Distributed ML
Optimal algorithms for distributed stochastic nonconvex optimization

Usman A. Khan
Electrical and Computer Engineering, Tufts University

Seminar in Mathematics, Physics & Machine Learning
IST, Lisbon, Portugal
July 09, 2021
Acknowledgments

Reza D. (2011-15)
C. Xi (2012-17)
S. Safavi (2013-17)
F. Saadatniaki (2014-19)

R. Xin (2016- )
M. I. Qureshi (2018- )
A. Swar (2020- )
H. Raja (2021- )
Research Overview
Learning from Data

- Data is everywhere and holds a significant potential
  - Image classification, Medical diagnosis, Credit card fraud, …

![Centralized and distributed learning architectures](image)

**Figure 1:** Centralized and distributed learning architectures

- Collecting all data at a central location may not be practical
  - Large, private, datasets with communication constraints

- Distributed methods rely on local processing and communication
A simple case study . . .

Figure 2: Test accuracy of a model trained with 10,000 32 × 32 pixel images

- When do distributed methods outperform their centralized analogs?
- How do we formally quantify such a comparison?
Some Preliminaries
Example: Recognizing Traffic Signs

- Identify STOP vs. YIELD sign

**Figure 3:** Binary classification: (Left) Training phase (Right) Testing phase

- Input data: images \( \{ \theta_j \} \) and their labels \( \{ y_j \} \)
- Model: A classifier \( x \) that predicts a label \( \hat{y}_j \) for each image \( \theta_j \)
  - Changing \( x \) changes the predicted label \( \hat{y}_j (x; \theta_j) \)

- Pick a classifier \( x^* \) that minimizes some loss over all images

\[
x^* = \arg\min_{x \in \mathbb{R}^p} \sum_j \ell(y_j, \hat{y}_j (x; \theta_j))
\]
Minimizing Functions

\[
\min_x f(x), \quad f := \sum_j \ell(y_j, \hat{y}_j(x; \theta_j)) : \mathbb{R}^p \to \mathbb{R}
\]

- Different predictors \( \hat{y} \) and losses \( \ell \) lead to different cost functions \( f \)
- **Quadratic**: Signal estimation, linear regression, LQR
- **(Strongly) convex**: Logistic regression, classification
- **Nonconvex**: Neural networks, reinforcement learning, blind sensing

**This talk**
- First-order (gradient-based) methods over various function classes
  - Search for a point \( x^* \in \mathbb{R}^p \) such that \( \nabla f(x^*) = 0_p \)
  - When the training data is distributed over a network of nodes (machines, devices, robots)
Basic Definitions

- \( f : \mathbb{R}^p \to \mathbb{R} \) is \( L \)-smooth and \( f(x) \geq f^* \geq -\infty, \forall x \)
  - 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 \( \mu \)-strongly-convex (convex and bounded below by a quadratic)
  - For SC functions, we have \( \kappa := \frac{L}{\mu} \geq 1 \)

Figure 4: Nonconvex: \( \sin(ax)(x + bx^2) \). Convexity. Strong Convexity.
Minimizing smooth (differentiable) functions $f : \mathbb{R}^p \rightarrow \mathbb{R}$
- Search for a stationary point $x^* \in \mathbb{R}^p$, i.e., $\nabla f(x^*) = 0_p$

**Figure 5:** Function classes restricted to $L$-smooth functions

- Nonconvex: $x^*$ may be a minimum, a maximum, or a saddle point
- Convex (and PL) functions: $f(x^*)$ is the unique global minimum
- Strongly convex functions: $x^*$ is the unique global minimizer
First-order methods (Gradient Descent)

\[ \min_{x \in \mathbb{R}^p} f(x) \]

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

![Diagram showing gradient descent](image)

**Figure 6**: Minimizing strongly convex functions: \( \mathbb{R} \to \mathbb{R} \) and \( \mathbb{R}^2 \to \mathbb{R} \)

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

- 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 (and PL): $f(\mathbf{x}_k) - f(\mathbf{x}^*) \to 0$ and $||\mathbf{x}_k - \mathbf{x}^*|| \to 0$ exponentially (linearly on the log-scale)
How to extend GD when the data is distributed?

