Stochastic Gradient Push for Distributed Deep Learning

Nicolas Loizou

The University of Edinburgh

September 2019: Montreal Institute for Learning Algorithms (MILA).

ICCOPT 2019, Berlin

joint work with Mido Assran, Mike Rabbat, Nicolas Ballas (Facebook AI Research)

(ICML 2019)

Overview

1. Introduction
   - Problem formulation
   - Motivation for Decentralized Topologies
   - Related Work: From Average Consensus to Optimization
2. The Algorithm: Stochastic Gradient Push
   - PushSum Protocol for consensus and SGD
   - Theoretical guarantees: Non-convex functions
3. Numerical Experiments
   - Training ResNet50 on ImageNet Classification task
   - Comparison with state-of-the-art
4. Conclusion & Future Directions of Research

Problem Formulation

Network of n nodes (machines/agents) cooperates to solve the stochastic consensus optimization problem:

$$\min_{x\in\mathbb{R}^d} f(x) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\mathbb{E}_{\xi\sim D_i} F_i(x;\xi)}_{f_i(x)}$$

Distributed Setting

- f is described by **too much data to be stored on a singe computer**
- a **single computer is not powerful enough** for the task at hand and we have access to multiple computers.

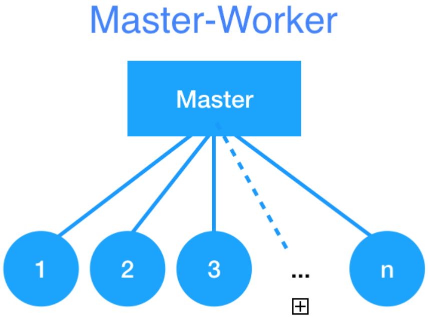
- Node $i \in [n]$ has local data following a distribution D_i.
- Nodes can locally evaluate stochastic gradients $\nabla F_i(\mathbf{x};\xi_i)$, $\xi_i \sim D_i$
- Communication is required to access information from other nodes.

GOAL: Each node evaluates the vector x that minimize $f(x)$.

On Parallel Stochastic Gradient Descent

Workhorse Algorithm: Stochastic Gradient Descent

$$x^{k+1} = x^k - \gamma^k \left(\frac{1}{n} \sum_i^n \nabla F_i(\mathbf{x}; \xi_i) \right)$$



- Leverage parallel computing resources to speed up training.
- Central node aggregates the gradient computed from the other nodes.

Communication Bottleneck

Communication traffic jam on the central node.

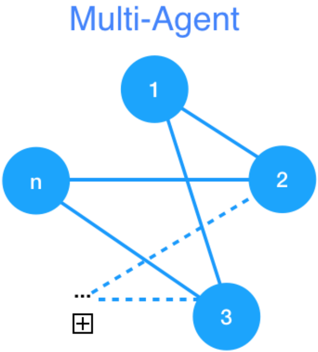


Figure:
$$x_i^{k+1} = x_i^k - \gamma^k \left(\frac{1}{N_i^k} \sum_{j \in N_i^k} \nabla F_j(x_j^k; \xi_i) \right)$$

On decentralized methods

- First developed by control theory/signal processing communities.
- Topology of network is known (application oriented).
- Goal: Design algorithm that converges for this topology.

Difference in Deep learning:

For training DNNs select the network the benefits the training!

Goal: Faster convergence!

Validation accuracy!

On Decentralized Methods

The decentralized optimization methods for machine learning and deep learning tasks based on average consensus methods.

Average Consensus (AC) Problem

Let each node $i \in [n]$ to "know" a private value $c_i \in \mathbb{R}$.
Goal: All nodes compute $\bar{c} := \frac{1}{n}\sum_i c_i$, in a distributed fashion.

Decentralized Protocols:

1. D-PSGD (PushPull parameter aggregation, neighboring nodes)
   AC: [Xiao et al 2005] $\rightarrow$ SC: [Nedic, Ozdaglar 2009]
   $\rightarrow$ DNNs: Lian et al. Neurips 2017. (symmetric communication)

On Decentralized Methods

The decentralized optimization methods for machine learning and deep learning tasks based on average consensus methods.

Average Consensus (AC) Problem

Let each node $i \in [n]$ to "know" a private value $c_i \in \mathbb{R}$.
Goal: All nodes compute $\bar{c} := \frac{1}{n} \sum_i c_i$, in a distributed fashion.

