sum_{s=0}^{t-1} \alpha(1-\alpha)^{t-s} \mathbf{g}^s - \alpha\eta \sum_{t=0}^{T-1} \mathbf{g}^t$$

Simple compression on $\mathbf{g}$:

$$\begin{aligned}\mathbf{x}^T &= \mathbf{x}^0 - \eta \sum_{t=0}^{T-1} \sum_{s=0}^{t} \alpha(1-\alpha)^{t-s} (\mathbf{g}^s - \mathbf{e}^s) \\ &= \mathbf{x}^0 - \eta \sum_{t=0}^{T-1} \sum_{s=0}^{t} \alpha(1-\alpha)^{t-s} \mathbf{g}^s + \eta \sum_{s=0}^{T-2} (1 - (1-\alpha)^{T-s}) \mathbf{e}^s + \eta\alpha \mathbf{e}^{T-1}\end{aligned}$$

Error compensation ($\mathbf{e}^{-1} = \mathbf{0}$):

$$\begin{aligned}\mathbf{x}^T &= \mathbf{x}^0 - \eta \sum_{t=0}^{T-1} \sum_{s=0}^{t} \alpha(1-\alpha)^{t-s} (\mathbf{g}^s + \mathbf{e}^{s-1} - \mathbf{e}^s) \\ &= \mathbf{x}^0 - \eta \sum_{t=0}^{T-1} \sum_{s=0}^{t} \alpha(1-\alpha)^{t-s} \mathbf{g}^s + \eta \sum_{s=0}^{T-2} \alpha(1-\alpha)^{T-1-s} \mathbf{e}^s + \eta\alpha \mathbf{e}^{T-1}\end{aligned}$$

When $\alpha = 1$ (no momentum), the error is just $\eta \mathbf{e}^{T-1}$. However, α is usually very small if momentum is applied.

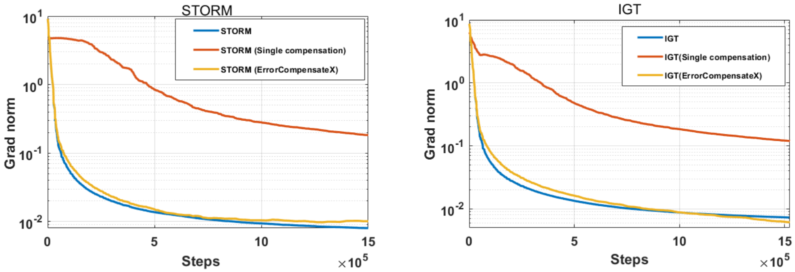
ErrorCompensatedX: Using the Previous Two Errors

ErrorCompensedX ($\mathbf{e}^{-1} = \mathbf{e}^{-2} = \mathbf{0}$):

$$\begin{aligned}\mathbf{x}^T &= \mathbf{x}^0 - \eta \sum_{t=0}^{T-1} \sum_{s=0}^{t} \alpha(1-\alpha)^{t-s}(\mathbf{g}^s + (1-\alpha)(\mathbf{e}^{s-1} - \mathbf{e}^{s-2}) + \mathbf{e}^{s-1} - \mathbf{e}^s) \\ &= \mathbf{x}^0 - \eta \sum_{t=0}^{T-1} \sum_{s=0}^{t} \alpha(1-\alpha)^{t-s}\mathbf{g}^s + \eta\alpha\mathbf{e}^{T-1}\end{aligned}$$

The error does not accumulate!

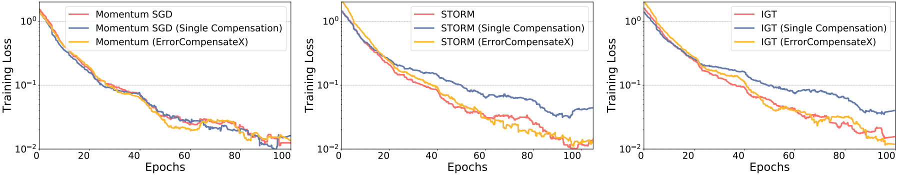
Convergence speed comparison on linear regression for STORM and IGT with different compression techniques.



The y-axis is the norm of the full gradient. The batch size equals 1, $\alpha_k = 1/k$, and we use 1-bit compression.

Epoch-wise convergence comparison on ResNet-50:

Training loss:



Test accuracy:

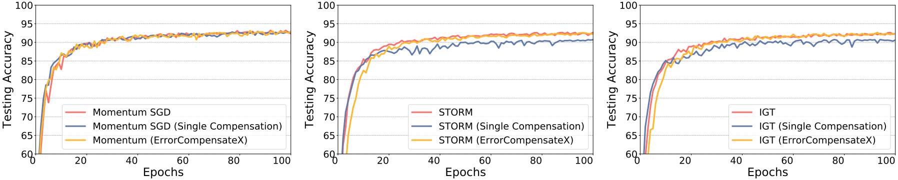
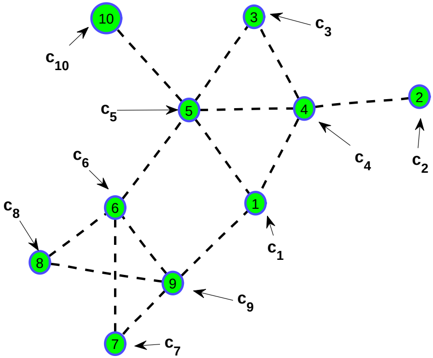


Table of Contents

- Settings:
  - - A connected network with n agents, denoted as $\mathcal{V} = \{1, 2, \cdots, n\}$.
  - - Each agent i holds a local number (vector) c_i privately.
  - - Agent i can exchange the number (vector) with agent j if and only if j is the neighbour of i, denoted as $j \in \mathcal{N}_i$.
- Objective: Each agent obtains the averaged number (vector) $\bar{c} = \frac{1}{n}\sum_{i=1}^{n} c_i$.



- Method: linear iterative averaging (gossip algorithm)
  - Assign each agent i an own model variable x_i.
  - Initialize $x_i^0 = c_i$.
  - Update the variable via

$$x_i^{k+1} = w_{ii} x_i^k + \sum_{j \in \mathcal{N}_i} w_{ij} x_j^k$$

    with given set of weights $\{w_{ij} : i, j \in \mathcal{V}\}$.
  - Examples of weight set:
    - Metropolis weights, i.e., $w_{ij} = \frac{1}{\max\{|\mathcal{N}_i|, |\mathcal{N}_j|\}+1}$ if $j \in \mathcal{N}_i$, $w_{ii} = 1 - \sum_{j \in \mathcal{N}_i} w_{ij}$ and $w_{ij} = 0$ otherwise.
    - Fastest distributed linear averaging in [Xiao-Boyd '04].

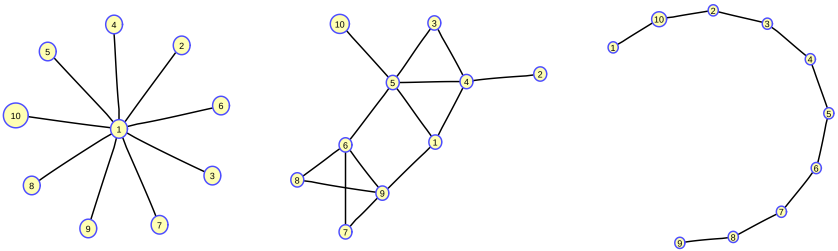


Figure: Star graph, random graph and line graph with 10 agents

- Consider the graph structure of the distributed system, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{E}$ is the set of connection relation among agents in $\mathcal{V}$.
- All agents form an undirected connected graph.
- Edges characterize the connection among agents.

- The problem is formulated as the minimization with constraint

$$\underset{\mathbf{x}_1,\cdots,\mathbf{x}_n\in\mathbb{R}^p}{\text{minimize}} \ \frac{1}{2}\sum_{i=1}^{n}\|\mathbf{x}_i-\mathbf{c}_i\|^2 \quad \text{s.t. } \mathbf{x}_1=\mathbf{x}_2\cdots=\mathbf{x}_n \tag{2}$$

- The linear iterative averaging is formulated as

$$\mathbf{X}^{k+1}=\mathbf{W}\mathbf{X}^k$$

where $\mathbf{X}=[\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_n]^\top$ and $\mathbf{W}$ is the matrix with entries in $\{w_{ij}: i,j\in\mathcal{V}\}$

- By construction, $\mathbf{W}$ is symmetric and doubly stochastic, i.e., $\mathbf{W1}=\mathbf{1}$ and $\mathbf{1}^\top\mathbf{W}=\mathbf{1}^\top$.
- Let $\mathbf{X}^\infty=\bar{\mathbf{c}}\mathbf{1}$ and $\Pi=\frac{1}{n}\mathbf{1}\mathbf{1}^\top$ (averaging matrix)
  - $\|\mathbf{X}^{k+1}-\mathbf{X}^\infty\|^2\le\lambda_{\max}^2(\mathbf{W}-\Pi)\|\mathbf{X}^k-\mathbf{X}^\infty\|^2$
  - Each $\mathbf{x}_i^k$ converges to $\bar{\mathbf{c}}$ linearly at the rate of $\underbrace{\lambda_{\max}^2(\mathbf{W}-\Pi)}_{=\max\{|\lambda_2(\mathbf{W})|,|\lambda_n(\mathbf{W})|\}}$ if $\|\mathbf{W}-\Pi\|_2<1$.

- $\mathbf{W}$ – mixing matrix (gossip matrix) defined over the undirected network $\mathcal{G}$:
  - symmetric and $\mathbf{W1} = \mathbf{1}$.
  - $\|\mathbf{W} - \Pi\|_2 < 1 \quad \Rightarrow \quad -1 < \lambda_n(\mathbf{W}) \leq \cdots \leq \lambda_2(\mathbf{W}) < \lambda_1(\mathbf{W}) = 1$.
- Decentralized consensus problem for averaging:

$$\underset{\mathbf{X}\in\mathbb{R}^{n\times p}}{\text{minimize}} \ \mathbf{f}(\mathbf{X}) := \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{c}_i\|^2 \quad \text{s.t. } (\mathbf{I} - \mathbf{W})\mathbf{X} = \mathbf{0}.$$

- Consensus: $\mathbf{WX} = \mathbf{X}$ iff $\mathbf{x}_1 = \mathbf{x}_2 = \cdots = \mathbf{x}_n$.
- The general decentralized consensus problem (DCP):

$$\underset{\mathbf{X}\in\mathbb{R}^{n\times p}}{\text{minimize}} \ \mathbf{f}(\mathbf{X}) := \sum_{i=1}^{n} f_i(\mathbf{x}_i) \quad \text{s.t. } (\mathbf{I} - \mathbf{W})\mathbf{X} = \mathbf{0}$$

where $f_i(\mathbf{x}_i)$ is a differentiable convex function.

