Distributed Training of ML Models: Optimal rates for stochastic non-convex problems

Usman A. Khan
Electrical and Computer Engineering, Tufts University

Machine Learning and Statistical Signal Processing for Data on Graphs
McGill–Bellairs Research Institute, Barbados
January 17, 2023

Acknowledgments


Reza D.


C. Xi


S. Safavi


F. Saadatniaki


R. Xin


M. I. Qureshi


A. Swar

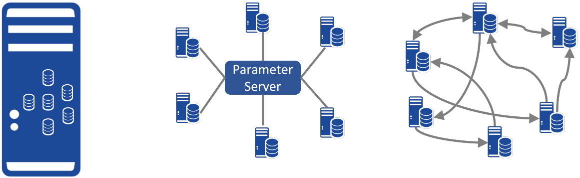

H. Raja

NSF

Motivation and Overview

Learning from Data

- Data science and machine learning hold a lot of potential
    - Image classification, Medical diagnosis, Credit card fraud, . . .
- What architectures do we use to run ML algorithms?
    - Centralized
    - Semi-centralized (or Federated)
    - Completely distributed



- This talk: Distributed peer-to-peer architectures

A simple case study . . .

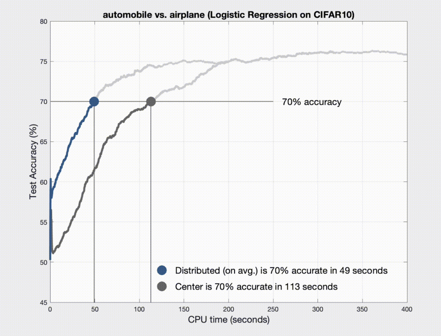


Figure 1: Performance of an ML model trained with 10,000 32×32 pixel images

- Can distributed methods match the centralized accuracy?
- Under what conditions?
- How fast?

Example: Recognizing Traffic Signs

- Identify STOP vs. YIELD sign

Figure 2: Image classification: (Left) Training phase (Right) Testing phase

- Input data: features (e.g., images) $\{\boldsymbol{\theta}_j\}$ and their labels $\{y_j\}$
- Model: A classifier $\mathbf{x}$ that predicts a label $\widehat{y}_j$ for each image $\boldsymbol{\theta}_j$
  - Changing $\mathbf{x}$ changes the predicted label $\widehat{y}_j(\mathbf{x}; \boldsymbol{\theta}_j)$

- Pick a classifier $\mathbf{x}^*$ that minimizes *some* loss ℓ over all images

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^p}{\operatorname{argmin}} \sum_j \ell\Big(y_j, \; \widehat{y}_j(\mathbf{x}; \boldsymbol{\theta}_j)\Big)$$

Example: Recognizing Traffic Signs (*cont...d*)

- Pick a classifier $\mathbf{x}^*$ that minimizes *some* loss over all images when the data is distributed over n machines

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^p}{\operatorname{argmin}} \sum_i \sum_j \ell\Big(y_{ij},\ \widehat{y}_{ij}(\mathbf{x}; \boldsymbol{\theta}_{ij})\Big)$$

$i \in$ machines $\quad j \in$ local dataset

  - Data sharing is not permitted
  - Machine communication has constraints

Minimizing Functions

$$\min_{\mathbf{x}} f(\mathbf{x}), \qquad f := \sum_i \sum_j \ell\Big(y_{ij},\ \widehat{y}_{ij}(\mathbf{x}; \boldsymbol{\theta}_{ij})\Big) : \mathbb{R}^p \rightarrow \mathbb{R}$$

- Different predictors $\widehat{y}$ and losses ℓ lead to different cost functions f

- **Quadratic**: Signal estimation, linear regression, LQR
- **(Strongly) convex**: Logistic regression, SVM
- **Nonconvex**: Neural networks, reinforcement learning, blind sensing
- **Stochastic**: Sampling from mini-batches, Imperfect gradients

- *This talk*
  - First-order (gradient-based) methods over various function classes
  - When the training data is distributed over a network of nodes (machines, devices, robots)

Some Preliminaries

Smooth function classes

- $f : \mathbb{R}^p \to \mathbb{R}$ is L-smooth and $f(\mathbf{x}) \geq f^* > -\infty, \forall \mathbf{x}$
  - Not necessarily convex, bounded above by a quadratic
  - Assumed throughout
- $f : \mathbb{R}^p \to \mathbb{R}$ is convex (lies above all of its tangents)
- f is μ-strongly-convex (convex and bounded below by a quadratic)
  - For S & SC functions, we have $\kappa := {}^{L}/_{\mu} \geq 1$

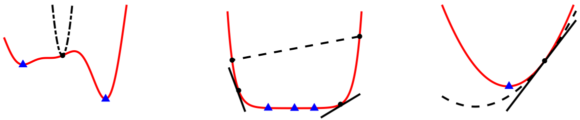


Figure 3: Nonconvex: $\sin(ax)(x + bx^2)$. Convexity. Strong Convexity.