- Let’s consider a simple example: Linear Regression
- Implement **local GD** at each node \(i\): 
  \[
  x_{k+1}^i = x_k^i - \alpha \cdot \nabla f_i(x_k^i)
  \]

**Figure 8:** Linear regression: Locally optimal solutions

- 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

Smooth and strongly convex problems with full gradients
Distributed Optimization

\[
\min_{x \in \mathbb{R}^p} F(x), \quad F(x) := \sum_{i=1}^{n} f_i(x)
\]

**Figure 9:** A peer-to-peer or edge computing architecture

**Assumptions**

- Each \( f_i \) is private to node \( i \)
- Each \( f_i \) is \( L_i \)-smooth and \( \mu_i \)-strongly-convex (assumed for now!)
- The nodes communicate over a network (a connected graph)

- \( F \) has a unique global minimizer \( x^* \) such that \( \nabla F(x^*) = 0_p \)
Distributed Gradient Descent (DGD)

\[ x_{k+1}^{i} = \sum_{r=1}^{n} w_{ir} \cdot x_{r}^{k} - \alpha \cdot \nabla f_{i}(x_{k}^{i}) \]

- **Mix and Descend** [Nedić et al. ’09]
  - The weight matrix \( W = \{w_{ij}\}_{i,j} \geq 0 \) sums to 1 on rows and columns
  - DGD converges linearly (on a log-scale) up to a steady-state error
  - Exact convergence with a decaying step-size but at a sublinear rate

![Diagram of an undirected graph](image1)

**Figure 10:** (Left) An undirected graph. (Right) DGD performance.
Recap

- GD and Distributed GD

Figure 11: 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:** \( \min_x \sum_i f_i(x) \), i.e., search for \( x^* \) such that \( \sum_i \nabla f_i(x^*) = 0 \)

- **DGD does not reach** \( x^* \) **because** \( x^* \) **is not its fixed point**
  \[
  x_{k+1}^i = \sum_{r=1}^n w_{ir} \cdot x_k^r - \alpha \cdot \nabla f_i(x_k^i)
  \]

  \[
  x^* \neq \frac{1}{\alpha} \cdot x^* - \alpha \cdot \nabla f_i(x^*)
  \]

  This is because \( \nabla f_i(x^*) \neq 0 \) but only the sum gradient is

- **We call this the local-vs.-global dissimilarity bias** (\( \eta \approx \| \nabla f_i - \nabla F \| \))

- **Fix:** Replace \( \nabla f_i(x_k^i) \) with \( y_k^i \) that **tracks** the global gradient \( \nabla F \)
  \[
  x_{k+1}^i = \sum_{r=1}^n w_{ir} \cdot x_k^r - \alpha \cdot y_k^i
  \]

- **Linear convergence in distributed optimization (SSC)**
  - Undirected graphs: [Xu et al. '15], [Lorenzo et al. '15]
  - Directed graphs: [Xi-Khan '15], [Xi-Xin-Khan '16,'17], [Xin-Khan '18]
Problem: $\min_x \sum_i f_i(x)$

DGD: $x_{k+1}^i = \sum_{r=1}^n w_{ir} \cdot x_k^r - \alpha \cdot \nabla f_i(x_k^i)$

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

Data: $x_0^i \in \mathbb{R}^p$; $\alpha > 0$; $\{a_{ir}\}_{r=1}^n$; $\{b_{ir}\}_{r=1}^n$; $y_0^i = \nabla f_i(x_0^i)$

for $k = 0, 1, \ldots$ do

\[
x_{k+1}^i = \sum_{r=1}^n a_{ir} \cdot x_k^r - \alpha \cdot y_k^r
\]

\[
y_{k+1}^i = \sum_{r=1}^n b_{ir} \cdot y_k^r + \nabla f_i(x_{k+1}^i) - \nabla f_i(x_k^i)
\]

end

AB converges linearly to $x^*$ with the help of Gradient Tracking
- Over both directed and undirected graphs

We can further add heavy-ball or Nesterov momentum
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 $\alpha_i$’s at the nodes
  - [Pu et al. ’18]: Over mean-connected graphs
  - [Saadatniaki-Xin-Khan ’18]: Over time-varying random graphs
  - Asynchronous, delays, nonconvex analysis (but without explicit rates)

- Condition number dependence
  - GD $\kappa$, AB undirected $\kappa^{5/4}$, AB directed $\kappa^2$

- AB 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*

- AB 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 are both open*
Performance comparison

- GD, HB, DGD, AB, ABm

Figure 12: 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 $\nu^2$

  ![Graph](image)