- The general decentralized consensus composite problem (DCCP):

$$\underset{\mathbf{X}\in\mathbb{R}^{n\times p}}{\text{minimize}}\ \mathbf{f}(\mathbf{X})+\mathbf{r}(\mathbf{X}) := \sum_{i=1}^{n} f_i(\mathbf{x}_i)+\sum_{i=1}^{n} r_i(\mathbf{x}_i) \quad \text{s.t. } (\mathbf{I}-\mathbf{W})\mathbf{X}=\mathbf{0},$$

  where $r_i(\mathbf{x})$ is a convex (possibly) non-smooth regularizer.
- Consider the setting:
  - $\mathbf{f}$ is L-smooth and μ-strongly convex, i.e.,

$$\frac{\mu}{2}\|\mathbf{a}-\mathbf{b}\|^2 \leq f_i(\mathbf{a})-f_i(\mathbf{b})-\langle\nabla f_i(\mathbf{b}),\mathbf{a}-\mathbf{b}\rangle \leq \frac{L}{2}\|\mathbf{a}-\mathbf{b}\|^2, \quad \forall \mathbf{a},\mathbf{b}\in\mathbb{R}^p.$$

  - $r_i(\mathbf{x})=r_j(\mathbf{x})$ for $i,j\in\mathcal{V}$ (shared regularizer)
  - The proximal gradient mapping,

$$\mathbf{prox}_{\eta r_i}(\mathbf{x})=\underset{\mathbf{y}\in\mathbb{R}^p}{\arg\min}\ r_i(\mathbf{y})+\frac{1}{2\eta}\|\mathbf{y}-\mathbf{x}\|^2.$$

- Decentralized Gradient Descent (DGD) in [Nedic-Ozdaglar '09]
  - aims to solve DCP.
  - combine mixing step (communication procedure) with gradient descent.

$$\mathbf{X}^{k+1} = \mathbf{W}\mathbf{X}^k - \eta \nabla \mathbf{f}(\mathbf{X}^k)$$

- In agent i's perspective,

$$\left(\mathbf{x}_i^{k+1} = \underbrace{w_{ii}\mathbf{x}_i^k + \sum_{j \in \mathcal{N}_i} w_{ij}\mathbf{x}_j^k}_{\text{mixing step}} - \underbrace{\eta \nabla f_i(\mathbf{x}_i^k)}_{\text{gradient descent}} \right)$$

- Taking fixed stepsize $\eta \in (0, \min\{\frac{1+\lambda_n(\mathbf{W})}{L}, \frac{1}{L+\mu}\}]$ in [Yuan et al. '16],

$$\|\mathbf{X}^k - \mathbf{X}^*\| \leq \mathcal{O}(\rho^k) + \mathcal{O}(\eta).$$

  - "Near" convergence: linear rate to the neighbourhood of $\mathbf{x}^*$.
  - Diminishing stepsize for exact convergence.

- One explanatory view:
  - Reformulate the iteration of DGD as

$$\mathbf{X}^{k+1} = \mathbf{X}^k - \underbrace{[(\mathbf{I} - \mathbf{W})\mathbf{X}^k + \eta\nabla \mathbf{f}(\mathbf{X}^k)]}_{\text{gradient descent}}$$

  - Gradient descent with stepsize 1 for the different problem

$$\underset{\mathbf{X}\in\mathbb{R}^{p\times d}}{\text{minimize}} \ \frac{1}{2}\|\sqrt{\mathbf{I}-\mathbf{W}}\mathbf{X}\|^2 + \eta\mathbf{f}(\mathbf{X})$$

  - The solution $\mathbf{X}^\dagger$ to this problem is generally non-consensus, i.e.,

$$\mathbf{W}\mathbf{X}^\dagger = \mathbf{X}^\dagger + \eta\nabla\mathbf{f}(\mathbf{X}^\dagger) \neq \mathbf{X}^\dagger.$$

  - $\eta \to 0$, i.e., decreasing η in iterations, guarantees the consensus.

Decentralized Gradient Descent

- Another explanatory view:
  - Recall DGD:
$$\mathbf{X}^{k+1} = \mathbf{W}\mathbf{X}^k - \eta\nabla\mathbf{f}(\mathbf{X}^k)$$
  - The limit point $\mathbf{X}^\infty$:
$$(\mathbf{I} - \mathbf{W})\mathbf{X}^\infty = -\eta\nabla\mathbf{f}(\mathbf{X}^\infty)$$
  - The consensus of $\mathbf{X}^\infty \quad \Leftrightarrow \quad \nabla\mathbf{f}(\mathbf{X}^\infty) = \mathbf{0}$.
  - In general, $\nabla\mathbf{f}(\mathbf{X}^*) \neq \mathbf{0}$. Instead,
$$\frac{1}{n}\mathbf{1}^\top\nabla\mathbf{f}(\mathbf{X}^*) = \frac{1}{n}\sum_{i=1}^{n}\nabla f_i(\mathbf{x}_i^*) = 0$$
- If we replace $\nabla\mathbf{f}(\mathbf{X}^k)$ by $\mathbf{Y}^k$ and $\mathbf{Y}^k \to \mathbf{0}$,
$$(\mathbf{I} - \mathbf{W})\mathbf{X}^\infty = -\eta\mathbf{Y}^\infty = \mathbf{0},$$

  we can achieve the consensus.

Gradient Tracking Method

- One form of tracking method to tackle DCP

$$\textbf{Aug-DGM:} \left\lfloor \begin{array}{l} \mathbf{X}^{k+1} = \mathbf{W}\mathbf{X}^{k} - \eta \mathbf{Y}^{k} \\ \mathbf{Y}^{k+1} = \mathbf{W}\mathbf{Y}^{k} + \nabla \mathbf{f}(\mathbf{X}^{k+1}) - \nabla \mathbf{f}(\mathbf{X}^{k}) \end{array} \right.$$

  - $\mathbf{Y}$ is the tracking variable.
  - $\mathbf{Y}$ preserve the gradient average

$$\mathbf{Y}^{0} = \nabla \mathbf{f}(\mathbf{X}^{0}) \quad \Rightarrow \quad \frac{1}{n}\mathbf{1}^{\top}\mathbf{Y}^{k} = \frac{1}{n}\mathbf{1}^{\top}\nabla \mathbf{f}(\mathbf{X}^{k})$$

  - The limit point of $\mathbf{Y}$ satisfies

$$(\mathbf{I} - \mathbf{W})\mathbf{Y}^{\infty} = \mathbf{0}$$
$$\mathbf{Y}^{\infty} = \frac{1}{n}\mathbf{1}\mathbf{1}^{\top}\nabla \mathbf{f}(\mathbf{X}^{\infty})$$

  - $\mathbf{X}^{\infty}$ reaches the consensus $\Leftrightarrow$ $\mathbf{Y}^{\infty} = \mathbf{0}$ $\Leftrightarrow$ $\mathbf{X}^{\infty}$ is solution.
- More forms of gradient tracking can be found in Aug-DGM [Xu et al. '15], Next [Di Lorenzo-Scutari '16],DIGing[Qu-Li '17, Nedic et al. '17] and SONATA [Scutari-Sun '19].

- The exact convergence with fixed stepsize

$$\eta \in \left(0, \frac{(1 - \max\{|\lambda_2(\mathbf{W})|, |\lambda_n(\mathbf{W})|\})^2}{(1 + \sqrt{\frac{L}{\mu} + 3})L}\right)$$

  $\mathbf{X}^k$ converges to $\mathbf{X}^*$ linearly.
- Extension: Push-Pull algorithm over directed network [Pu et al. '20].
- Drawback: one more communication per iteration.
- Question:

  1. Linear convergent algorithm with better communication rounds?
  2. Algorithm with larger stepsize?

- EXTRA proposed in [Shi et al. '15] uses one more historical variable to reach consensus.
- Consider two consecutive iterations of DGD with different mixing matrices

$$\left\lfloor \begin{array}{l} \mathbf{X}^{k+2} = \mathbf{W}\mathbf{X}^{k+1} - \eta\nabla\mathbf{f}(\mathbf{X}^{k+1}) \\ \mathbf{X}^{k+1} = \frac{\mathbf{I}+\mathbf{W}}{2}\mathbf{X}^{k} - \eta\nabla\mathbf{f}(\mathbf{X}^{k}) \end{array} \right.$$

- Combine two iterations,

$$\mathbf{X}^{k+2} - \mathbf{X}^{k+1} = \mathbf{W}\mathbf{X}^{k+1} - \frac{\mathbf{I}+\mathbf{W}}{2}\mathbf{X}^{k} - \eta\nabla\mathbf{f}(\mathbf{X}^{k+1}) + \eta\nabla\mathbf{f}(\mathbf{X}^{k})$$

$$\Downarrow$$

$$\textbf{EXTRA:}\ \ \mathbf{X}^{k+2} = \frac{\mathbf{I}+\mathbf{W}}{2}(2\mathbf{X}^{k+1} - \mathbf{X}^{k}) - \eta\nabla\mathbf{f}(\mathbf{X}^{k+1}) + \eta\nabla\mathbf{f}(\mathbf{X}^{k})$$

EXTRA/NIDS

- Consensus,

$$\mathbf{X}^{\infty} = \frac{\mathbf{I}+\mathbf{W}}{2}\mathbf{X}^{\infty} - \eta\nabla\mathbf{f}(\mathbf{X}^{\infty}) + \eta\nabla\mathbf{f}(\mathbf{X}^{\infty}) \quad \Rightarrow \quad (\mathbf{I}-\mathbf{W})\mathbf{X}^{\infty} = \mathbf{0}.$$

- Optimality, taking telescopic sum,

$$\cancel{\mathbf{X}^1} = \mathbf{W}\mathbf{X}^0 - \cancel{\eta\nabla\mathbf{f}(\mathbf{X}^0)}$$

$$\mathbf{X}^2 - \cancel{\mathbf{X}^1} = \mathbf{W}\mathbf{X}^1 - \frac{\mathbf{I}+\mathbf{W}}{2}\mathbf{X}^0 - \cancel{\eta\nabla\mathbf{f}(\mathbf{X}^1)} + \cancel{\eta\nabla\mathbf{f}(\mathbf{X}^0)}$$

$$\vdots$$

$$\mathbf{X}^{k+1} - \cancel{\mathbf{X}^k} = \mathbf{W}\mathbf{X}^k - \frac{\mathbf{I}+\mathbf{W}}{2}\mathbf{X}^{k-1} - \eta\nabla\mathbf{f}(\mathbf{X}^k) + \cancel{\eta\nabla\mathbf{f}(\mathbf{X}^{k-1})}$$

$$\Downarrow$$

$$\mathbf{X}^{k+1} = \underbrace{\mathbf{W}\mathbf{X}^k - \eta\nabla\mathbf{f}(\mathbf{X}^k)}_{\text{DGD}} - \underbrace{\sum_{t=0}^{k-1}\frac{\mathbf{I}-\mathbf{W}}{2}\mathbf{X}^t}_{\text{correction}}$$