Finding minima of smooth functions $f : \mathbb{R}^p \to \mathbb{R}$

- Search for a *stationary point* $\mathbf{x}^* \in \mathbb{R}^p$, i.e., $\nabla f(\mathbf{x}^*) = \mathbf{0}_p$

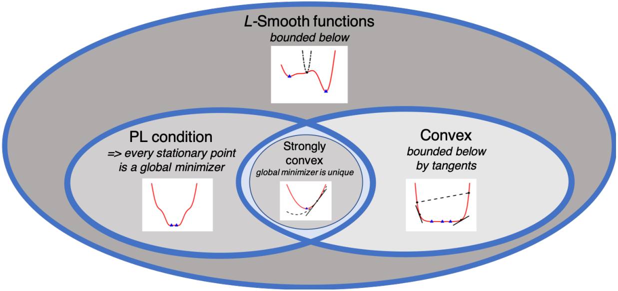


Figure 4: Function classes restricted to L-smooth functions

- Nonconvex: $\mathbf{x}^*$ may be a minimum, a maximum, or a saddle point
- Convex functions: $f(\mathbf{x}^*)$ is the unique global minimum
- Strongly convex functions: $\mathbf{x}^*$ is the unique global minimizer

First-order methods (Gradient Descent)

$$\min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x})$$

- Search for a **stationary point** $\mathbf{x}^*$, i.e., $\nabla f(\mathbf{x}^*) = \mathbf{0}_p$
- Intuition: Take a step in the direction opposite to the gradient
  - At $\star$, $\nabla f(\mathbf{x}^*) = \mathbf{0}_p$

Figure 5: Minimizing strongly convex functions: $\mathbb{R} \to \mathbb{R}$ and $\mathbb{R}^2 \to \mathbb{R}$

- **Gradient Descent (GD) intuition**: $\mathbf{x}_{new} \simeq \mathbf{x}_{old} - \nabla f(\mathbf{x}_{old})$
  - $f(\mathbf{x}_{new}) \leq f(\mathbf{x}_{old})$

GD: Performance metrics and Rates

- **Gradient Descent:** $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \cdot \nabla f(\mathbf{x}_k)$

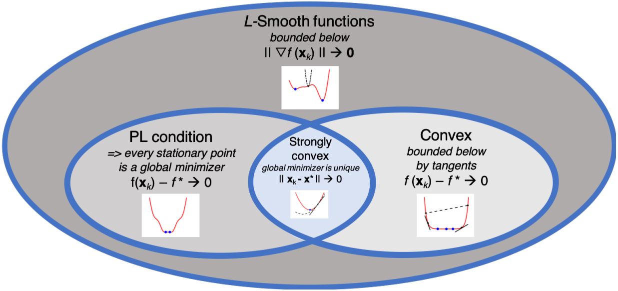


Figure 6: Function classes restricted to L-smooth functions

- Convergence rates of GD (non-stochastic and not accelerated):
  - Nonconvex: $||\nabla f(\mathbf{x}_k)|| \to 0$ at $\mathcal{O}(1/\sqrt{k})$
  - Convex: $f(\mathbf{x}_k) - f(\mathbf{x}^*) \to 0$ at $\mathcal{O}(1/k)$
  - SC: $f(\mathbf{x}_k) - f(\mathbf{x}^*) \to 0$ exponentially (linearly on the log-scale)
    - $\|\mathbf{x}_k - \mathbf{x}^*\|$ is also often used

Gradient Descent: Optimality

- **Gradient Descent**: $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \cdot \nabla f(\mathbf{x}_k)$
- Is this the fastest *first-order* algorithm? No!
- Consider ellipses in $\mathbb{R}^2$: $f(x_1, x_2) = Lx_1^2 + \mu x_2^2$, with $L \gg \mu$
  - L-smooth and μ-strongly convex functions, in general

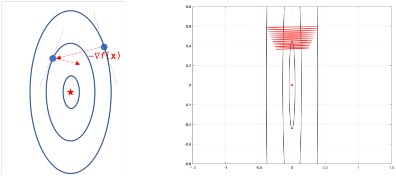


Figure 7: Convergence when $\kappa := L/\mu = 100$: First 25 iterations

- To get ϵ close to the minimizer, GD requires $\kappa \ln(1/\epsilon)$ iterations

The Heavy-ball Method (Accelerated GD)

- Gradient descent **with heavy-ball acceleration** [Polyak 1964]

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \cdot \nabla f(\mathbf{x}_k) + \beta \cdot (\mathbf{x}_k - \mathbf{x}_{k-1})$$

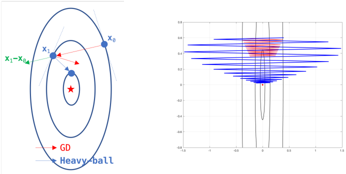


Figure 8: Convergence when $\kappa = 100$: First 25 iterations

- To get ϵ close, for L-smooth and μ-strongly convex functions

$$\text{GD: } \kappa \ln(1/\epsilon) \qquad \text{vs.} \qquad \text{HB: } \sqrt{\kappa} \ln(1/\epsilon)$$

- Nesterov acceleration has similar results with better guarantees
- *This type of acceleration does not buy us much in stochastic nonconvex problems .. more on this later!*

Distributed optimization

How to extend GD when the data is distributed?