Decentralized Protocols:

1. D-PSGD (PushPull parameter aggregation, neighboring nodes)
   AC: [Xiao et al 2005] $\rightarrow$ SC: [Nedic, Ozdaglar 2009]
   $\rightarrow$ DNNs: Lian et al. Neurips 2017. (symmetric communication)
2. AD-PSGD (PushPull parameter aggregation, pair of nodes)
   AC: [Boyd et al 2006] [Loizou et al. 2019] $\rightarrow$ SC: [Nedic et al. 2010]
   $\rightarrow$ DNNs: Lian et al. ICML 2018. (symmetric communication)

On Decentralized Methods

The decentralized optimization methods for machine learning and deep learning tasks based on average consensus methods.

Average Consensus (AC) Problem

Let each node $i \in [n]$ to "know" a private value $c_i \in \mathbb{R}$.
Goal: All nodes compute $\bar{c} := \frac{1}{n}\sum_i c_i$, in a distributed fashion.

Decentralized Protocols:

1. D-PSGD (PushPull parameter aggregation, neighboring nodes)
   AC: [Xiao et al 2005] $\rightarrow$ SC: [Nedic, Ozdaglar 2009]
   $\rightarrow$ DNNs: Lian et al. Neurips 2017. (symmetric communication)
2. AD-PSGD (PushPull parameter aggregation, pair of nodes)
   AC: [Boyd et al 2006] [Loizou et al. 2019]$\rightarrow$ SC: [Nedic et al. 2010]
   $\rightarrow$ DNNs: Lian et al. ICML 2018. (symmetric communication)
3. Push-Sum (directed, time varying graphs)
   AC: [Kempe et al. 2003] $\rightarrow$ SC: [Nedic, Olshevsky 2016]
   $\rightarrow$ DNNs: This Work

Push-Sum Protocol [Kempe et al. FOCS 2003]

Let $x_i^0 \in \mathbb{R}^d$ be a vector at node i. Goal:Evaluate $\frac{1}{n}\sum_i^n x_i^0$

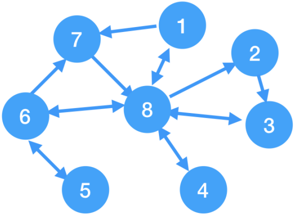


Figure: Directed graph.

Column stochastic matrix $\boldsymbol{P}$:

$$\boldsymbol{P}_{i,j} > 0 \Leftrightarrow (j \to i) \in \mathcal{E}$$
$$\sum_j \boldsymbol{P}_{i,j} = 1$$

Push-Sum Protocol:

1. Initialize $x_i^0 \in \mathbb{R}^d$ and $\omega_i^0 = 1$.
2. Let $\mathbf{X}^0 \in R^{n\times d}$ and $\omega \in R^n$
3. Iterate for $k \geq 0$:
   - $\mathbf{X}^k = \boldsymbol{P}\mathbf{X}^{k-1} = \boldsymbol{P}^k\mathbf{X}^0$
   - $\omega^k = \boldsymbol{P}\omega^{k-1} = \boldsymbol{P}^k\omega^0$
   - $z_i^k = \frac{x_i^k}{\omega_i^k} \to (\frac{1}{n}\sum_i^n x_i^0)$

Thus the update at node i:

$$\boldsymbol{x}_i^{k+1} = \sum_{j=1}^n p_{i,j}^k \boldsymbol{x}_j^k = \sum_{j\in\mathcal{N}_i^{\text{in}(k)}} p_{i,j}^k \boldsymbol{x}_j^k \ .$$

Stochastic Gradient Push (SGP)

Initialize:
$\gamma > 0$, $\boldsymbol{x}_i^{(0)} = \boldsymbol{z}_i^{(0)} \in \mathbb{R}^d$ and $w_i^{(0)} = 1$ for all nodes $i \in \{1, 2, \ldots, n\}$
For iterations $k = 0, 1, 2, \cdots, K$, at node i do:

1. **Local Update:** (SGD/momentum/Adam)
   - Sample new mini-batch $\xi_i^{(k)} \sim \mathcal{D}_i$, compute $\nabla \boldsymbol{F}_i(\boldsymbol{z}_i^{(k)}; \xi_i^{(k)})$
   - Update mini-batch gradient : $\boldsymbol{x}_i^{(k+\frac{1}{2})} = \boldsymbol{x}_i^{(k)} - \gamma \nabla \boldsymbol{F}_i(\boldsymbol{z}_i^{(k)}; \xi_i^{(k)})$
2. **Communication:** (τ-overlap SGP)
   - Send $\left(p_{j,i}^{(k)} \boldsymbol{x}_i^{(k+\frac{1}{2})}, p_{j,i}^{(k)} w_i^{(k)}\right)$ to out-neighbors $j \in \mathcal{N}_i^{\text{out}(k)}$
   - Receive $\left(p_{i,j}^{(k)} \boldsymbol{x}_j^{(k+\frac{1}{2})}, p_{i,j}^{(k)} w_j^{(k)}\right)$ from in-neighbors $j \in \mathcal{N}_i^{\text{in}(k)}$
3. **Aggregate:** (Push-Sum Protocol)
   - $\boldsymbol{x}_i^{(k+1)} = \sum_{j \in \mathcal{N}_i^{\text{in}(k)}} p_{i,j}^{(k)} \boldsymbol{x}_j^{(k+\frac{1}{2})}$
   - $w_i^{(k+1)} = \sum_{j \in \mathcal{N}_i^{\text{in}(k)}} p_{i,j}^{(k)} w_j^{(k)}$
   - $\boldsymbol{z}_i^{(k+1)} = \boldsymbol{x}_i^{(k+1)} / w_i^{(k+1)}$

Algorithm 1 Stochastic Gradient Push (SGP), [Nedić, Olshevsky 2016]

Require: Initialize $\gamma > 0$, $\boldsymbol{x}_i^{(0)} = \boldsymbol{z}_i^{(0)} \in \mathbb{R}^d$ and $w_i^{(0)} = 1$ for all nodes $i \in \{1, 2, \dots, n\}$

for $k = 0, 1, 2, \cdots, K$ **do**

  Sample new mini-batch $\xi_i^{(k)} \sim \mathcal{D}_i$ from local distribution

  Compute a local stochastic mini-batch gradient at $\boldsymbol{z}_i^{(k)}$: $\nabla \boldsymbol{F}_i(\boldsymbol{z}_i^{(k)}; \xi_i^{(k)})$

  $\boldsymbol{x}_i^{(k+\frac{1}{2})} = \boldsymbol{x}_i^{(k)} - \gamma \nabla \boldsymbol{F}_i(\boldsymbol{z}_i^{(k)}; \xi_i^{(k)})$

  Send $\left(p_{j,i}^{(k)} \boldsymbol{x}_i^{(k+\frac{1}{2})}, p_{j,i}^{(k)} w_i^{(k)}\right)$ to out-neighbors $j \in \mathcal{N}_i^{\text{out}(k)}$;
   receive $\left(p_{i,j}^{(k)} \boldsymbol{x}_j^{(k+\frac{1}{2})}, p_{i,j}^{(k)} w_j^{(k)}\right)$ from in-neighbors $j \in \mathcal{N}_i^{\text{in}(k)}$

  $\boldsymbol{x}_i^{(k+1)} = \sum_{j \in \mathcal{N}_i^{\text{in}(k)}} p_{i,j}^{(k)} \boldsymbol{x}_j^{(k+\frac{1}{2})}$

  $w_i^{(k+1)} = \sum_{j \in \mathcal{N}_i^{\text{in}(k)}} p_{i,j}^{(k)} w_j^{(k)}$

  $\boldsymbol{z}_i^{(k+1)} = \boldsymbol{x}_i^{(k+1)} / w_i^{(k+1)}$

end for

The Problem: Main Assumptions

$$\min_{x \in \mathbb{R}^d} f(x) = \frac{1}{n} \sum_{i=1}^{n} \underbrace{\mathbb{E}_{\xi \sim D_i} F_i(x; \xi)}_{f_i(x)}$$