- The consensus of $\mathbf{X}^\infty$ implies the optimality

$$\cancel{\mathbf{X}^\infty} = \cancel{\mathbf{W}\mathbf{X}^\infty} - \eta\nabla\mathbf{f}(\mathbf{X}^\infty) - \sum_{t=0}^{\infty}\frac{\mathbf{I}-\mathbf{W}}{2}\mathbf{X}^t.$$

$$\Downarrow$$

$$\eta\mathbf{1}^\top\nabla\mathbf{f}(\mathbf{X}^\infty) = \eta\sum_{i=1}^{n}\nabla f_i(\mathbf{x}_i^\infty) = -\mathbf{1}^\top\sum_{t=0}^{\infty}\frac{\mathbf{I}-\mathbf{W}}{2}\mathbf{X}^t = \mathbf{0}.$$

- The exact linear convergence with fixed stepsize

$$\eta \in \left(0, \frac{1+\lambda_n(\mathbf{W})}{L}\right).$$

- Extension: larger stepsize over relaxed mixing matrix [Li-Yan '21], i.e.,

$$-\frac{5}{3} < \lambda_n(\mathbf{W}) \leq \cdots \leq \lambda_2(\mathbf{W}) < \lambda_1(\mathbf{W}) = 1,$$

the linear convergence is preserved when $\lambda_n(\mathbf{W}) \leq -1$

$$\eta \in \left(0, \frac{5+3\lambda_n(\mathbf{W})}{4L}\right).$$

- NIDS is proposed in [Li et al. '19].

$$\textbf{NIDS:}\quad \mathbf{X}^{k+2} = \frac{\mathbf{I}+\mathbf{W}}{2}[2\mathbf{X}^{k+1} - \mathbf{X}^{k} - \eta\nabla\mathbf{f}(\mathbf{X}^{k+1}) + \eta\nabla\mathbf{f}(\mathbf{X}^{k})]$$

$$\textbf{EXTRA:}\quad \mathbf{X}^{k+2} = \frac{\mathbf{I}+\mathbf{W}}{2}(2\mathbf{X}^{k+1} - \mathbf{X}^{k}) - \eta\nabla\mathbf{f}(\mathbf{X}^{k+1}) + \eta\nabla\mathbf{f}(\mathbf{X}^{k})$$

- The only difference is the communication of gradient, but NIDS converges linearly with

$$\eta \in \left(0, \frac{2}{L}\right).$$

- The stepsize is independent on the network.
- The stepsize is consistent with that in gradient descent
- Extension: NIDS also preserves linear convergence over relaxed mixing matrix with the same stepsize.

EXTRA/NIDS

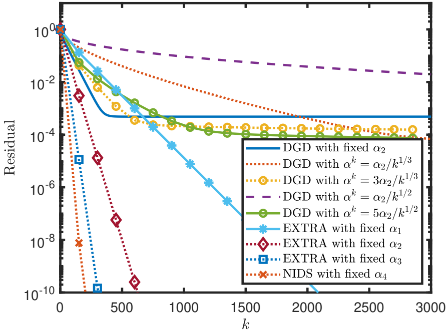
- Compared to gradient tracking methods, EXTRA/NIDS converges faster due to the larger stepsize.
- EXTRA/NIDS communicates only once per iteration.
- EXTRA/NIDS has proximal variant to solve DCCP (DCP+regularizer).

$$\textbf{PG-EXTRA:}\quad \begin{cases} \mathbf{Y}^{k+2} = \mathbf{W}\mathbf{X}^{k+1} + \mathbf{Y}^{k+1} - \frac{\mathbf{I}+\mathbf{W}}{2}\mathbf{X}^{k} - \eta\nabla\mathbf{f}(\mathbf{X}^{k+1}) + \eta\nabla\mathbf{f}(\mathbf{X}^{k}) \\ \mathbf{X}^{k+2} = \mathbf{prox}_{\eta\mathbf{r}}(\mathbf{Y}^{k+2}) \end{cases}$$

$$\textbf{NIDS:}\quad \begin{cases} \mathbf{Y}^{k+2} = \mathbf{W}\mathbf{X}^{k+1} + \mathbf{Y}^{k+1} - \frac{\mathbf{I}+\mathbf{W}}{2}\left[\mathbf{X}^{k} + \eta\nabla\mathbf{f}(\mathbf{X}^{k+1}) - \eta\nabla\mathbf{f}(\mathbf{X}^{k})\right] \\ \mathbf{X}^{k+2} = \mathbf{prox}_{\eta\mathbf{r}}(\mathbf{Y}^{k+2}) \end{cases}$$

- Drawback: it is difficult to adapt and analyze EXTRA/NIDS for directed network.

A Numerical Example



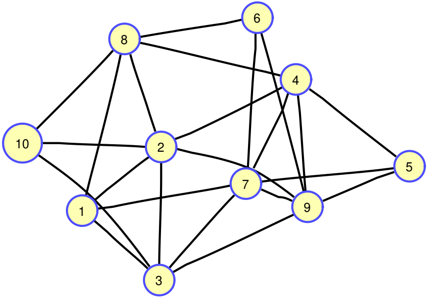
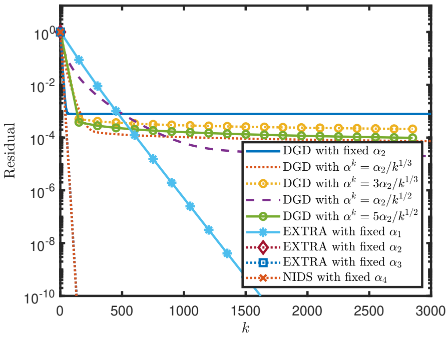


Figure: LEFT: the error $\frac{\|\mathbf{x}^k - \mathbf{x}^*\|_F}{\|\mathbf{x}^0 - \mathbf{x}^*\|_F}$ vs iterations for DGD with different stepsizes, EXTRA with three stepsizes, and NIDS. RIGHT: The random network with 10 nodes. Figure from [Li-Yan '21]



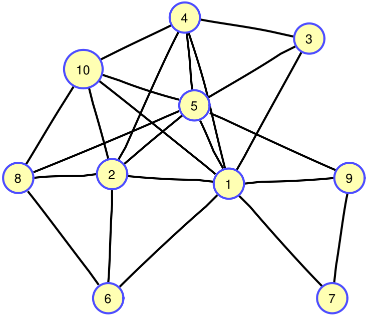


Figure: LEFT: the error $\frac{\|\mathbf{x}^k - \mathbf{x}^*\|_F}{\|\mathbf{x}^0 - \mathbf{x}^*\|_F}$ vs iterations for DGD with different stepsizes, EXTRA with three stepsizes, and NIDS. RIGHT: The random network with 10 nodes. Figure from [Li-Yan '21]

Other Algorithms

- Dual Averaging – [Duchi et al. '11]
- Augmented Lagrangian Method – [Gharesifard-Cortés '13]
- Fast Distributed Gradient Method – [Jakovetić et al. '14]
- D-ADMM – [Shi et al. '14]
- DLM – [Ling et al. '15]
- Stochastic Gradient Push – [Nedić-Olshevsky '16]
- SDCS – [Lan et al. '20]

For more algorithms, refer to survey [Nedić et al. '18]

- PUDA proposed in [Alghunaim et al. '20] to solve DCCP (DCP + non-smooth regularizer)

$$\textbf{PUDA:}\quad \left\lfloor \begin{aligned} \mathbf{Z}^{k+1} &= (\mathbf{I}-\mathbf{C})\mathbf{X}^k - \eta\nabla\mathbf{f}(\mathbf{X}^k) - \mathbf{B}\mathbf{Y}^k \\ \mathbf{Y}^{k+1} &= \mathbf{Y}^k + \mathbf{B}\mathbf{Z}^{k+1} \\ \mathbf{X}^{k+1} &= \mathbf{prox}_{\eta\mathbf{r}}(\bar{\mathbf{A}}\mathbf{Z}^{k+1}) \end{aligned} \right.$$

- $\bar{\mathbf{A}}, \mathbf{B}, \mathbf{C}$ are symmetric matrices dependent on mixing matrix $\mathbf{W}$.
- When $\mathbf{r} = \mathbf{0}$, i.e., no regularizer, the framework covers many algorithms.
  - **Aug-DGM:** $\bar{\mathbf{A}} = \mathbf{W}^2$, $\mathbf{B} = \mathbf{I} - \mathbf{W}$ and $\mathbf{C} = \mathbf{0}$.
  - **DIGing:** $\bar{\mathbf{A}} = \mathbf{I}, \mathbf{B} = \mathbf{I} - \mathbf{W}$ and $\mathbf{C} = \mathbf{I} - \mathbf{W}^2$.
  - **EXTRA:** $\bar{\mathbf{A}} = \mathbf{I}, \mathbf{B} = \frac{\mathbf{I}-\mathbf{W}}{2}$ and $\mathbf{C} = \frac{\mathbf{I}-\mathbf{W}}{2}$.
  - **NIDS:** $\bar{\mathbf{A}} = \frac{\mathbf{I}+\mathbf{W}}{2}, \mathbf{B} = \frac{\mathbf{I}-\mathbf{W}}{2}$ and $\mathbf{C} = \mathbf{0}$.
- PUDA provides (new) proximal variants for these algorithms.

- Assumptions on $\overline{\mathbf{A}}, \mathbf{B}$ and $\mathbf{C}$:
  - (1) $\mathbf{BX} = \mathbf{0} \quad \Leftrightarrow \quad \mathbf{x}_1 = \mathbf{x}_2 = \cdots = \mathbf{x}_n$.
  - (2) $\mathbf{C} = \mathbf{0}$ or $\mathbf{CX} = \mathbf{0} \quad \Leftrightarrow \quad \mathbf{BX} = \mathbf{0}$.
  - (3) $\overline{\mathbf{A}}^2 \preceq \mathbf{I} - \mathbf{B}^2$ and $\mathbf{0} \preceq \mathbf{C} \prec 2\mathbf{I}$.
- Global linear convergence

Theorem (Linear rate with fixed stepsize)

Let $\eta < \frac{2-\lambda_1(\mathbf{C})}{L}$, *it holds that*

$$\|\mathbf{X}^k - \mathbf{X}^*\|^2 + \|\mathbf{Y}^k - \mathbf{Y}^*\|^2 \le \gamma(\|\mathbf{X}^{k-1} - \mathbf{X}^*\|^2 + \|\mathbf{Y}^{k-1} - \mathbf{Y}^*\|^2),$$

where

$$\gamma = \max\{1 - \eta\mu(2 - \lambda_1(\mathbf{C}) - \eta L), 1 - \lambda_{\min}(\mathbf{B}^2)\} < 1.$$

Logistic regression with $\ell_2 + \ell_1$ regularizer.