Linear regression over distributed data

$$\min_{\texttt{slope, int.}} \sum_{i=1}^{\text{machines}} \sum_{j=1}^{\text{local data}} \left(y_{ij} - (\texttt{slope} \cdot d_{ij} + \texttt{int.})\right)^2$$

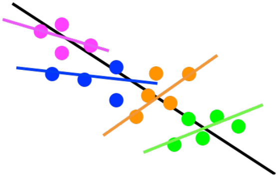


Figure 9: Linear regression: Locally optimal solutions

- Implement **local GD** at each node i: $\mathbf{x}_{k+1}^i = \mathbf{x}_k^i - \alpha \cdot \nabla f_i(\mathbf{x}_k^i)$
- Local GD does not lead to agreement on the optimal solution
- Requirements for a distributed algorithm
  - Agreement: Each node agrees on the same solution
  - Optimality: The agreed upon solution is the optimal

Distributed Optimization *(formally)*

$$\min_{\mathbf{x}\in\mathbb{R}^p} F(\mathbf{x}), \qquad F(\mathbf{x}) := \sum_{i=1}^{n} f_i(\mathbf{x})$$

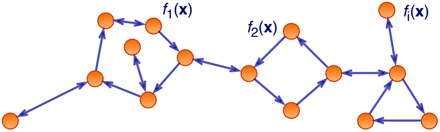


Figure 10: A peer-to-peer or edge computing architecture

Assumptions

- Each f_i is private to node i (nodes do not share their data)
- Each f_i is L_i-smooth and μ_i-strongly-convex (*assumed for now!*)
- The nodes communicate over a network (a connected graph)

Distributed Gradient Descent (DGD)

$$\mathbf{x}_{k+1}^i = \sum_{r=1}^{n} w_{ir} \cdot \mathbf{x}_k^r - \alpha \cdot \nabla f_i(\mathbf{x}_k^i)$$

- Mix and Descend [Nedić et al. '09]
  - The weight matrix $W = \{w_{ij}\}_{\geq 0}$ is doubly stochastic
  - DGD converges linearly (on a log-scale) up to a steady-state error for smooth and strongly convex problems
  - Exact convergence with a decaying step-size but at a sublinear rate

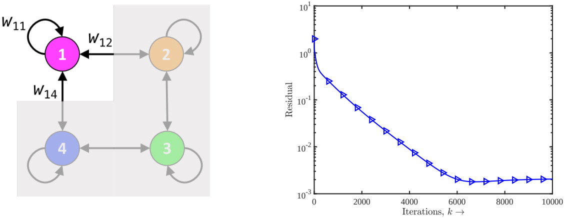


Figure 11: (Left) An undirected graph. (Right) DGD performance.

Recap

- GD and Distributed GD

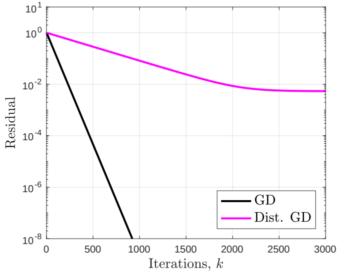


Figure 12: Performance for smooth and strongly convex problems

- How do we remove the steady-state error in DGD?

Distributed Gradient Descent with Gradient Tracking

GT-DGD: Intuition

- Problem: $\min_{\mathbf{x}} \sum_i f_i(\mathbf{x})$, i.e., search for $\mathbf{x}^*$ such that $\sum_i \nabla f_i(\mathbf{x}^*) = \mathbf{0}_p$

- DGD does not reach $\mathbf{x}^*$ because $\mathbf{x}^*$ is not its fixed point

$$\mathbf{x}_{k+1}^i = \sum_{r=1}^n w_{ir} \cdot \mathbf{x}_k^r - \alpha \cdot \nabla f_i(\mathbf{x}_k^i)$$
$$\mathbf{x}^* \neq 1 \cdot \mathbf{x}^* - \alpha \cdot \nabla f_i(\mathbf{x}^*)$$

  - This is because $\nabla f_i(\mathbf{x}^*) \neq 0$ but only the sum gradient is

- Fix: Replace $\nabla f_i(\mathbf{x}_k^i)$ with $\mathbf{y}_k^i$ that **tracks** the global gradient ∇F

- Linear convergence in distributed optimization (SSC)
  - Undirected graphs: [Xu et al. '15], [Lorenzo et al. '15]
  - Directed graphs[1,2]: [Xi-Khan '15], [Xi-Xin-Khan '16,'17], [Xin-Khan '18]

---

1. C. Xi and U. A. Khan, "DEXTRA: A Fast Algorithm for Optimization Over Directed Graphs," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, 4980-4993, Oct. 2017. Arxiv: Oct. 2015.
2. R. Xin, S. Pu, A. Nedić, and U. A. Khan, "A general framework for decentralized optimization with first-order methods," *Proceedings of the IEEE*, 118(11), pp. 1869-1889, Nov. 2020.

