Distributed Optimization for Machine Learning

Lecture 15 - Communication-efficient Distributed Training - Part II

Tianyi Chen

School of Electrical and Computer Engineering
Cornell Tech, Cornell University

October 20, 2025

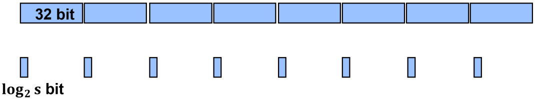
Table of Contents

Sparsification

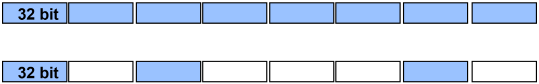
Goal: Reduce num of communicated entries by making vectors *sparse*.

Q: What is sparse?

- **Quantization:**



- **Sparsification:**



Idea: Communicate only a few coordinates and set the rest to zero.

Stochastic sparsification

For any $\boldsymbol{v} \in \mathbb{R}^d$, define a sparsified vector $Q(\boldsymbol{v})$ coordinate-wise by:

$$[Q(\boldsymbol{v})]_j = \begin{cases} \dfrac{v_j}{p_j}, & \text{with probability } p_j, \\ 0, & \text{with probability } 1 - p_j, \end{cases} \qquad j = 1, \ldots, d.$$

Let $\boldsymbol{p} = (p_1, \ldots, p_d)$ be a predetermined probability vector belonging to a simplex ($p_j \in (0, 1], \sum_{j=1}^{d} p_j = 1$).

Performance of stochastic sparsification

Lemma. For $\boldsymbol{v} \in \mathbb{R}^d$ and a sparsified vector $Q(\boldsymbol{v})$, it follows that

(i) **Unbiasedness:** $\mathbb{E}[Q(\boldsymbol{v})] = \boldsymbol{v}$ since $\mathbb{E}\big[[Q(\boldsymbol{v})]_j\big] = \frac{v_j}{p_j} p_j = v_j$

(ii) **Variance bound:** $\mathbb{E}\big[\|Q(\boldsymbol{v}) - \boldsymbol{v}\|_2^2\big] \leq \max_j \frac{1-p_j}{p_j} \|\boldsymbol{v}\|_2^2$

Wang, H., Sievert, S., Liu, S., Charles, Z., Papailiopoulos, D., and Wright, S. Atomo: Communication-efficient Learning via Atomic Sparsification, *NeurIPS 2018*

Deterministic sparsification

D1) Threshold-based rule

For any $\boldsymbol{v} \in \mathbb{R}^d$, denote the sparsified vector $Q(\boldsymbol{v})$.

$$[Q(\boldsymbol{v})]_j = \begin{cases} v_j, & \text{if } |v_j| \geq \gamma, \\ 0, & \text{otherwise,} \end{cases} \qquad \gamma : \text{predefined threshold.}$$

Idea: Only transmit coordinates whose magnitudes exceed τ.

Deterministic sparsification

D2) Memory-based threshold rule

If the algorithm transmits:

$$\begin{aligned} \text{Original:} \quad & \boldsymbol{v}^{(0)},\ \boldsymbol{v}^{(1)},\ \ldots,\ \boldsymbol{v}^{(K)} \\ \text{Sparsified:} \quad & Q(\boldsymbol{v}^{(0)}),\ Q(\boldsymbol{v}^{(1)}),\ \ldots,\ Q(\boldsymbol{v}^{(K)}) \end{aligned}$$

Initialize: $\tilde{\boldsymbol{v}}^{(0)} = \boldsymbol{v}^{(0)}$.

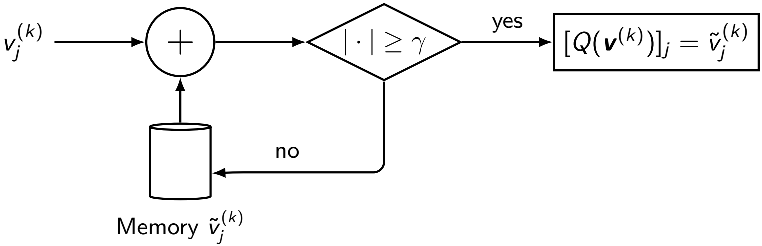
For $k = 0, 1, \ldots, K-1$:

$$[Q(\boldsymbol{v}^{(k)})]_j = \begin{cases} \tilde{v}_j^{(k)}, & \text{if } \left|\tilde{v}_j^{(k)}\right| \geq \gamma, \\ 0, & \text{otherwise.} \end{cases}$$

$$\tilde{\boldsymbol{v}}^{(k+1)} = \boldsymbol{v}^{(k+1)} + \left(\tilde{\boldsymbol{v}}^{(k)} - Q(\boldsymbol{v}^{(k)})\right).$$