$$f_i(\mathbf{x}_i) = \frac{1}{L}\sum_{l=1}^{L} \ln(1 + \exp(-y_{i,l}\mathbf{w}_{i,l}^\top \mathbf{x}_i)) + \frac{\lambda}{2}\|\mathbf{x}_i\|^2, \quad r_i(\mathbf{x}_i) = \rho\|\mathbf{x}_i\|_1$$

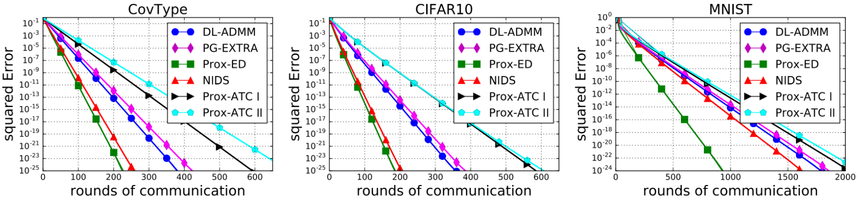
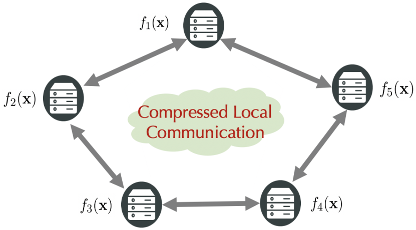


Figure: Simulation results for three datasets. Prox-ED is a new proximal variant of NIDS. Prox-ATC I and II are proximal variants of gradient tracking methods. Figure from [Alghunaim et al. '20]

Table of Contents

Selected algorithms

- DGD-type algorithms:
  - DCD-SGD [Tang et al. '18]
  - Choco-SGD [Koloskova et al. '19]
- Primal-dual algorithms:
  - LEAD [Liu et al. '21]
  - LessBit [Kovalev et al. '21]
  - Prox-LEAD [Li et al. '21]
- Gradient-tracking algorithms:
  - C-GT [Liao et al. '21]



DCD-PSGD: Difference Compression

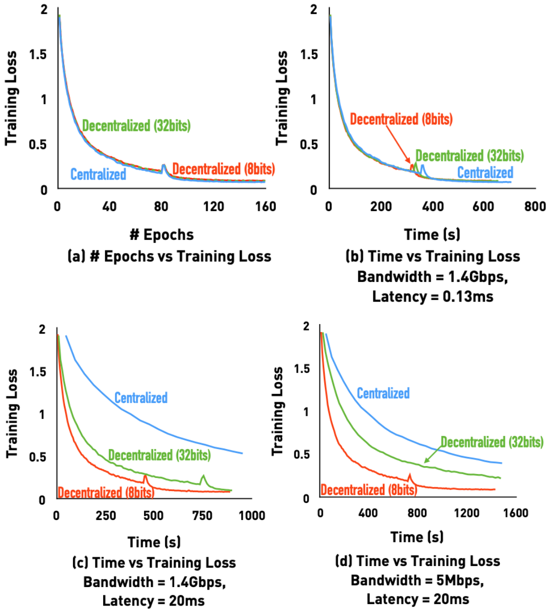
- P-DSGD [Lian et al. '17]: no compression

$$\mathbf{X}^{k+1} = \mathbf{W}\mathbf{X}^{k} - \eta \nabla \mathbf{F}(\mathbf{X}^{k}; \xi^{k})$$

- DCD-PSGD [Tang et al. '18]: difference compression

$$\mathbf{X}^{k+\frac{1}{2}} = \mathbf{W}\mathbf{X}^{k} - \eta \nabla \mathbf{F}(\mathbf{X}^{k}; \xi^{k})$$
$$[\mathbf{W}\mathbf{X}^{k} = \mathbf{W}\mathbf{X}^{k-1} + \mathbf{W}Q(\mathbf{X}^{k-1+\frac{1}{2}} - \mathbf{X}^{k-1})]$$
$$\mathbf{X}^{k+1} = \mathbf{X}^{k} + Q(\mathbf{X}^{k+\frac{1}{2}} - \mathbf{X}^{k})$$

- Convergence: for non-convex and smooth problems: $O(\frac{1}{\sqrt{nK}})$.
- Drawbacks: 1) requires compression with high precision; 2) convergence bias.


Training Loss
Decentralized (32bits)
Centralized
Decentralized (8bits)
Epochs
(a) # Epochs vs Training Loss
Decentralized (8bits)
Decentralized (32bits)
Centralized
Time (s)
(b) Time vs Training Loss
Bandwidth = 1.4Gbps,
Latency = 0.13ms
Centralized
Decentralized (32bits)
Decentralized (8bits)
Time (s)
(c) Time vs Training Loss
Bandwidth = 1.4Gbps,
Latency = 20ms
Centralized
Decentralized (32bits)
Decentralized (8bits)
Time (s)
(d) Time vs Training Loss
Bandwidth = 5Mbps,
Latency = 20ms

Choco-Gossip: Average Preserving

- Gossip (no compression)
  - Decentralized Average (Gossip):
  $$\mathbf{X}^{k+1} = \mathbf{W}\mathbf{X}^k = \mathbf{X}^k - (\mathbf{I} - \mathbf{W})\mathbf{X}^k$$
  - A relaxed form:
  $$\mathbf{X}^{k+1} = \mathbf{X}^k - \gamma(\mathbf{I} - \mathbf{W})\mathbf{X}^k$$
  - Convergence to consensus: $\|\mathbf{X}^k - \mathbf{X}^*\|_F^2 \leq (1 - \gamma\delta)^{2k}\|\mathbf{X}^0 - \mathbf{X}^*\|_F^2$ where $\delta = 1 - |\lambda_2(\mathbf{W})|$.

Choco-Gossip: Average Preserving

- Gossip (no compression)
  - Decentralized Average (Gossip):

$$\mathbf{X}^{k+1} = \mathbf{W}\mathbf{X}^k = \mathbf{X}^k - (\mathbf{I} - \mathbf{W})\mathbf{X}^k$$

  - A relaxed form:

$$\mathbf{X}^{k+1} = \mathbf{X}^k - \gamma(\mathbf{I} - \mathbf{W})\mathbf{X}^k$$

  - Convergence to consensus: $\|\mathbf{X}^k - \mathbf{X}^*\|_F^2 \leq (1 - \gamma\delta)^{2k}\|\mathbf{X}^0 - \mathbf{X}^*\|_F^2$ where $\delta = 1 - |\lambda_2(\mathbf{W})|$.

- Quantized gossip (with compression)
  - Choco-gossip [Koloskova et al. '19]: difference compression

$$\hat{\mathbf{X}}^{k+1} = \hat{\mathbf{X}}^k + Q(\mathbf{X}^k - \hat{\mathbf{X}}^k)$$

$$\mathbf{X}^{k+1} = \mathbf{X}^{k+1} - \gamma(\mathbf{I} - \mathbf{W})\hat{\mathbf{X}}^{k+1}$$

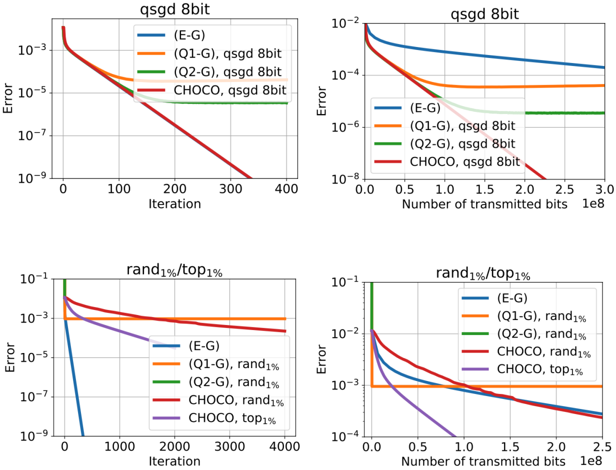
$$[\mathbf{W}\hat{\mathbf{X}}^{k+1} = \mathbf{W}\hat{\mathbf{X}}^k + \mathbf{W}Q(\mathbf{X}^k - \hat{\mathbf{X}}^k)]$$

  - Average preseving:

$$\frac{1}{n}\sum_i \mathbf{X}_i^{k+1} = \frac{1}{n}\sum_i \mathbf{X}_i^k = \bar{\mathbf{X}}^*$$

  - Convergence: $\mathbb{E}\mathbf{E}^k \leq \left(1 - \mathcal{O}(\delta^2(1 - C))\right)^k \mathbb{E}\mathbf{E}^0$ with a special stepsize γ, when Q is a C-contracted compression operator. $\mathbf{E}^k = \|\mathbf{X}^k - \mathbf{X}^*\|_F^2 + \|\mathbf{X}^k - \hat{\mathbf{X}}^{t+1}\|_F^2$

Choco-Gossip: Experiment



Average consensus on the ring topology

- P-DSGD (a slight different form): no compression

$$\mathbf{X}^{k+1} = \mathbf{W}(\mathbf{X}^k - \gamma \nabla \mathbf{F}(\mathbf{X}^k; \xi^k))$$

- Choco-SGD [Koloskova et al. '19]: with compression

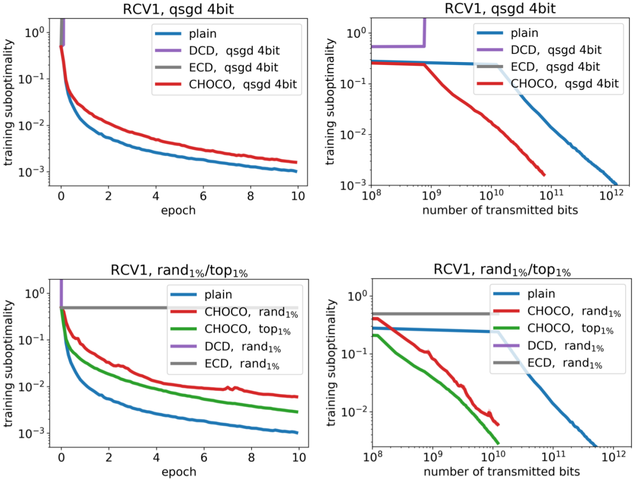
$$\mathbf{X}^{k+\frac{1}{2}} = \mathbf{X}^k - \eta_k \nabla \mathbf{F}(\mathbf{X}^k; \xi^k)$$

$$\hat{\mathbf{X}}^{k+1} = \hat{\mathbf{X}}^{k+1} + Q(\mathbf{X}^{k+\frac{1}{2}} - \hat{\mathbf{X}}^k)$$

$$\mathbf{X}^{k+1} = \mathbf{X}^{k+\frac{1}{2}} - \gamma(\mathbf{I} - \mathbf{W})\hat{\mathbf{X}}^{k+1}$$

$$[\mathbf{W}\hat{\mathbf{X}}^{k+1} = \mathbf{W}\hat{\mathbf{X}}^k + \mathbf{W}Q(\mathbf{X}^k - \hat{\mathbf{X}}^k)]$$

- Convergence: for smooth and μ-strongly convex problems, when K is sufficiently large: $\mathbb{E}f(x_{\text{avg}}^K) - f^* = O(\frac{\sigma^2}{\mu n K})$.
- Drawbacks: 1) hard to tune γ and η_k; 2) slow convergence; 3) convergence bias



Convergence to the optimality on a ring topology

Compression for Primal-Dual Algorithms

- Communication Compression for decentralized optimization
  - DCD-SGD, ECE-SGD [Tang et al. '18]
  - QDGD [Reisizadeh et al. '19a]
  - QuanTimed-DSGD [Reisizadeh et al. '19b]
  - DeepSqueeze [Tang et al. '19]
  - CHOCO-SGD [Koloskova et al. '19]
  - ...
- Reduce to DGD-type algorithms, which suffer from convergence bias

$$\mathbf{X}^* \neq \mathbf{W}\mathbf{X}^* - \eta\nabla\mathbf{F}(\mathbf{X}^*).$$

  The performance degrade on heterogeneous data, and they converge slowly.
- Since primal-dual algorithms are effective in handling the convergence bias, can we design primal-dual algorithms with compression?