AB Algorithm *(our work)*

- Problem: $\min_{\mathbf{x}} \sum_i f_i(\mathbf{x})$
- DGD: $\mathbf{x}_{k+1}^i = \sum_{r=1}^n w_{ir} \cdot \mathbf{x}_k^r - \alpha \cdot \nabla f_i(\mathbf{x}_k^i)$

Algorithm 1 [Xin-Khan '18]: at each node i

Data: $\mathbf{x}_0^i \in \mathbb{R}^p$; $\alpha > 0$; $\{a_{ir}\}_{r=1}^n$; $\{b_{ir}\}_{r=1}^n$; $\mathbf{y}_0^i = \nabla f_i(\mathbf{x}_0^i)$
for $k = 0, 1, \ldots,$ **do**

$$\mathbf{x}_{k+1}^i = \sum_{r=1}^n a_{ir} \cdot \mathbf{x}_k^r - \alpha \cdot \mathbf{y}_k^r$$

$$\mathbf{y}_{k+1}^i = \sum_{r=1}^n b_{ir} \cdot \mathbf{y}_k^r + \nabla f_i(\mathbf{x}_{k+1}^i) - \nabla f_i(\mathbf{x}_k^i)$$

end

- $A = \{a_{ir}\}$ is row stochastic and $B = \{b_{ir}\}$ is column stochastic
  - Existing work by then: Both A and B are doubly stochastic
- AB unifies many existing algorithms
  - ADDOPT [Xi-Xin-Khan '16] and PUSH-DIGing [Nedić et al. '17]
  - FROST [Xin-Xi-Khan '16, '18]†: Only requires RS matrices

† *EURASIP 2022 Best Journal Paper Award for articles published over 2017-2021*

AB Algorithm *(our work)*

- Problem: $\min_{\mathbf{x}} \sum_i f_i(\mathbf{x})$
- DGD: $\mathbf{x}_{k+1}^i = \sum_{r=1}^n w_{ir} \cdot \mathbf{x}_k^r - \alpha \cdot \nabla f_i(\mathbf{x}_k^i)$

Algorithm 1 [Xin-Khan '18]: at each node i

Data: $\mathbf{x}_0^i \in \mathbb{R}^p$; $\alpha > 0$; $\{a_{ir}\}_{r=1}^n$; $\{b_{ir}\}_{r=1}^n$; $\mathbf{y}_0^i = \nabla f_i(\mathbf{x}_0^i)$
for $k = 0, 1, \ldots,$ **do**

$$\mathbf{x}_{k+1}^i = \sum_{r=1}^n a_{ir} \cdot \mathbf{x}_k^r - \alpha \cdot \mathbf{y}_k^r + \beta \cdot (\mathbf{x}_k^i - \mathbf{x}_{k-1}^i)$$

$$\mathbf{y}_{k+1}^i = \sum_{r=1}^n b_{ir} \cdot \mathbf{y}_k^r + \nabla f_i(\mathbf{x}_{k+1}^i) - \nabla f_i(\mathbf{x}_k^i)$$

end

- AB converges linearly to $\mathbf{x}^*$ with the help of **Gradient Tracking**
  - For SSC functions and over both directed and undirected graphs
- We can further add heavy-ball or Nesterov momentum

AB Algorithm *(our work)*

- Challenge: In general, neither a row stochastic matrix A nor a column stochastic matrix B leads to a contraction in 2-norm

$$\|W - W^{\infty}\|_2 < 1 \qquad (W\mathbf{1} = \mathbf{1}, \mathbf{1}^\top W = \mathbf{1}^\top)$$
$$\|A - A^{\infty}\|_2 \not< 1$$
$$\|B - B^{\infty}\|_2 \not< 1$$

- Key to the precise analysis and rates are two new norms
- Matrix norms induced by weighted vector norms

$$\|\cdot\|_A := \|\mathrm{diag}(\sqrt{\boldsymbol{\pi}_r})(A - A^{\infty})\mathrm{diag}(\sqrt{\boldsymbol{\pi}_r})^{-1}\|_2 \; < 1$$
$$\|\cdot\|_B := \|\mathrm{diag}(\sqrt{\boldsymbol{\pi}_c})(B - B^{\infty})\mathrm{diag}(\sqrt{\boldsymbol{\pi}_c})^{-1}\|_2 < 1$$

where $\boldsymbol{\pi}_r^\top A = \boldsymbol{\pi}_r^\top$ and $B\boldsymbol{\pi}_c = \boldsymbol{\pi}_c$

AB: Results (Smooth and Strongly convex)