Main Assumptions

- *L-smooth:* $\exists\ L > 0$: $\|\nabla f_i(\boldsymbol{x}) - \nabla f_i(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|$.
- *Bounded variance:* $\mathbb{E}_{\xi \sim D_i} \|\nabla F_i(\boldsymbol{x}; \xi) - \nabla f_i(\boldsymbol{x})\|^2 \leq \sigma^2 \quad \forall i, \forall \boldsymbol{x}$
- *Similar objectives:* $\frac{1}{n} \sum_{i=1}^{n} \|\nabla f_i(\boldsymbol{x}) - \nabla f(\boldsymbol{x})\|^2 \leq \zeta^2 \quad \forall \boldsymbol{x}$.
- *Mixing Connectivity:* $\exists\ B, \Delta > 0$: graph with edge set $\bigcup_{k=lB}^{(l+1)B-1} E^{(k)}$ is strongly connected and has diameter at most Δ for every $l \geq 0$. Here $E^{(k)} = \{(i,j) \colon p_{i,j}^{(k)} > 0\}$.
- *Bounded Delays (Overlap-SGP)*: $\exists\ \tau \in \mathbb{Z}_+$, such that the delay, satisfies $k' - k \leq \tau$

Relation of SGP to other approaches

Parallel SGD (AllReduce gradient aggregation / all nodes)

Topology: fully-connected at every iteration

Uniform mixing weights: That is, $p_{j,i}^{(k)} = 1/n$ for all $i, j = 1, \ldots, n$

In this case the push-sum weight $w_i^k = 1, \quad \forall k, \forall i \in [n]$

D-PSGD

Topology: Static, undirected, and connected at every iteration

Symmetric mixing weights: $p_{j,i}^{(k)} = p_{i,j}^{(k)}$ for all (i,j) : $\sum_j p_{j,i}^{(k)} = 1$;

In this case the push-sum weight $w_i^k = 1, \quad \forall k, \forall i \in [n]$

Directed time-varying graphs

Topology: Directed, potentially time-varying, and B-strongly connected

Nodes choose *mixing weights* $p_{j,i}^{(k)}$ independently of one another;

In this case, push-sum weight w_i^k may differ between nodes at any given iteration, and we obtain a new operating regime.

Main Theoretical Contributions

Theorem (Convergence of $\overline{\boldsymbol{x}}^{(k)}$)

Suppose that main assumptions hold. Run SGP for K iterations with step-size $\gamma = \sqrt{n/K}$. Let $f^* = \inf_{\boldsymbol{x}} f(\boldsymbol{x})$ and assume that $f^* > -\infty$. There exist constants C and $q \in [0, 1)$, which depend on Δ, $\boldsymbol{P}^{(k)}$ and τ such that when:

$$K \geq \max \left\{ \frac{nL^4 C^4 60^2}{(1-q)^4}, \frac{L^4 C^4 P_1^2 n}{(1-q)^4 (f(\overline{\boldsymbol{x}}^{(0)}) - f^* + \frac{L\sigma^2}{2})^2}, \frac{L^2 C^2 n P_2}{(1-q)^2 (f(\overline{\boldsymbol{x}}^{(0)}) - f^* + \frac{L\sigma^2}{2})}, n \right\}$$

then

$$\frac{\sum_{k=0}^{K-1} \mathbb{E} \left\| \nabla f(\overline{\boldsymbol{x}}^{(k)}) \right\|^2}{K} \leq \frac{12(f(\overline{\boldsymbol{x}}^{(0)}) - f^* + \frac{L\sigma^2}{2})}{\sqrt{nK}}.$$

where $\overline{\boldsymbol{x}}^{(k)} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i^{(k)}$ (average of the nodes' parameters).

Remark: Centralized parallel SGD converges also with $O(1/\sqrt{nK})$.

Main Theoretical Contributions

Theorem (Convergence to stationary point)

Under the same assumptions,

$$\frac{1}{nK}\sum_{k=0}^{K-1}\sum_{i=1}^{n}\mathbb{E}\left\|\overline{\boldsymbol{x}}^{(k)}-\boldsymbol{z}_i^{(k)}\right\|^2 \leq O\left(\frac{1}{K}+\frac{1}{K^{3/2}}\right),$$

and

$$\frac{1}{nK}\sum_{k=0}^{K-1}\sum_{i=1}^{n}\mathbb{E}\left\|\nabla f(\boldsymbol{z}_i^k)\right\|^2 \leq O\left(\frac{1}{\sqrt{nK}}+\frac{1}{K}+\frac{1}{K^{3/2}}\right).$$

As K grows:

- Variables $\boldsymbol{z}_i^{(k)} \longrightarrow \overline{\boldsymbol{x}}^{(k)}$,
- Convergence to a stationary point.
- For fixed n and large K, the $1/\sqrt{nK}$ term will dominate the other factors.