LEAD

- LEAD [Liu et al. '21] is the first primal-dual decentralized algorithm with compression that attains linear convergence.
- **Decentralized consensus problem** (DCP)

$$\mathbf{X}^* = \underset{\mathbf{X}\in\mathbb{R}^{n\times p}}{\arg\min} \underbrace{\sum_{i=1}^{n} f_i(\mathbf{x}_i)}_{=:\mathbf{F}(\mathbf{X})}, \quad \text{s.t. } (\mathbf{I}-\mathbf{W})\mathbf{X} = \mathbf{0}, \tag{3}$$

- Consider the equivalent min-max problem

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times p}} \max_{\mathbf{S}\in\mathbb{R}^{n\times p}} \mathbf{F}(\mathbf{X}) + \langle \mathbf{B}^{\frac{1}{2}}\mathbf{X}, \mathbf{S}\rangle, \tag{4}$$

where $\mathbf{B} = \frac{\mathbf{I}-\mathbf{W}}{2}$.

- Consider the equivalent min-max problem

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times p}} \max_{\mathbf{S}\in\mathbb{R}^{n\times p}} \mathbf{F}(\mathbf{X}) + \langle \mathbf{B}^{\frac{1}{2}}\mathbf{X}, \mathbf{S}\rangle, \tag{5}$$

- We apply primal-dual hybrid gradient method (PDHG) in [Zhu-Chan '08]:

$$\left\lfloor \begin{array}{l} \textbf{PDHG}: \\ \mathbf{X}^{k+1} = \mathop{\arg\min}_{\mathbf{X}\in\mathbb{R}^{n\times p}} \mathbf{F}(\mathbf{X}) + \langle \mathbf{B}^{\frac{1}{2}}\mathbf{X}, \mathbf{S}^k\rangle, \\ \mathbf{S}^{k+1} = \mathbf{S}^k + \lambda\mathbf{B}^{\frac{1}{2}}\mathbf{X}^{k+1}. \end{array} \right.$$

- Consider the equivalent min-max problem

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times p}} \max_{\mathbf{S}\in\mathbb{R}^{n\times p}} \mathbf{F}(\mathbf{X}) + \langle \mathbf{B}^{\frac{1}{2}}\mathbf{X}, \mathbf{S}\rangle, \tag{5}$$

- We apply primal-dual hybrid gradient method (PDHG) in [Zhu-Chan '08]:

$$\left\lfloor \begin{array}{l} \textbf{PDHG}: \\ \mathbf{X}^{k+1} = \underset{\mathbf{X}\in\mathbb{R}^{n\times p}}{\arg\min}\ \mathbf{F}(\mathbf{X}) + \langle \mathbf{B}^{\frac{1}{2}}\mathbf{X}, \mathbf{S}^k\rangle, \\ \mathbf{S}^{k+1} = \mathbf{S}^k + \lambda \mathbf{B}^{\frac{1}{2}}\mathbf{X}^{k+1}. \end{array} \right.$$

- We solve **X**-subproblem inexactly by two-step gradient descent with stepsize η:

$$\left\lfloor \begin{array}{l} \textbf{inexact PDHG}: \\ \mathbf{X}^{k+1} = \mathbf{X}^k - \eta\nabla\mathbf{F}(\mathbf{X}^k) - \eta\mathbf{B}^{\frac{1}{2}}\mathbf{S}^k, \\ \mathbf{Y}^{k+1} = \mathbf{X}^{k+1} - \eta\nabla\mathbf{F}(\mathbf{X}^{k+1}) - \eta\mathbf{B}^{\frac{1}{2}}\mathbf{S}^k, \\ \mathbf{S}^{k+1} = \mathbf{S}^k + \lambda \mathbf{B}^{\frac{1}{2}}\mathbf{Y}^{k+1}. \end{array} \right.$$

- To share the computation of $\nabla \mathbf{F}(\mathbf{X}^k)$ between iterations, we switch the order and let $\mathbf{D} = \mathbf{B}^{\frac{1}{2}}\mathbf{S}$:

$$\left| \begin{array}{l} \textbf{inexact PDHG}: \\ \mathbf{Y}^{k+1} = \mathbf{X}^k - \eta \nabla \mathbf{F}(\mathbf{X}^k) - \eta \mathbf{D}^k, \\ \mathbf{D}^{k+1} = \mathbf{D}^k + \frac{\lambda}{2}(\mathbf{I} - \mathbf{W})\mathbf{Y}^{k+1}, \\ \mathbf{X}^{k+1} = \mathbf{X}^k - \eta \nabla \mathbf{F}(\mathbf{X}^k) - \eta \mathbf{D}^{k+1}. \end{array} \right. \tag{6}$$

- There is only one time communication in **D** step.
- We propose a new **compression procedure** for communication over decentralized networks.

- LEAD: set $\lambda = \frac{\gamma}{\eta}$

$$\mathbf{Y}^k = \mathbf{X}^k - \eta \nabla \mathbf{F}(\mathbf{X}^k; \xi^k) - \eta \mathbf{D}^k$$
$$\hat{\mathbf{Y}}^k = \mathit{CompressionProcedure}(\mathbf{Y}^k)$$
$$\mathbf{D}^{k+1} = \mathbf{D}^k + \frac{\gamma}{2\eta}(\mathbf{I} - \mathbf{W})\hat{\mathbf{Y}}^k = \mathbf{D}^k + \frac{\gamma}{2\eta}(\hat{\mathbf{Y}}^k - \hat{\mathbf{Y}}^k_w)$$
$$\mathbf{X}^{k+1} = \mathbf{X}^k - \eta \nabla \mathbf{F}(\mathbf{X}^k; \xi^k) - \eta \mathbf{D}^{k+1}$$

- Compression procedure

$$\mathbf{Q}^k = \text{Compress}(\mathbf{Y}^k - \mathbf{H}^k) \quad \triangleright \ \mathit{Compression}$$
$$\hat{\mathbf{Y}}^k = \mathbf{H}^k + \mathbf{Q}^k$$
$$\hat{\mathbf{Y}}^k_w = \mathbf{H}^k_w + \mathbf{W}\mathbf{Q}^k \quad \triangleright \ \mathit{Communication}$$
$$\mathbf{H}^{k+1} = (1 - \alpha)\mathbf{H}^k + \alpha \hat{\mathbf{Y}}^k$$
$$\mathbf{H}^{k+1}_w = (1 - \alpha)\mathbf{H}^k_w + \alpha \hat{\mathbf{Y}}^k_w$$

How LEAD works?

- Gradient Correction

$$\mathbf{X}^{k+1} = \mathbf{X}^k - \eta(\nabla \mathbf{F}(\mathbf{X}^k; \xi^k) + \mathbf{D}^{k+1})$$

$$\mathbf{F}(\mathbf{X}^k; \xi^k) + \mathbf{D}^{k+1} \to \mathbf{0}$$

- Difference Compression

$$\mathbf{Q}^k = \text{Compress}(\mathbf{Y}^k - \mathbf{H}^k)$$

$$\mathbf{Y}^k \to \mathbf{X}^*, \mathbf{H}^k \to \mathbf{X}^* \Rightarrow \mathbf{Y}^k - \mathbf{H}^k \to \mathbf{0} \Rightarrow \|\mathbf{Q}^k - (\mathbf{Y}^k - \mathbf{H}^k)\| \to 0$$

How LEAD works?

- Gradient Correction

$$\mathbf{X}^{k+1} = \mathbf{X}^k - \eta(\nabla \mathbf{F}(\mathbf{X}^k; \xi^k) + \mathbf{D}^{k+1})$$

$$\mathbf{F}(\mathbf{X}^k; \xi^k) + \mathbf{D}^{k+1} \to \mathbf{0}$$

- Difference Compression

$$\mathbf{Q}^k = \text{Compress}(\mathbf{Y}^k - \mathbf{H}^k)$$

$$\mathbf{Y}^k \to \mathbf{X}^*, \mathbf{H}^k \to \mathbf{X}^* \Rightarrow \mathbf{Y}^k - \mathbf{H}^k \to \mathbf{0} \Rightarrow \|\mathbf{Q}^k - (\mathbf{Y}^k - \mathbf{H}^k)\| \to 0$$

- Implicit Error Compensation

$$\mathbf{E}^k = \hat{\mathbf{Y}}^k - \mathbf{Y}^k$$

$$\mathbf{D}^{k+1} = \mathbf{D}^k + \frac{\gamma}{2\eta}(\hat{\mathbf{Y}}^k - \hat{\mathbf{Y}}^k_w) = \mathbf{D}^k + \frac{\gamma}{2\eta}(\mathbf{I} - \mathbf{W})\mathbf{Y}^k + \frac{\gamma}{2\eta}(\mathbf{E}^k - \mathbf{W}\mathbf{E}^k)$$

- A global average view

$$\overline{\mathbf{X}}^{k+1} = \overline{\mathbf{X}}^k - \eta \overline{\nabla} \mathbf{F}(\mathbf{X}^k; \xi^k)$$

$$\mathbf{X}^k \to \overline{\mathbf{X}}^k$$

- Advantages: 1) faster convergence; 2) support heterogeneous data well; 2) easy to tune stepsizes η, α and γ (simply setting $\alpha = 0.5$ and $\gamma = 1$ works well).