- Linear convergence of AB over both directed and undirected graphs
  - [Xin-Khan '18]: For a range of step-sizes $\alpha \in (0, \bar{\alpha}]$
  - [Xin-Khan '18]: For non-identical step-sizes α_i's at the nodes
  - [Pu et al. '18]: Over mean-connected graphs
  - [Saadatniaki-Xin-Khan '18]: Over time-varying random graphs
  - [Various authors]: Asynchronous, delays, nonconvex analysis (but without explicit rates)
- Condition number dependence
  - GD κ, AB undirected $\kappa^{5/4}$, AB directed κ^2
- Gradient tracking with heavy-ball momentum
  - [Xin-Khan '18]: Linear convergence for a range of alg. parameters
  - *Acceleration is not proved analytically and remains an open problem*
- Gradient tracking with Nesterov momentum
  - [Qu et al. '18]: Undirected graphs $\kappa^{5/7}$
  - [Xin-Jakovetić-Khan '19]: Convergence and acceleration are shown numerically over directed graphs
  - *Directed graphs: Convergence and acceleration both remain open*

Performance comparison

- GD, HB, DGD, AB, ABm

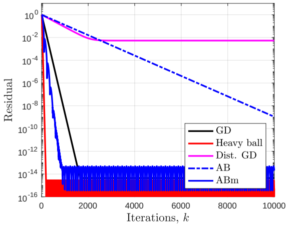


Figure 13: Performance for smooth and strongly convex problems, $\kappa = 100$

- Addition of gradient tracking recovers linear convergence (proved!)
- Acceleration can be shown numerically but it is not proved (yet!)

- What happens when the gradients are imperfect?

Distributed Stochastic Optimization

- Stochastic gradients with noise variance ν^2

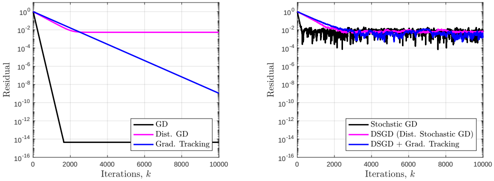


Figure 14: Full gradients ($\nu^2 = 0$) vs. stochastic gradients

- **DSGD**: Residual decays **linearly** to an error ball [Yuan et al. '19]

$$\limsup_{k\to\infty} \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}[\|\mathbf{x}_k^i - \mathbf{x}^*\|_2^2] = \mathcal{O}\Big(\frac{\alpha}{n\mu}\nu^2 + \frac{\alpha^2\kappa^2}{1-\lambda}\nu^2 + \frac{\alpha^2\kappa^2}{(1-\lambda)^2}\eta\Big),$$

where η quantifies local-vs.-global bias $\simeq \left\|\nabla f_i(\mathbf{x}^*) - \sum_i \nabla f_i(\mathbf{x}^*)\right\|$

- **Gradient tracking eliminates η but the variance remains**

Distributed Stochastic Optimization

Batch problems: The GT+VR framework

Batch problems: Setup

$$\min_{\texttt{slope, int.}} \sum_{i=1}^{\text{machines}} \underbrace{\sum_{j=1}^{\text{local data}} \underbrace{\left(y_{ij} - (\texttt{slope} \cdot d_{ij} + \texttt{int.})\right)^2}_{f_{ij}(\mathbf{x})}}_{f_i(\mathbf{x})}$$

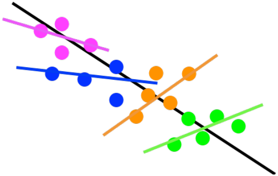


Figure 15: Linear regression (revisited)

- Each node i possesses a local batch of m_i data samples
- The local cost f_i is the sum over all data samples $\sum_{j=1}^{m_i} f_{ij}$

Batch Problems

- Minimize $F := \sum_i \sum_j f_{ij}$ over arbitrary data distributions

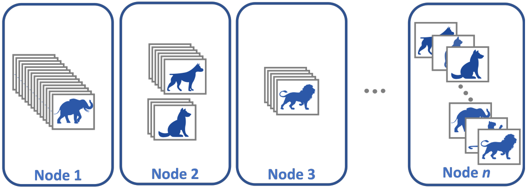


Figure 16: Arbitrary data distribution over the network

- Computing local batch gradient $\sum_j \nabla f_{ij}$ is typically expensive

- Distributed Stochastic GD: $\mathbf{x}_{k+1}^i = \sum_r w_{ir} \cdot \mathbf{x}_k^r - \alpha \cdot \nabla f_{i\tau_k}(\mathbf{x}_k^i)$
  - Choose τ_k randomly from $1, \ldots, m_i$
  - Challenges: $\nabla f_{i\tau_k} \neq \nabla f_i \neq \nabla F$

GT+VR framework

- Problem: Minimize $F := \sum_i \sum_j f_{ij}$

- The GT+VR framework[1,2]: From $\nabla f_{i,\tau_k}$ to ∇F
  - Local variance reduction: **Sample** then **Estimate**

$$\nabla f_{i,\tau_k} + \text{VR} \rightarrow \mathbf{v}_i \simeq \nabla f_i = \sum_{j=1}^{m_i} \nabla f_{ij}$$