EndFor

Deterministic sparsification



For $k = 0, 1, \ldots, K-1$:

$$[Q(\boldsymbol{v}^{(k)})]_j = \begin{cases} \tilde{v}_j^{(k)}, & \text{if } \left|\tilde{v}_j^{(k)}\right| \geq \gamma, \\ 0, & \text{otherwise.} \end{cases}$$

$$\tilde{\boldsymbol{v}}^{(k+1)} = \boldsymbol{v}^{(k+1)} + \left(\tilde{\boldsymbol{v}}^{(k)} - Q(\boldsymbol{v}^{(k)})\right).$$

EndFor

Deterministic sparsification

D3) Top-k sparsification rule*

Consider $\boldsymbol{\pi} \in \mathbb{R}^d$ as a permutation of $\{1, 2, \ldots, d\}$ such that for $\boldsymbol{v} \in \mathbb{R}^d$,

$$|v_{\pi(1)}| \geq |v_{\pi(2)}| \geq \cdots \geq |v_{\pi(d)}|.$$

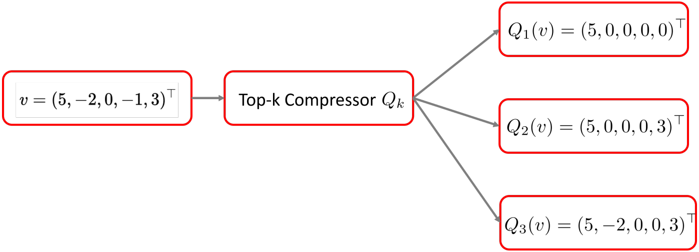
Then the j-th entry of the sparsified vector is:

$$[Q_k(\boldsymbol{v})]_j = \begin{cases} v_j, & \text{if } j = \pi(j) \text{ and } j \leq k, \\ 0, & \text{otherwise.} \end{cases}$$

Stich, S.U., Cordonnier, J.-B., and Jaggi, M. Sparsified SGD with Memory, *NeurIPS 2018*

Deterministic sparsification

D3) Top-k sparsification rule*



The error due to sparsification:

$$\|Q_k(\mathbf{v}) - \mathbf{v}\|_2^2 \le \left(1 - \frac{k}{d}\right)\|\mathbf{v}\|_2^2.$$

Stich, S.U., Cordonnier, J.-B., and Jaggi, M. Sparsified SGD with Memory, *NeurIPS 2018*

Implementation of quantized / sparsified gradient descent

For iteration $k = 1, 2, \ldots, K$:

1. **Server broadcasts** the current model parameter $\boldsymbol{x}^k$ to all workers.
2. **For each worker** $i = 1, 2, \ldots, n$ (in parallel):
   - Worker i calculates $\boldsymbol{v}_i^{(k)} = \nabla F_i(\boldsymbol{x}^k)$.
   - Worker i computes sparsified/quantized gradient $Q(\boldsymbol{v}_i^{(k)})$.
   - Worker i uploads $Q(\boldsymbol{v}_i^{(k)})$ to the server.
3. **Server updates** the global model:

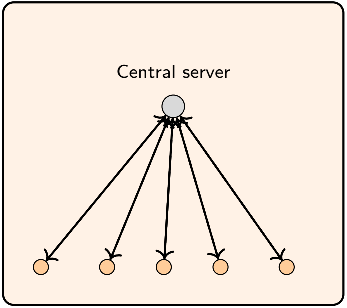
$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \frac{\alpha}{n} \sum_{i=1}^{n} Q\left(\boldsymbol{v}_i^{(k)}\right).$$

Remark: Quantization / sparsification can be performed either at the server side or at the worker side or both.