LEAD: Convergence

$$\kappa_f = \frac{L}{\mu}, \quad \kappa_g = \frac{\lambda_{\max}(\mathbf{I} - \mathbf{W})}{\lambda_{\min}^{+}(\mathbf{I} - \mathbf{W})}$$

- Complexity bounds when $\sigma = 0$
  - LEAD converges to the ϵ-accurate solution with the iteration complexity

$$\mathcal{O}\left(\left((1 + C)(\kappa_f + \kappa_g) + C\kappa_f\kappa_g\right) \log \frac{1}{\epsilon}\right).$$

LEAD: Convergence

$$\kappa_f = \frac{L}{\mu}, \quad \kappa_g = \frac{\lambda_{\max}(\mathbf{I}-\mathbf{W})}{\lambda^{+}_{\min}(\mathbf{I}-\mathbf{W})}$$

- Complexity bounds when $\sigma = 0$
  - LEAD converges to the ϵ-accurate solution with the iteration complexity

$$\mathcal{O}\left(\left((1+C)(\kappa_f+\kappa_g)+C\kappa_f\kappa_g\right)\log\frac{1}{\epsilon}\right).$$

  - When $C = 0$ (i.e., no compression) or $C \leq \frac{\kappa_f+\kappa_g}{\kappa_f\kappa_g+\kappa_f+\kappa_g}$, the iteration complexity is

$$\mathcal{O}\left((\kappa_f+\kappa_g)\log\frac{1}{\epsilon}\right).$$

  This recovers the convergence rate of **NIDS** [Li et al. '19].

LEAD: Convergence

$$\kappa_f = \frac{L}{\mu}, \quad \kappa_g = \frac{\lambda_{\max}(\mathbf{I} - \mathbf{W})}{\lambda_{\min}^{+}(\mathbf{I} - \mathbf{W})}$$

- Complexity bounds when $\sigma = 0$
  - LEAD converges to the ϵ-accurate solution with the iteration complexity

$$\mathcal{O}\Big(\big((1 + C)(\kappa_f + \kappa_g) + C\kappa_f\kappa_g\big) \log \frac{1}{\epsilon}\Big).$$

  - When $C = 0$ (i.e., no compression) or $C \leq \frac{\kappa_f + \kappa_g}{\kappa_f\kappa_g + \kappa_f + \kappa_g}$, the iteration complexity is

$$\mathcal{O}\Big(\big(\kappa_f + \kappa_g\big) \log \frac{1}{\epsilon}\Big).$$

    This recovers the convergence rate of **NIDS** [Li et al. '19].
  - With $C = 0$ (or $C \leq \frac{\kappa_f + \kappa_g}{\kappa_f\kappa_g + \kappa_f + \kappa_g}$) and fully connected communication graph (i.e., $\mathbf{W} = \frac{\mathbf{1}\mathbf{1}^\top}{n}$), the iteration complexity is

$$\mathcal{O}(\kappa_f log \frac{1}{\epsilon}).$$

    This recovers the convergence rate of **gradient descent** [Nesterov '13].

$$\kappa_f = \frac{L}{\mu}, \quad \kappa_g = \frac{\lambda_{\max}(\mathbf{I} - \mathbf{W})}{\lambda_{\min}^{+}(\mathbf{I} - \mathbf{W})}$$

- Complexity bounds when $\sigma = 0$
  - LEAD converges to the ϵ-accurate solution with the iteration complexity

$$\mathcal{O}\Big(\big((1 + C)(\kappa_f + \kappa_g) + C\kappa_f\kappa_g\big)\log\frac{1}{\epsilon}\Big).$$

  - When $C = 0$ (i.e., no compression) or $C \leq \frac{\kappa_f+\kappa_g}{\kappa_f\kappa_g+\kappa_f+\kappa_g}$, the iteration complexity is

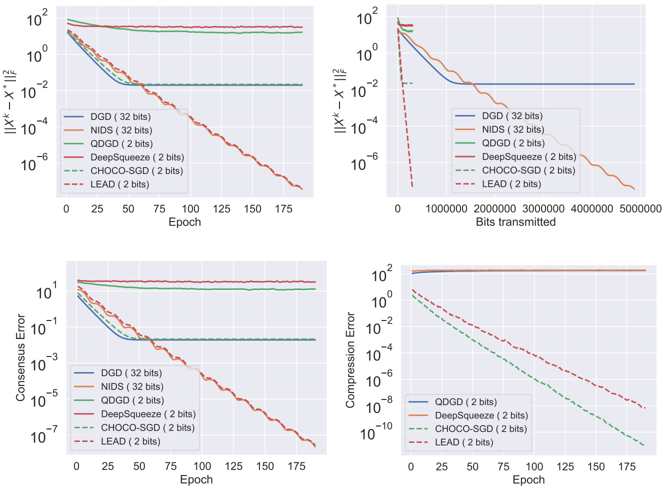
$$\mathcal{O}\Big((\kappa_f + \kappa_g)\log\frac{1}{\epsilon}\Big).$$

    This recovers the convergence rate of **NIDS** [Li et al. '19].
  - With $C = 0$ (or $C \leq \frac{\kappa_f+\kappa_g}{\kappa_f\kappa_g+\kappa_f+\kappa_g}$) and fully connected communication graph (i.e., $\mathbf{W} = \frac{\mathbf{1}\mathbf{1}^\top}{n}$), the iteration complexity is

$$\mathcal{O}(\kappa_f log\frac{1}{\epsilon}).$$

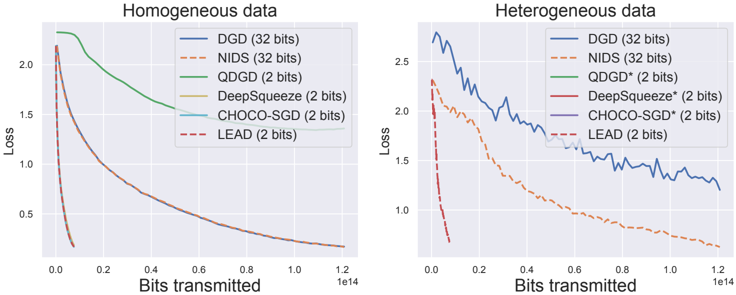
    This recovers the convergence rate of **gradient descent** [Nesterov '13].
- Complexity bounds when $\sigma > 0$
  - Sublinear rate

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left\|\mathbf{x}_i^k - \mathbf{x}^*\right\|^2 \lesssim \mathcal{O}\left(\frac{1}{k}\right)$$



Linear regression (full-gradient)

LEAD: Experiment



Stochastic optimization on deep learning
(AlexNet trained on CIFAR10; $*$ means divergence)

- Consider the equivalent min-max problem

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times p}} \max_{\mathbf{S}\in\mathbb{R}^{n\times p}} \mathbf{F}(\mathbf{X}) + \langle(\mathbf{I}-\mathbf{W})^{\frac{1}{2}}\mathbf{X}, \mathbf{S}\rangle, \tag{7}$$

- Apply one step primal descent and one step dual ascent:

$$\left\lfloor \begin{aligned} \mathbf{X}^{k+1} &= \mathbf{X}^{k} - \eta\nabla\mathbf{F}(\mathbf{X}^{k}) - \eta\mathbf{D}^{k}, \\ \mathbf{D}^{k+1} &= \mathbf{D}^{k} + \theta(\mathbf{I}-\mathbf{W})\mathbf{X}^{k+1}, \end{aligned} \right.$$

where $\mathbf{D}^k = (\mathbf{I}-\mathbf{W})^{\frac{1}{2}}\mathbf{S}^k$. It is a special case of PDGM [Alghunaim-Sayed '20].

- Consider the equivalent min-max problem

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times p}} \max_{\mathbf{S}\in\mathbb{R}^{n\times p}} \mathbf{F}(\mathbf{X}) + \langle (\mathbf{I}-\mathbf{W})^{\frac{1}{2}}\mathbf{X}, \mathbf{S}\rangle, \tag{7}$$

- Apply one step primal descent and one step dual ascent:

$$\left\lfloor \begin{array}{l} \mathbf{X}^{k+1} = \mathbf{X}^k - \eta\nabla\mathbf{F}(\mathbf{X}^k) - \eta\mathbf{D}^k, \\ \mathbf{D}^{k+1} = \mathbf{D}^k + \theta(\mathbf{I}-\mathbf{W})\mathbf{X}^{k+1}, \end{array} \right.$$

where $\mathbf{D}^k = (\mathbf{I}-\mathbf{W})^{\frac{1}{2}}\mathbf{S}^k$. It is a special case of PDGM [Alghunaim-Sayed '20].

- LessBit [Kovalev et al. '21] proposes a similar compression procedure as in LEAD [Liu et al. '21] and apply the compression on $\mathbf{X}^{k+1}$:

$$\left\lfloor \begin{array}{l} \mathbf{X}^{k+1} = \mathbf{X}^k - \eta\nabla\mathbf{F}(\mathbf{X}^k) - \eta\mathbf{D}^k, \\ \hat{\mathbf{X}}^{k+1} = \mathit{CompressionProcedure}(\mathbf{X}^{k+1}) \\ \mathbf{D}^{k+1} = \mathbf{D}^k + \theta(\mathbf{I}-\mathbf{W})\hat{\mathbf{X}}^{k+1} \end{array} \right.$$

- It considers several gradient estimators: Dual gradient/GD/SGD/Loopless SVRG.
- Convergence complexity (full-gradient): $\mathcal{O}((C + \kappa_f\kappa_g + C\kappa_f\widetilde{\kappa_g})\log\frac{1}{\epsilon})$

Prox-LEAD

- Prox-LEAD proposed in [Li et al. '21] considers the decentralized consensus composite problem with regularizer:

$$\mathbf{X}^* = \underset{\mathbf{X}\in\mathbb{R}^{n\times p}}{\arg\min} \underbrace{\sum_{i=1}^{n} f_i(\mathbf{x}_i)}_{=:\mathbf{F}(\mathbf{X})} + \underbrace{\sum_{i=1}^{n} r(\mathbf{x}_i)}_{=:\mathbf{R}(\mathbf{X})}, \quad \text{s.t. } (\mathbf{I}-\mathbf{W})^{\frac{1}{2}}\mathbf{X} = \mathbf{0}, \tag{8}$$

- The equivalent min-max problem:

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times p}} \max_{\mathbf{S}\in\mathbb{R}^{n\times p}} \mathbf{F}(\mathbf{X}) + \langle (\mathbf{I}-\mathbf{W})^{\frac{1}{2}}\mathbf{X}, \mathbf{S}\rangle + \mathbf{R}(\mathbf{X}). \tag{9}$$

- Prox-LEAD proposed in [Li et al. '21] considers the decentralized consensus composite problem with regularizer:

$$\mathbf{X}^* = \underset{\mathbf{X}\in\mathbb{R}^{n\times p}}{\arg\min} \underbrace{\sum_{i=1}^{n} f_i(\mathbf{x}_i)}_{=:\mathbf{F}(\mathbf{X})} + \underbrace{\sum_{i=1}^{n} r(\mathbf{x}_i)}_{=:\mathbf{R}(\mathbf{X})}, \quad \text{s.t. } (\mathbf{I}-\mathbf{W})^{\frac{1}{2}}\mathbf{X} = \mathbf{0}, \tag{8}$$

- The equivalent min-max problem:

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times p}} \max_{\mathbf{S}\in\mathbb{R}^{n\times p}} \mathbf{F}(\mathbf{X}) + \langle(\mathbf{I}-\mathbf{W})^{\frac{1}{2}}\mathbf{X}, \mathbf{S}\rangle + \mathbf{R}(\mathbf{X}). \tag{9}$$