  - Global gradient tracking: **Fuse** the estimates over the network

$$\mathbf{v}_i + \text{GT} \rightarrow \mathbf{y}_i \simeq \nabla F = \sum_{i=1}^{n} \nabla f_i$$

- Popular VR methods: SAG, SAGA, SVRG, SPIDER, SARAH
- Our work[1]: GT-SAGA, GT-SVRG, GT-SARAH[2]

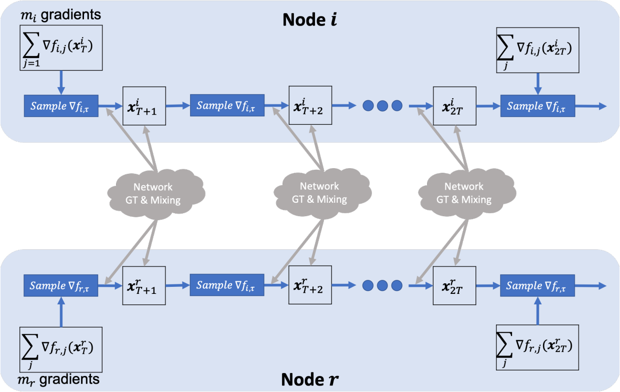
---

1. R. Xin, S. Kar, and U. A. Khan, "Gradient tracking and variance reduction for decentralized optimization and machine learning," *IEEE Signal Processing Magazine*, 37(3), pp. 102-113, May 2020.

2. R. Xin, U. A. Khan, and S. Kar, "Fast decentralized nonconvex finite-sum optimization with recursive variance reduction," *SIAM Journal on Optimization*, 32(1), 2022.

GT-SARAH

- GT-SARAH (StochAstic Recursive grAdient algoritHm)
  - The blue boxes show sample and estimate



GT-SARAH: Smooth and nonconvex
Almost sure and mean-squared results

$$\min_{\mathbf{x}} \sum_{i=1}^{n} \sum_{j=1}^{m} f_{ij}(\mathbf{x})$$

- GT plus SARAH based VR
  - Assume $m_i = m, \forall i$, for simplicity
  - The estimate at each node converges to a stationary point both in almost sure and mean-squared sense

Theorem (Xin-Khan-Kar '20)

At each node i, GT-SARAH's iterate $\mathbf{x}_k^i$ follows

$$\mathbb{P}\left(\lim_{k\to\infty} \|\nabla F(\mathbf{x}_k^i)\| = 0\right) = 1 \qquad \textit{and} \qquad \lim_{k\to\infty} \mathbb{E}\left[\left\|\nabla F(\mathbf{x}_k^i)\right\|^2\right] = 0.$$

GT-SARAH: Smooth and nonconvex

$$\min_{\mathbf{x}} \sum_{i=1}^{n} \sum_{j=1}^{m} f_{ij}(\mathbf{x})$$

- Total of $N = nm$ data points divided equally among n nodes
- How many gradient computations are required to reach an ϵ-accurate solution?

Theorem (Gradient computation complexity, Xin-Khan-Kar '20)

Under a certain constant step-size α, GT-SARAH, with $\mathcal{O}(m)$ inner loop iterations, reaches an ϵ-optimal stationary point of the global cost F in

$$\mathcal{H} := \mathcal{O}\left(\max\left\{N^{1/2}, \frac{n}{(1-\lambda)^2}, \frac{(n+m)^{1/3} n^{2/3}}{1-\lambda}\right\}\left(c \cdot L + \frac{1}{n}\sum_{i=1}^{n} \left\|\nabla f_i(\overline{\mathbf{x}}_0)\right\|^2\right)\frac{1}{\epsilon}\right)$$

gradient computations across all nodes, where $c := F(\overline{\mathbf{x}}_0) - F^*$.

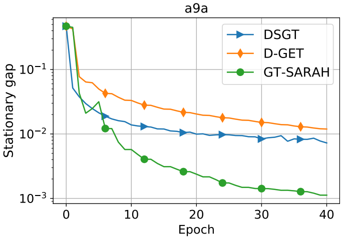
GT-SARAH: Smooth and nonconvex
Optimal complexity

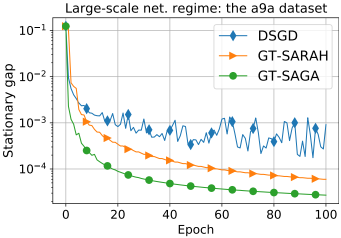
$$\min_{\mathbf{x}} \sum_{i=1}^{n} \sum_{j=1}^{m} f_{ij}(\mathbf{x})$$

- Total of $N = nm$ data points divided equally among n nodes
- How many gradient computations are required to reach an ϵ-accurate solution?
- In a big-data regime $n \leq \mathcal{O}(m(1-\lambda)^6)$: $\mathcal{H} = \mathcal{O}(N^{1/2}\epsilon^{-1})$
  - Matches the centralized optimal lower bound [SPIDER: Fang et al. '18]
- Other notable features [Xin-Kar-Khan '20, Xin-Khan-Kar '22]:
  - *Independent of the variance and the local vs. global bias*
  - Network-topology independent convergence rate and performance
  - Linear speedup: GT-SARAH is n times faster than the cent. SARAH