Numerical Evaluation

- Training ResNet50 on ImageNet Classification task
- System: 8 GPUs/nodes. Look ar scaling from 4-32 nodes (32-256 GPUs)
- Communication over 10 Gbps Ethernet (high latency scenario) and 100 Gbps Infiniband networks (no communication bottleneck)
- Comparison with State-of-the-art:
  - (i) AllReduced-based SGD [Goyal et al. 2017]
  - (ii) D-PSGD, decentralized push-pull stochastic gradient descent [Lian et al. NIPS 2017]
  - (iii) AD-PSGD , asynchronous decentralized push-pull stochastic gradient descent [Lian et al. ICML 2018]
- **Code:**
  `https://github.com/facebookresearch/stochastic_gradient_push`

Graph Topology: Directed Exponential Graph

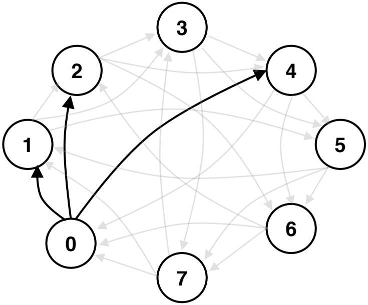
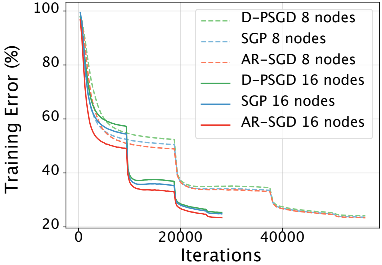
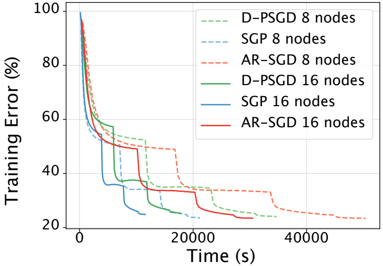


Figure: 8-node directed exponential graph, highlighting node 0's out-neighbours.

- **Cyclic strategy:**
  Each node sends and receives one message per update.
  Node i sends to:
  - $i + 2^0$mod n
  - $i + 2^1$mod n
  - $\ldots$
  - $i + 2^{\log_2(n-1)}$mod n
- Mixing Matrices $\mathbf{P}^k$
  - Node i choose its mixing weights (column i of $\mathbf{P}^k$)
  - Uniform mixing weights.
  - For one-peer-per-node case: Each column of $\mathbf{P}^k$ has exactly two non-zero entries, both equal to $1/2$.

Scaling and Convergence





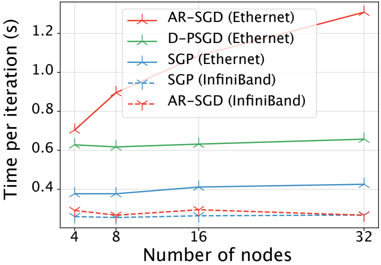
(a) Iteration-wise convergence (b) Time-wise convergence

Figure: Comparison of AllReduce-SGD (AR-SGD), SGP and D-PSGD on 8–16 nodes interconnected via 10 Gbps Ethernet. All methods are run for 90 epochs.

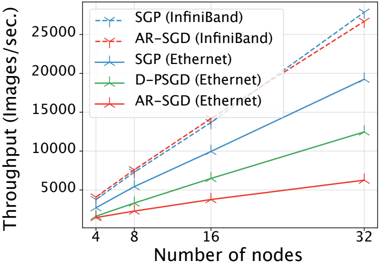
Remarks:

- Increasing $\sharp$ nodes by 2 $\Rightarrow$ $\sharp$ iterations is decreasing by 2.
- SGP completes 90 epochs in less time than other methods.

Scaling and Convergence



(a)



(b)

Figure: Comparison of AllReduce-SGD (AR-SGD), SGP and D-PSGD on 4–32 nodes interconnected via 10 Gbps Ethernet and 100Gbps-InfiniBand.

Remarks:

- InfiniBand: all methods, constant time per iteration
- Ethernet: SGP is the faster method (1.5x faster than D-PSGD)

Scaling and Convergence

	4 nodes (32 GPUs)		8 nodes (64 GPUs)		16 nodes (128 GPUs)		32 nodes (256 GPUs)	
AR-SGD	76.2%	22.0 hrs.	76.4%	14.0 hrs.	76.3%	8.5 hrs.	76.2%	5.1 hrs.
D-PSGD	76.4%	19.7 hrs.	76.1%	9.7 hrs.	75.9%	5.0 hrs.	74.4%	2.6 hrs.
SGP	76.3%	11.8 hrs.	76.4%	5.9 hrs.	75.9%	3.2 hrs.	75.0%	1.7 hrs.