- We adapt the inexact PDHG with an additional proximal gradient step:

$$\left|\begin{aligned} &\mathbf{Y}^{k+1} = \mathbf{X}^k - \eta\nabla\mathbf{F}(\mathbf{X}^k) - \eta\mathbf{D}^k, \\ &\mathbf{D}^{k+1} = \mathbf{D}^k + \frac{\lambda}{2}(\mathbf{I}-\mathbf{W})\mathbf{Y}^{k+1}, \\ &\mathbf{V}^{k+1} = \mathbf{X}^k - \eta\nabla\mathbf{F}(\mathbf{X}^k) - \eta\mathbf{D}^{k+1} = \Big(\mathbf{I} - \frac{\eta\lambda}{2}(\mathbf{I}-\mathbf{W})\Big)\mathbf{Y}^{k+1}, \\ &\mathbf{X}^{k+1} = \mathbf{prox}_{\eta\mathbf{R}}(\mathbf{V}^{k+1}). \end{aligned}\right. \tag{10}$$

where

$$\mathbf{prox}_{\eta\mathbf{R}}(\mathbf{X}) = \underset{\mathbf{Y}\in\mathbb{R}^{n\times p}}{\arg\min}\, \mathbf{R}(\mathbf{Y}) + \frac{1}{2\eta}\|\mathbf{Y}-\mathbf{X}\|^2.$$

- We apply the compression procedure on $\mathbf{Y}^{k+1}$:

$$
\left\lfloor
\begin{aligned}
\mathbf{Y}^{k+1} &= \mathbf{X}^k - \eta \nabla \mathbf{F}(\mathbf{X}^k) - \eta \mathbf{D}^k, \\
\hat{\mathbf{Y}}^{k+1} &= \mathit{CompressionProcedure}(\mathbf{Y}^{k+1}), \\
\mathbf{D}^{k+1} &= \mathbf{D}^k + \frac{\lambda}{2}(\mathbf{I} - \mathbf{W})\hat{\mathbf{Y}}^{k+1}, \\
\mathbf{V}^{k+1} &= \left(\mathbf{I} - \frac{\eta\lambda}{2}(\mathbf{I} - \mathbf{W})\right)\hat{\mathbf{Y}}^{k+1}, \\
\mathbf{X}^{k+1} &= \mathbf{prox}_{\eta \mathbf{R}}(\mathbf{V}^{k+1}).
\end{aligned}
\right.
\tag{11}
$$

- Complexity with full-gradient:

$$\mathcal{O}\left(\left((1+C)(\kappa_f+\kappa_g)+\sqrt{C}(1+C)\kappa_f\kappa_g\right)\log\frac{1}{\epsilon}\right).$$

Algorithm	**R**	**∇F**	**Comp.**	**Convergence complexity**
Dual Gradient Descent	✗	✗	✗	$\widetilde{\mathcal{O}}(\kappa_f\kappa_g)$
LessBit-Option A Kovalev et al. (2021a)	✗	✗	✓	$\widetilde{\mathcal{O}}(C+\kappa_f\kappa_g+C\kappa_f\widetilde{\kappa_g})$
PDGM Alghunaim and Sayed (2020)	✗	✓	✗	$\widetilde{\mathcal{O}}(\kappa_f+\kappa_f\kappa_g)$
LessBit-Option B Kovalev et al. (2021a)	✗	✓	✓	$\widetilde{\mathcal{O}}(C+\kappa_f\kappa_g+C\kappa_f\widetilde{\kappa_g})$
NIDS Li and Yan (2021)	✗	✓	✗	$\widetilde{\mathcal{O}}(\kappa_f+\kappa_g)$
LEAD Liu et al. (2021)	✗	✓	✓	$\widetilde{\mathcal{O}}((1+C)(\kappa_f+\kappa_g)+C\kappa_f\kappa_g)$
PUDA Alghunaim et al. (2020)	✓	✓	✗	$\widetilde{\mathcal{O}}(\kappa_f+\kappa_g)$
Prox-LEAD **this paper, Algorithm 1**	✓	✓	✓	$\widetilde{\mathcal{O}}((1+C)(\kappa_f+\kappa_g)+\sqrt{C}(1+C)\kappa_f\kappa_g)$

Convergence complexity comparison
($\widetilde{\mathcal{O}}$ hides the factor $\log\frac{1}{\epsilon}$)

- Complexity with stochastic-gradient:
  - The general stochastic setting:
$$f_i(\mathbf{x}_i) = \mathbb{E}_{\xi_i \sim \mathcal{D}_i} f_i(\mathbf{x}_i, \xi_i).$$
  - The finite-sum setting:
$$f_i(\mathbf{x}_i) = \frac{1}{m} \sum_{j=1}^{m} f_{ij}(\mathbf{x}_i).$$

The general setting	The finite-sum setting	
	Loopless SVRG	SAGA
Sample $\xi_i \sim \mathcal{D}_i$	Sample $l \in [m] \sim \mathcal{P}_i$ randomly	
$\mathbf{g}_i = \nabla f_i(\mathbf{x}_i, \xi_i)$.	Sample $\omega \in \{0,1\} \sim Bernoulli(p)$, $\mathbf{g}_i = \frac{1}{mp_{il}}(\nabla f_{il}(\mathbf{x}_i) - \nabla f_{il}(\tilde{\mathbf{x}}_i)) + \nabla f_i(\tilde{\mathbf{x}}_i)$, $\tilde{\mathbf{x}}_i = \omega \cdot \mathbf{x}_i + (1-\omega) \cdot \tilde{\mathbf{x}}_i$.	$\mathbf{g}_i = \frac{1}{mp_{il}}(\nabla f_{il}(\mathbf{x}_i) - \nabla f_{il}(\tilde{\mathbf{x}}_{il})) + \frac{1}{m}\sum_{j=1}^{m} \nabla f_{ij}(\tilde{\mathbf{x}}_{ij})$, $\tilde{\mathbf{x}}_{il} = \mathbf{x}_i$.

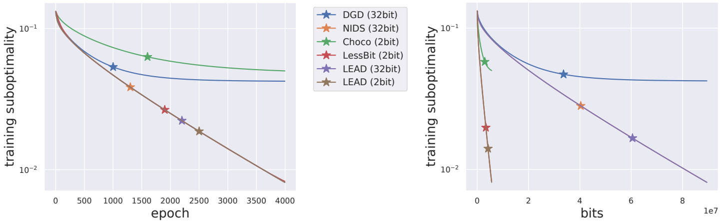
Stochastic gradient oracle (SGO)

Prox-LEAD

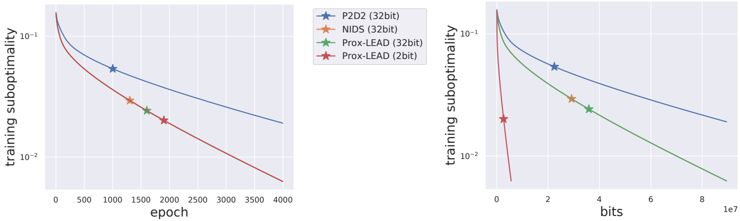
Algorithm	α, η, γ	**Convergence complexity**
Prox-LEAD-GD	fixed	$\widetilde{\mathcal{O}}((1+C)(\kappa_f+\kappa_g)+\sqrt{C}(1+C)\kappa_f\kappa_g)$
Prox-LEAD-SGD	$\mathcal{O}(\frac{1}{k})$	$\mathcal{O}\Big(\big((1+C)^2\kappa_f\kappa_g+\frac{\sigma^2}{L^2}(1+C)^4\kappa_f^2\kappa_g^2\big)\frac{1}{\epsilon}\Big)$
Prox-LEAD-LSVRG	fixed	$\widetilde{\mathcal{O}}((1+C)(\kappa_f+\kappa_g)+\sqrt{C}(1+C)\kappa_f\kappa_g+p^{-1})$
Prox-LEAD-SAGA	fixed	$\widetilde{\mathcal{O}}((1+C)(\kappa_f+\kappa_g)+\sqrt{C}(1+C)\kappa_f\kappa_g+m)$

Summary of the convergence complexities for Prox-LEAD
($\widetilde{\mathcal{O}}$ hides the factor $\log\frac{1}{\epsilon}$)

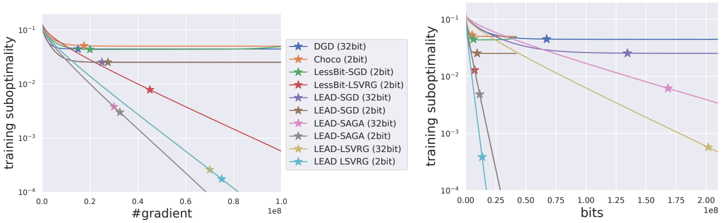
Prox-LEAD: Experiment



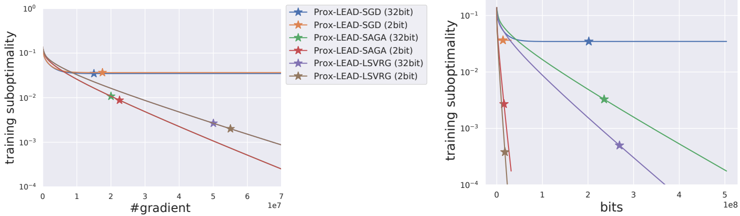
ℓ_2 regularizer with full gradient



$\ell_2 + \ell_1$ regularizer with full gradient



ℓ_2 regularizer with stochastic gradient



$\ell_2 + \ell_1$ regularizer with stochastic gradient

- Gradient tracking (Aug-DGM) [Xu et al. '15]:

$$\mathbf{X}^{k+1} = \mathbf{W}\mathbf{X}^k - \eta\mathbf{Y}^k$$
$$\mathbf{Y}^{k+1} = \mathbf{W}\mathbf{Y}^k + \nabla\mathbf{F}(\mathbf{X}^{k+1}) - \nabla\mathbf{F}(\mathbf{X}^k)$$

- A relaxed form:

$$\mathbf{X}^{k+1} = \mathbf{X}^k - \gamma(\mathbf{I} - \mathbf{W})\mathbf{X}^k - \eta\mathbf{Y}^k$$
$$\mathbf{Y}^{k+1} = \mathbf{Y}^k - \gamma(\mathbf{I} - \mathbf{W})\mathbf{Y}^k + \nabla\mathbf{F}(\mathbf{X}^{k+1}) - \nabla\mathbf{F}(\mathbf{X}^k)$$

Gradient Tracking with Compression

- Gradient tracking (Aug-DGM) [Xu et al. '15]:

$$\mathbf{X}^{k+1} = \mathbf{W}\mathbf{X}^{k} - \eta\mathbf{Y}^{k}$$
$$\mathbf{Y}^{k+1} = \mathbf{W}\mathbf{Y}^{k} + \nabla\mathbf{F}(\mathbf{X}^{k+1}) - \nabla\mathbf{F}(\mathbf{X}^{k})$$