Experiments: Nonconvex binary classification

- Performance Comparison





- Big-data regime
- 10×10 grid graph

- IoT regime
- Nearest neighbor graph

Distributed optimization: Demo

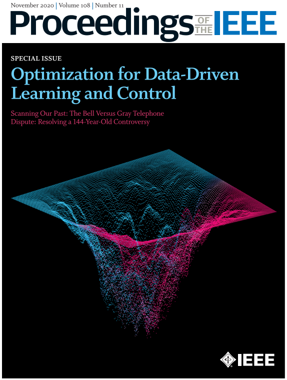
- Full gradient, distributed linear regression, $n = 100$ nodes
  - One data point per node; collaborate to learn the slope and intercept
- `https://www.eecs.tufts.edu/~khan/Demos/LR_int_digraph_KHAN_n500_1.mp4`

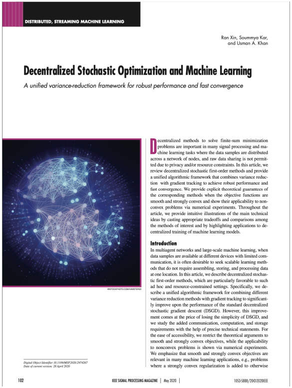
Conclusions

- Gradient tracking for distributed optimization
  - GT eliminates the local vs. global dissimilarity bias
  - Linear convergence for smooth and strongly convex problems
  - Acceleration is achievable but analysis is hard!
- GT+VR: Gradient tracking for distributed batch optimization
  - GT-SAGA, GT-SVRG, GT-SARAH (**optimal** in the big-data regime)
  - Network synchrony and storage tradeoffs
- Gradient tracking for distributed streaming problems
  - Shown best known rates for strongly convex and nonconvex problems
  - Decaying step-sizes eliminate the variance due to the stochastic grad
  - Hybrid VR techniques
- Network-independent convergence behavior
- Outperforms the centralized analogs in applicable regimes

There is a lot more being done and a lot more to do!

- *Some reader-friendly overview articles*
- P-IEEE Special Issue, vol. 108, no. 11, Nov. 2020
  U. A. Khan, *Lead Editor*, with Guest Eds.: Bajwa, Nedić, Rabbat, Sayed
- Our May 2020 SPM article





DISTRIBUTED, STREAMING MACHINE LEARNING

Ran Xin, Soummya Kar, and Usman A. Khan

Decentralized Stochastic Optimization and Machine Learning

A unified variance-reduction framework for robust performance and fast convergence

Decentralized methods to solve finite-sum minimization problems are important in many signal processing and machine learning tasks where the data samples are distributed across a network of nodes, and raw data sharing is not permitted due to privacy and/or resource constraints. In this article, we review decentralized stochastic first-order methods and provide a unified algorithmic framework that combines variance reduction with gradient tracking to achieve robust performance and fast convergence. We provide explicit theoretical guarantees of the corresponding methods when the objective functions are smooth and strongly convex and show their applicability to nonconvex problems via numerical experiments. Throughout the article, we provide intuitive illustrations of the main technical ideas by casting appropriate tradeoffs and comparisons among the methods of interest and by highlighting applications to decentralized training of machine learning models.

Introduction

In multiagent networks and large-scale machine learning, when data samples are available at different devices with limited communication, it is often desirable to seek scalable learning methods that do not require assembling, storing, and processing data at one location. In this article, we describe decentralized stochastic first-order methods, which are particularly favorable to such ad hoc and resource-constrained settings. Specifically, we describe a unified algorithmic framework for combining different variance reduction methods with gradient tracking to significantly improve upon the performance of the standard decentralized stochastic gradient descent (DSGD). However, this improvement comes at the price of losing the simplicity of DSGD, and we study the added communication, computation, and storage requirements with the help of precise technical statements. For the ease of accessibility, we restrict the theoretical arguments to smooth and strongly convex objectives, while the applicability to nonconvex problems is shown via numerical experiments. We emphasize that smooth and strongly convex objectives are relevant in many machine learning applications, e.g., problems where a strongly convex regularization is added to otherwise

Digital Object Identifier 10.1109/MSP.2020.2974267
Date of current version: 28 April 2020

102 IEEE SIGNAL PROCESSING MAGAZINE | May 2020 | 1053-5888/20©2020IEEE

GT-SARAH: Analysis

GT-SARAH: Analysis

- Use the L-smoothness of F

$$F(\mathbf{y}) \leq F(\mathbf{x}) + \langle \nabla F(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \tfrac{L}{2} \left\| \mathbf{y} - \mathbf{x} \right\|^2, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^p,$$

  to establish the following lemma

Lemma (Descent inequality)