Table: Top-1 validation accuracy (%) and training time (hours), when communicating over 10 Gbps Ethernet for AR-SGD, SGP and D-PSGD. SGP is using 1-peer communication topology. All methods are run for 90 epochs.

- SGP outperforms D-PSGD and AllReduce-SGD in terms of total training time
- Validation accuracy degrades for larger networks (16-32 nodes)

Communication and the speed-accuracy tradeoff.

We explore the effect of communication topology on the speed-accuracy tradeoff (16-31 nodes).

	16 nodes (128 GPUs)		**32 nodes (256 GPUs)**	
AR-SGD	76.3%	8.5 hrs.	76.2%	5.2 hrs.
2P-SGP	76.2%	5.1 hrs.	75.7%	2.5 hrs.
1P-SGP	75.9%	3.2 hrs.	75.0%	1.7 hrs.
AR/1P-SGP	76.2%	4.8 hrs.	75.4%	2.8 hrs.
2P/1P-SGP	76.0%	3.5 hrs.	75.1%	1.8 hrs.

Table: Top-1 validation accuracies (%) and training time (hours) for 1P-SGP (1-peer topology); 2P-SGP (2-peer topology), AR-SGD (AllReduce SGD), AR/1P-SGP (AllReduce first 30 epochs, 1-peer topology last 60 epochs), and 2P/1P-SGP (2-peer topology first 30 epochs, 1-peer topology last 60 epochs), all communicating over 10 Gbps Ethernet.

Overlap SGP (O-SGP)

	Train Acc.	Val. Acc.	Train Time
AR-SGD	76.9%	76.3%	8.5 hrs.
D-PSGD	75.6%	75.9%	4.9 hrs.
AD-PSGD	74.7%	75.5%	2.9 hrs.
SGP	75.6%	75.9%	3.2 hrs.
1-OSGP	77.1%	75.7%	1.8 hrs.

Table: Comparing state-of-the-art synchronous and asynchronous gossip-based approaches to 1-OSGP, an implementation of synchronous SGP where communication is overlapped with 1 gradient step (all messages are always received with 1-iteration of staleness). Experiments are run for 90 epochs over 16 nodes (128 GPUs) interconnected via 10 Gbps Ethernet.

- Overlapping communication and computation: 1)*speeds up training* and 2)*no accuracy degradation*.
- synchronous 1-OSGP is faster than asynchronous AD-PSGD, and achieves better training and validation accuracy.

Fixed runtime budget.

We now compare the methods based on *runtime budget* and not epoch budget.

	Train Acc.	Val. Acc.	Train Time
AR-SGD	76.9%	76.2%	5.1 hrs. (90 epochs)
AD-PSGD	80.3%	76.9%	4.7 hrs. (270 epochs)
SGP	80.0%	77.1%	4.6 hrs. (270 epochs)
1-OSGP	81.8%	77.1%	2.7 hrs. (270 epochs)

Table: Comparing AllReduce SGD (AR-SGD) and SGP under a fix runtime budget. Experiments are run over 1-peer graph topologies, using 32 nodes (256 GPUs) interconnected via 10 Gbps Ethernet.

- Given a similar runtime, SGP outperforms AR-SGD for both training and validation accuracy.
- Running 1-OSGP for the same number of epochs than SGP outperforms SGD while improving the overal training efficiency.

Conclusion

Conclusion

- SGP and O-SGP for accelerating distributed training of DNNs.
- Theoretical convergence guarantees in the smooth non-convex setting, matching known convergence rates for parallel SGD.
- Extensive Numerical Experiments.

Future Directions

Combining techniques for accelerating distributed training of DNNs:

- Compressed messages (Alistarh et al.,2017; Wen et al.,2017; Jia et al.,2018; Koloskova et al.,2019, Tang et al. 2019)
- Truly asynchronous gossip-based variant (Jin et al.,2016 ; Lian et al., 2018)

Extensions on analysis:
Provide analysis for the momentum variant, stage-wise learning rate, remove strong assumptions.

Thank You!