- A relaxed form:

$$\mathbf{X}^{k+1} = \mathbf{X}^{k} - \gamma(\mathbf{I} - \mathbf{W})\mathbf{X}^{k} - \eta\mathbf{Y}^{k}$$
$$\mathbf{Y}^{k+1} = \mathbf{Y}^{k} - \gamma(\mathbf{I} - \mathbf{W})\mathbf{Y}^{k} + \nabla\mathbf{F}(\mathbf{X}^{k+1}) - \nabla\mathbf{F}(\mathbf{X}^{k})$$

- A compressed gradient tracking algorithm (C-GT) [Liao et al. '21]:

$$\hat{\mathbf{X}}^{k} = \mathit{CompressionProcedure}(\mathbf{X}^{k})$$
$$\hat{\mathbf{Y}}^{k} = \mathit{CompressionProcedure}(\mathbf{Y}^{k})$$
$$\mathbf{X}^{k+1} = \mathbf{X}^{k} - \gamma(\mathbf{I} - \mathbf{W})\hat{\mathbf{X}}^{k} - \eta\mathbf{Y}^{k}$$
$$\mathbf{Y}^{k+1} = \mathbf{Y}^{k} - \gamma(\mathbf{I} - \mathbf{W})\hat{\mathbf{Y}}^{k} + \nabla\mathbf{F}(\mathbf{X}^{k+1}) - \nabla\mathbf{F}(\mathbf{X}^{k})$$

- Drawbacks: 1) linear convergence rate is worse than LEAD; 2) it requires double communication cost
- Advantage: it is easier to extend to more general network assumption, such as directed networks (Compressed Push-Pull (CPP) [Song et al. '21]) and dynamic networks.

Table of Contents

Conclusion

- Summary
  Covered by this talk:
  - Communication bottleneck in distributed machine learning
  - Compression operators for communication compression
  - Improved techniques: difference compression & error compensation
  - Centralized algorithms with compression
  - Decentralized algorithms
  - Decentralized algorithms with compression

  Not covered:
  - Asynchronized algorithms
  - Periodic update in local SGD
  - Device sampling in federated learning

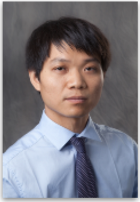
- Future Direction
  - Theoretical analysis under weaker assumptions
  - Acceleration
  - Asynchronization
  - Device sampling
  - Periodic communication
  - Directed and dynamic networks
  - Compressed counterparts

Acknowledgement

Tutorial website: `https://lxiaorui.github.io/distopt/`

Xiaorui Liu

Yao Li



Ming Yan

Jiliang Tang

Thanks for the funding supports from National Science Founding (NSF), Army Research Office (ARO) and Facebook Faculty Research Award.

Sulaiman A Alghunaim and Ali H Sayed.
Linear convergence of primal–dual gradient methods and their performance in distributed optimization.
Automatica, 117:109003, 2020.

Sulaiman A Alghunaim, Ernest K Ryu, Kun Yuan, and Ali H Sayed.
Decentralized proximal gradient algorithms with linear convergence rates.
IEEE Transactions on Automatic Control, 66(6):2787–2794, 2020.

Paolo Di Lorenzo and Gesualdo Scutari.
Next: In-network nonconvex optimization.
IEEE Transactions on Signal and Information Processing over Networks, 2(2): 120–136, 2016.

John C Duchi, Alekh Agarwal, and Martin J Wainwright.
Dual averaging for distributed optimization: Convergence analysis and network scaling.
IEEE Transactions on Automatic control, 57(3):592–606, 2011.

Bahman Gharesifard and Jorge Cortés.
Distributed continuous-time convex optimization on weight-balanced digraphs.
IEEE Transactions on Automatic Control, 59(3):781–786, 2013.

Dušan Jakovetić, Joao Xavier, and José MF Moura.
Fast distributed gradient methods.
IEEE Transactions on Automatic Control, 59(5):1131–1146, 2014.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Urban Stich, and Martin Jaggi.
Error feedback fixes SignSGD and other gradient compression schemes.
In *Proceedings of the 36th International Conference on Machine Learning*, pages 3252–3261. PMLR, 2019.

Anastasia Koloskova, Sebastian U. Stich, and Martin Jaggi.
Decentralized stochastic optimization and gossip algorithms with compressed communication.
In *Proceedings of the 36th International Conference on Machine Learning*, pages 3479–3487. PMLR, 2019.

Dmitry Kovalev, Anastasia Koloskova, Martin Jaggi, Peter Richtarik, and Sebastian Stich.
A linearly convergent algorithm for decentralized optimization: Sending less bits for free!
In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 4087–4095. PMLR, 13–15 Apr 2021.
URL `http://proceedings.mlr.press/v130/kovalev21a.html`.

Guanghui Lan, Soomin Lee, and Yi Zhou.
Communication-efficient algorithms for decentralized and stochastic optimization.
Mathematical Programming, 180(1):237–284, 2020.

Yao Li and Ming Yan.
On the linear convergence of two decentralized algorithms.
Journal of Optimization Theory and Applications, 189(1):271–290, 2021.

Yao Li, Xiaorui Liu, Jiliang Tang, Ming Yan, and Kun Yuan.
Decentralized composite optimization with compression, 2021.

Zhi Li, Wei Shi, and Ming Yan.
A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates.
IEEE Transactions on Signal Processing, 67(17):4494–4506, 2019.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu.
Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent.
In *Advances in Neural Information Processing Systems*, pages 5330–5340, 2017.

Yiwei Liao, Zhuorui Li, Kun Huang, and Shi Pu.
Compressed gradient tracking methods for decentralized optimization with linear convergence, 2021.

Qing Ling, Wei Shi, Gang Wu, and Alejandro Ribeiro.
Dlm: Decentralized linearized alternating direction method of multipliers.
IEEE Transactions on Signal Processing, 63(15):4051–4064, 2015.

Xiaorui Liu, Yao Li, Jiliang Tang, and Ming Yan.

A double residual compression algorithm for efficient distributed learning.
The 23rd International Conference on Artificial Intelligence and Statistics, 2020.

Xiaorui Liu, Yao Li, Rongrong Wang, Jiliang Tang, and Ming Yan.
Linear convergent decentralized optimization with compression.
In *International Conference on Learning Representations*, 2021.
URL https://openreview.net/forum?id=84gjULz1t5.

Konstantin Mishchenko, Eduard Gorbunov, Martin Takáč, and Peter Richtárik.
Distributed learning with compressed gradient differences.
arXiv preprint arXiv:1901.09269, 2019.

Angelia Nedić and Alex Olshevsky.
Stochastic gradient-push for strongly convex functions on time-varying directed graphs.
IEEE Transactions on Automatic Control, 61(12):3936–3947, 2016.

Angelia Nedic and Asuman Ozdaglar.
Distributed subgradient methods for multi-agent optimization.
IEEE Transactions on Automatic Control, 54(1):48–61, 2009.

Angelia Nedic, Alex Olshevsky, and Wei Shi.
Achieving geometric convergence for distributed optimization over time-varying graphs.
SIAM Journal on Optimization, 27(4):2597–2633, 2017.

Angelia Nedić, Alex Olshevsky, and Michael G Rabbat.
Network topology and communication-computation tradeoffs in decentralized optimization.
Proceedings of the IEEE, 106(5):953–976, 2018.

Yurii Nesterov.
Introductory lectures on convex optimization: A basic course, volume 87.
Springer Science & Business Media, 2013.

Shi Pu, Wei Shi, Jinming Xu, and Angelia Nedić.
Push–pull gradient methods for distributed optimization in networks.
IEEE Transactions on Automatic Control, 66(1):1–16, 2020.

Guannan Qu and Na Li.
Harnessing smoothness to accelerate distributed optimization.
IEEE Transactions on Control of Network Systems, 5(3):1245–1260, 2017.

Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, and Ramtin Pedarsani.
An exact quantized decentralized gradient descent algorithm.
IEEE Transactions on Signal Processing, 67(19):4934–4947, 2019a.

Amirhossein Reisizadeh, Hossein Taheri, Aryan Mokhtari, Hamed Hassani, and Ramtin Pedarsani.
Robust and communication-efficient collaborative learning.
In *Advances in Neural Information Processing Systems*, pages 8388–8399, 2019b.

Gesualdo Scutari and Ying Sun.
Distributed nonconvex constrained optimization over time-varying digraphs.
Mathematical Programming, 176(1):497–544, 2019.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu.
1-bit stochastic gradient descent and application to data-parallel distributed training of speech DNNs.
In *Interspeech 2014*, September 2014.

Wei Shi, Qing Ling, Kun Yuan, Gang Wu, and Wotao Yin.
On the linear convergence of the admm in decentralized consensus optimization.
IEEE Transactions on Signal Processing, 62(7):1750–1761, 2014.

Wei Shi, Qing Ling, Gang Wu, and Wotao Yin.
Extra: An exact first-order algorithm for decentralized consensus optimization.
SIAM Journal on Optimization, 25(2):944–966, 2015.

Zhuoqing Song, Lei Shi, Shi Pu, and Ming Yan.
Compressed gradient tracking for decentralized optimization over general directed networks, 2021.

Sebastian U. Stich, Jean-Baptiste Cordonnier, and Martin Jaggi.
Sparsified SGD with memory.
In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 4452–4463, 2018.

Hanlin Tang, Shaoduo Gan, Ce Zhang, Tong Zhang, and Ji Liu.
Communication compression for decentralized training.
In *Advances in Neural Information Processing Systems*, pages 7652–7662. 2018.

Hanlin Tang, Xiangru Lian, Shuang Qiu, Lei Yuan, Ce Zhang, Tong Zhang, and Ji Liu.
Deepsqueeze: Decentralization meets error-compensated compression.
CoRR, abs/1907.07346, 2019.
URL http://arxiv.org/abs/1907.07346.

Jiaxiang Wu, Weidong Huang, Junzhou Huang, and Tong Zhang.
Error compensated quantized SGD and its applications to large-scale distributed optimization.
In *Proceedings of the 35th International Conference on Machine Learning*, pages 5325–5333, 2018.

Lin Xiao and Stephen Boyd.
Fast linear iterations for distributed averaging.
Systems & Control Letters, 53(1):65–78, 2004.

Jinming Xu, Shanying Zhu, Yeng Chai Soh, and Lihua Xie.
Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes.
In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 2055–2060. IEEE, 2015.

Kun Yuan, Qing Ling, and Wotao Yin.
On the convergence of decentralized gradient descent.
SIAM Journal on Optimization, 26(3):1835–1854, 2016.

Mingqiang Zhu and Tony Chan.
An efficient primal-dual hybrid gradient algorithm for total variation image restoration.
UCLA CAM Report, 34:8–34, 2008.