  **Figure 13**: Full gradients ($\nu^2 = 0$) vs. stochastic gradients

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

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

  where $\eta$ quantifies the local-vs.-global dissimilarity bias

- **Gradient tracking eliminates $\eta$ but the variance remains**
Distributed Stochastic Optimization

Nonconvex problems
Distributed Stochastic Optimization: Measurement Model

\[
\min_x F(x), \quad F(x) := \sum_{i=1}^{n} f_i(x), \quad f_i : \mathbb{R}^p \rightarrow \mathbb{R}
\]

- **Online/Streaming**: Given some \( x \in \mathbb{R}^p \), each node \( i \) makes a noisy measurement of the local gradient \( \nabla f_i(x) \)

- **Offline/Batch**: Each node \( i \) possesses a local dataset with \( m_i \) data points and their corresponding labels, i.e., \( \nabla f_i(x) = \sum_{j=1}^{m_i} \nabla f_{i,j}(x) \)

---

**Figure 14**: (Left) Online streaming data (Right) Offline batch data
Distributed Stochastic Optimization: Communication Model

- Controllable topology
- $\# \text{ nodes} \ll \# \text{ local samples}$
- Big-data regime

- Ad hoc topology
- $\# \text{ local samples} \text{ is small}$
- IoT regime

Figure 15: Data Center

Figure 16: Internet of Things
Distributed Stochastic Optimization

- Gradient tracking eliminates $\eta$ (the local-vs.-global dissimilarity bias) but the variance $\nu^2$ remains.

- Can we quantify the improvement due to gradient tracking?
- Can we eliminate the steady-state error due to the variance?
- What can we say about different function classes?
Batch problems: The GT+VR framework
GT+VR framework

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

![Figure 17: Arbitrary data distribution over the network](image)

- Local Gradient computation $\sum_{j=1}^{m_i} \nabla f_{i,j}$ is prohibitively expensive
- Traditionally: $x_{k+1}^i = \sum_r w_{ir} \cdot x_r^i - \alpha \cdot \nabla f_{i,\tau}(x_k^i)$
- Performance is impacted due to sampling and local vs. global bias
GT+VR framework

- The GT+VR framework: From $\nabla f_{i,\tau}$ to $\nabla F = \sum_{i=1}^{n} \sum_{j=1}^{m_i} \nabla f_{i,j}$
  - Local variance reduction: **Sample** then **Estimate**
    
    $\nabla f_{i,\tau} \rightarrow \nabla f_i = \sum_{j=1}^{m_i} \nabla f_{i,j}$

  - Global gradient tracking: **Fuse** the estimates over the network
    
    $\nabla f_i \rightarrow \nabla F = \sum_{i=1}^{n} \nabla f_i$

- Popular VR methods: SAG, SAGA, SVRG, SPIDER, SARAH
- Our work\(^1\): GT-SAGA, GT-SVRG, GT-SARAH

---

GT-SAGA

- GT-SAGA: Requires $\mathcal{O}(m_i p)$ storage at each node

![Diagram of GT-SAGA at node i]

**Figure 18:** GT-SAGA at node $i$

- [Xin-Kar-Khan: May ’20, Xin-Khan-Kar: Nov. ’20]
  - Strongly convex problems: Linear convergence, improved rates
  - Linear speedup and network-independent convergence for both nonconvex and nonconvex with PL
GT-SARAH (StochAstic Recursive grAdient algoritHm)

- No storage but additional network synchrony when $m_i \neq m_r$
GT-SAGA vs. GT-SARAH

- A space vs. time tradeoff: Storage vs. Synchronization

- GT-SAGA: For ad hoc problems with heterogeneous data
- GT-SARAH: For very large-scale problem in controlled settings

We can show\textsuperscript{1,2} these tradeoffs theoretically!!!

\footnotesize

GT-SARAH: Smooth and nonconvex

- GT plus SARAH based VR
  - Assume $m_i = m, \forall i$, for simplicity

**Theorem (Almost sure and mean-squared results, Xin-Khan-Kar '20)**

*At each node $i$, GT-SARAH’s iterate $x^i_k$ follows*

\[
P \left( \lim_{k \to \infty} \| \nabla F(x^i_k) \| = 0 \right) = 1 \quad \text{and} \quad \lim_{k \to \infty} \mathbb{E} \left[ \| \nabla F(x^i_k) \|^2 \right] = 0.
\]
GT-SARAH: Smooth and nonconvex

\[
\min_{x} \sum_{i=1}^{n} \sum_{j=1}^{m} f_{i,j}(x)
\]