If the step-size follows that $0 < \alpha \leq \frac{1}{2L}$, *then we have*

$$\mathbb{E}\left[F\left(\overline{\mathbf{x}}^{T+1,K}\right)\right] \leq F(\overline{\mathbf{x}}^{0,1}) - \frac{\alpha}{2}\sum_{k,t}^{K,T}\mathbb{E}\left[\left\|\nabla F(\overline{\mathbf{x}}^{t,k})\right\|^2\right]$$

$$-\alpha\left(\frac{1}{4}\sum_{k,t}^{K,T}\mathbb{E}\left[\left\|\overline{\mathbf{v}}^{t,k}\right\|^2\right] - \sum_{k,t}^{K,T}\mathbb{E}\left[\left\|\overline{\mathbf{v}}^{t,k} - \overline{\nabla \mathbf{f}}(\mathbf{x}^{t,k})\right\|^2\right] - L^2\sum_{k,t}^{K,T}\mathbb{E}\left[\frac{\left\|\mathbf{x}^{t,k} - \mathbf{1}\otimes\overline{\mathbf{x}}^{t,k}\right\|^2}{n}\right]\right)$$

- The object in red has two errors that we need to bound
  - Gradient estimation error: $\mathbb{E}[\|\overline{\mathbf{v}}^{t,k} - \overline{\nabla \mathbf{f}}(\mathbf{x}^{t,k})\|^2]$
  - Agreement error: $\mathbb{E}[\|\mathbf{x}^{t,k} - \mathbf{1}\otimes\overline{\mathbf{x}}^{t,k}\|^2]$

GT-SARAH: Analysis

Lemma (Gradient estimation error)

We have $\forall k \geq 1$,

$$\sum_{t=0}^{T} \mathbb{E}\left[\|\bar{\mathbf{v}}^{t,k} - \overline{\nabla \mathbf{f}}(\mathbf{x}^{t,k})\|^2\right] \leq \frac{3\alpha^2 T L^2}{n} \sum_{t=0}^{T-1} \mathbb{E}\left[\|\bar{\mathbf{v}}^{t,k}\|^2\right] + \frac{6TL^2}{n} \sum_{t=0}^{T} \mathbb{E}\left[\frac{\|\mathbf{x}^{t,k} - \mathbf{1} \otimes \bar{\mathbf{x}}^{t,k}\|^2}{n}\right].$$

Lemma (Agreement error)

If the step-size follows $0 < \alpha \leq \frac{(1-\lambda^2)^2}{8\sqrt{42}L}$, *then*

$$\sum_{k=1}^{K} \sum_{t=0}^{T} \mathbb{E}\left[\frac{\|\mathbf{x}^{t,k} - \mathbf{1} \otimes \bar{\mathbf{x}}^{t,k}\|^2}{n}\right] \leq \frac{64\alpha^2}{(1-\lambda^2)^3} \frac{\|\nabla \mathbf{f}(\mathbf{x}^{0,1})\|^2}{n} + \frac{1536\alpha^4 L^2}{(1-\lambda^2)^4} \sum_{k=1}^{K} \sum_{t=0}^{T} \mathbb{E}\left[\|\bar{\mathbf{v}}^{t,k}\|^2\right].$$

- *Agreement error is coupled with the gradient estimation error*
- *Derive an LTI system that describes their evolution*
- *Analyze the LTI dynamics to obtain the agreement error lemma*

- Use the two lemmas back in the descent inequality

GT-SARAH: Analysis

Lemma (Refined descent inequality)

For $0 < \alpha \leq \overline{\alpha} := \min\left\{\frac{(1-\lambda^2)^2}{4\sqrt{42}}, \frac{\sqrt{n}}{\sqrt{6T}}, \left(\frac{2n}{3n+12T}\right)^{\frac{1}{4}}\frac{1-\lambda^2}{6}\right\}\frac{1}{2L}$, *we have*

$$\frac{1}{n}\sum_{i,k,t}^{n,K,T}\mathbb{E}\left[\|\nabla F(\mathbf{x}_i^{t,k})\|^2\right] \leq \frac{4(F(\overline{\mathbf{x}}^{0,1}) - F^*)}{\alpha} + \left(\frac{3}{2} + \frac{6T}{n}\right)\frac{256\alpha^2 L^2}{(1-\lambda^2)^3}\frac{\left\|\nabla \mathbf{f}(\mathbf{x}^{0,1})\right\|^2}{n}.$$

- Taking $K \to \infty$ on both sides leads to $\sum_{k,t}^{\infty,T}\mathbb{E}[\|\nabla F(\mathbf{x}_i^{t,k})\|^2] < \infty$
  - Mean-squared and a.s. results follow

- Divide both sides by $K \cdot T$ and solve for K when the R.H.S $\leq \epsilon$
  - Gradient computation complexity follows by nothing that GT-SARAH computes $n(m + 2T)$ gradients per iteration across all nodes
  - Choose α as the maximum and $T = \mathcal{O}(m)$ to obtain the optimal rate