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

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

Under a certain constant step-size \(\alpha\), GT-SARAH, with \(O(m)\) inner loop iterations, reaches an \(\epsilon\)-optimal stationary point of the global cost \(F\) in

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

gradient computations across all nodes, where \(c := F(\bar{x}_0) - F^*\).
GT-SARAH: Smooth and nonconvex

\[ \min_x \sum_{i=1}^{n} \sum_{j=1}^{m} f_{i,j}(x) \]

- Total of \( N = nm \) data points divided equally among \( n \) nodes
- How many gradient computations are required to reach an \( \epsilon \)-accurate solution?
- In a certain big-data regime \( n \leq \mathcal{O}(m(1 - \lambda)^6) \): \( \mathcal{H} = \mathcal{O}(N^{1/2} \epsilon^{-1}) \)
  - Independent of the network topology
  - Linear speedup compared to centralized SARAH
GT-SARAH: Smooth and nonconvex

- Minimize a sum of $N := nm$ smooth nonconvex functions
- The rate $O(N^{1/2} \epsilon^{-1})$ in the big-data regime matches the centralized algorithmic lower bound for this problem class [SPIDER: Fang et al. '18]

- Independent of the variance of local gradient estimators
- Independent of the local vs. global dissimilarity bias
- Independent of the network
- Linear speedup
  GT-SARAH is $n$ times faster than the centralized SARAH
Experiments: Nonconvex binary classification

- Performance Comparison

- Big-data regime
  - 10 × 10 grid graph

- IoT regime
  - Nearest neighbor graph
Experiments: Nonconvex binary classification

- Effect of network topology in GT-SAGA

- Big-data regime

- IoT regime
What happens for streaming data where VR is not applicable?

- **GT-DSGD\(^1\)**: Vanilla distributed SGD + GT
- Decaying stepsizes can be used to kill the variance

- **GT-HSGD\(^2\)**: A novel way for variance reduction
  \[ \beta \cdot \text{(Local stoch. gradient)} + (1 - \beta) \cdot \text{(inner loop of SARAH)} \]
- Outperforms existing methods with a \( \beta \in (0, 1) \)

---


Distributed optimization: Demo

- Full gradient, distributed linear regression, \( n = 100 \) nodes
  - Each node possesses one data point
  - Collaborate to learn the slope and intercept
Conclusions

- Gradient tracking for distributed optimization
  - GT eliminates the local vs. global dissimilarity bias
  - Linear convergence for smooth and strongly convex problems
  - Acceleration is possible but analysis is hard!

- GT+VR: Gradient tracking for distributed batch optimization
  - GT-SAGA: State-of-the-art in the IoT regime
  - GT-SARAH: State-of-the-art in the big-data regime

- Gradient tracking for distributed online stochastic optimization
  - Shown best known rates for strongly convex and nonconvex problems in applicable regimes
  - 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!

P-IEEE Special Issue, vol. 108, no. 11
U. A. Khan, Lead Editor
with Guest Editors: W. U. Bajwa, A. Nedić, M. G. Rabbat, A. H. Sayed
GT-SARAH: Analysis
GT-SARAH: Analysis

- Use the $L$-smoothness of $F$ to establish the following lemma

\[ F(y) \leq F(x) + \langle \nabla F(x), y - x \rangle + \frac{L}{2} \| y - x \|^2 \quad \forall x, y \in \mathbb{R}^p \]

**Lemma (Descent inequality)**

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

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

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

We have $\forall k \geq 1$,

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

Lemma (Agreement error)

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

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

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

$$
\frac{1}{n} \sum_{i,k,t}^n \mathbb{E} \left[ \| \nabla F(x_i^{t,k}) \|^2 \right] \leq \frac{4(F(x_0^{0,1}) - F^*)}{\alpha} + \left( \frac{3}{2} + \frac{6T}{n} \right) \frac{256\alpha^2 L^2}{(1-\lambda^2)^3} \frac{\| \nabla f(x_0^{0,1}) \|^2}{n}.
$$

- Taking $K \rightarrow \infty$ on both sides leads to $\sum_{k,t}^\infty \mathbb{E} [\| \nabla F(x_i^{t,k}) \|] < \infty$
  - 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 noting that GT-SARAH computes $n(m + 2T)$ gradients per iteration across all nodes
  - Choose $\alpha$ as the maximum and $T = \mathcal{O}(m)$ to obtain the optimal rate