Table of Contents

Reduce the number of workers

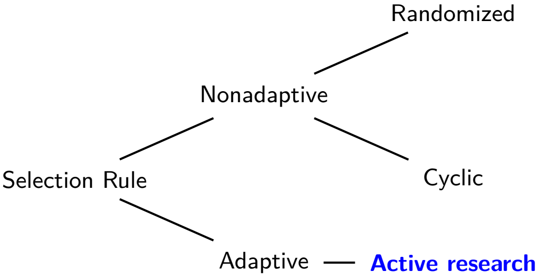
Goal: Reduce the number of workers participating in communication.



Local data on nodes

Methods for worker selection

Idea: Only a subset of workers upload/download gradients at each round, based on either fixed (nonadaptive) or dynamic (adaptive) rules.



Non-adaptive randomized rule

For iteration $k = 1, 2, \ldots, K$:

S1) Server randomly selects worker $i_k \in \{1, \ldots, n\}$ (or a set $\mathcal{I}_k \subseteq \{1, \ldots, n\}$), and sends $\boldsymbol{x}^k$ to worker i_k (or all $i \in \mathcal{I}_k$).

1. Worker i_k computes and uploads $\nabla F_{i_k}(\boldsymbol{x}^k)$.
2. **Server updates $\boldsymbol{x}^k$ via:**

   Option I (SGD):
   $$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha \nabla F_{i_k}(\boldsymbol{x}^k).$$

   Option II (Randomized Incremental Aggregated Gradient (RIAG)):
   $$\boldsymbol{x}_i^{k+1} = \begin{cases} \boldsymbol{x}^k, & i = i_k, \\ \boldsymbol{x}_i^k, & i \neq i_k, \end{cases} \qquad \boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha \nabla F_{i_k}(\boldsymbol{x}^k) - \alpha \sum_{i \neq i_k} \nabla F_i(\boldsymbol{x}_i^k).$$

Memory overhead for RIAG

If the server pursues Option II, it stores a table $\in \mathbb{R}^{d\times n}$.

∇F_1	∇F_2	∇F_3	$\cdots$	∇F_n

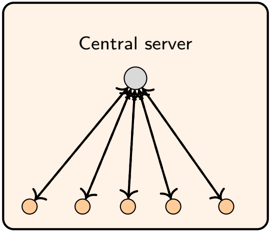
Overcome memory overhead:

Store the summation $\nabla_k = \sum_{i=1}^n \nabla F_i(\boldsymbol{x}_i^k)$.
Worker uploads only the change of gradients:

$$\nabla_k^i = \nabla F_i(\boldsymbol{x}^k) - \nabla F_i(\boldsymbol{x}_i^k).$$

Server updates the summation via:

$$\nabla_{k+1} = \nabla_k + \nabla_k^i.$$



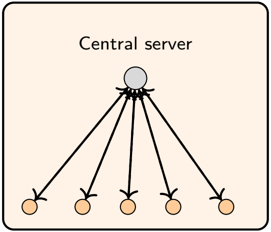
Local data on nodes

Non-adaptive cyclic rule

For $k = 1, 2, \ldots, K$:

1. Server selects worker $i_k = k \bmod n$.
2. Server sends $\boldsymbol{x}^k$ to worker i_k.
3. Worker i_k computes and uploads $\nabla F_{i_k}(\boldsymbol{x}^k)$.
4. Server updates $\boldsymbol{x}^k$ via Option I or II.

CIAG: Cyclic Incremental Aggregated Gradient



Local data on nodes

Theoretical guarantees of CIAG

Theorem 3 (Convergence of CIAG)

Under the L-smooth and μ-strongly convex assumption, if the stepsize α in CIAG satisfies:

$$0 < \alpha \leq \frac{1}{n(\mu + L)},$$

then CIAG achieves an **R-linear convergence rate:**

$$\|\boldsymbol{x}^k - \boldsymbol{x}^*\|_2^2 \leq \rho^k \|\boldsymbol{x}^0 - \boldsymbol{x}^*\|_2^2, \quad \text{for some } 0 < \rho < 1.$$

Plan: adaptive worker selection

Compare: Gradient Descent vs. RIAG/CIAG

Tradeoff Factors:

(c1) Amount of communication per iteration

(c2) Number of iterations required for convergence

Observation:

RIAG/CIAG $\approx \frac{1}{n}$ communications as GD (fewer uploads per iteration),

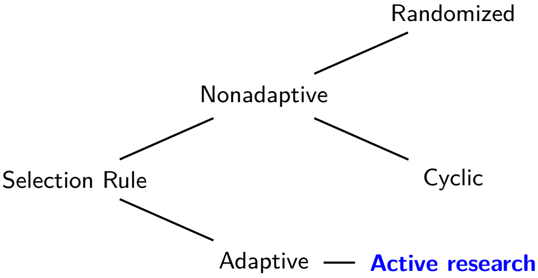
GD $\approx \frac{1}{n}$ iterations as RIAG/CIAG (faster convergence per round).

Total Communication Cost:

$$\text{Total communication rounds} = (c1) \times (c2).$$

Methods for worker selection

Idea: Only a subset of workers upload/download gradients at each round, based on either fixed (nonadaptive) or dynamic (adaptive) rules.



Adaptive worker selection - best tradeoff

A slight generalization of Incremental Aggregated Gradient (IAG):

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha \sum_{i \in \mathcal{I}^k} \nabla F_i(\boldsymbol{x}^k) - \alpha \sum_{i \notin \mathcal{I}^k} \nabla F_i(\boldsymbol{x}_i^k),$$

Special cases:

- **RIAG (Randomized IAG):** $\mathcal{I}^k = \{i_k\}$, with i_k randomly generated.
- **CIAG (Cyclic IAG):** $\mathcal{I}^k = \{k \bmod n\}$.
- **GD (Full Gradient Descent):** $\mathcal{I}^k = \{1, 2, \ldots, n\}$.

Incremental aggregated gradient

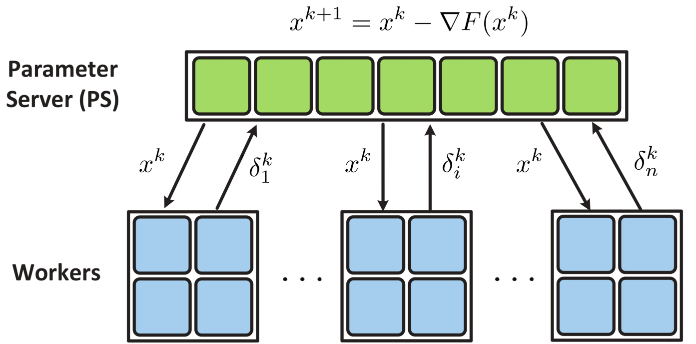
$$\begin{aligned}
\boldsymbol{x}^{k+1} &= \boldsymbol{x}^k - \alpha \sum_{i \in \mathcal{I}^k} \nabla F_i(\boldsymbol{x}^k) - \alpha \sum_{i \notin \mathcal{I}^k} \nabla F_i(\boldsymbol{x}_i^k) \\
&= \underbrace{\boldsymbol{x}^k - \alpha \sum_{i=1}^{n} \nabla F_i(\boldsymbol{x}^k)}_{\text{GD update}} + \alpha \underbrace{\sum_{i \notin \mathcal{I}^k} \Big(\nabla F_i(\boldsymbol{x}^k) - \nabla F_i(\boldsymbol{x}_i^k) \Big)}_{\boldsymbol{\delta}_i^k}
\end{aligned}$$

Error of using old gradients: $\boldsymbol{\delta}_i^k$

Intuition: If $\|\boldsymbol{\delta}_i^k\|$ are small relative to $\sum_{i=1}^n \|\nabla F_i(\boldsymbol{x}^k)\|$, then the price paid for saving uploads/downloads is small.

Incremental aggregated gradient

Question: The intuition is good but *how to quantify small?*



Toward adaptive worker selection

Design an *adaptive selection rule* by analyzing the IAG iteration.

Lemma (IAG)

Under the L-smooth assumption of $F(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} F_i(\boldsymbol{x})$, $\boldsymbol{x}^{k+1}$ *is generated by performing one-step generic IAG update given* $\boldsymbol{x}^k$ *and* $\{\boldsymbol{x}_i^k\}_{i=1}^n$, *then:*

$$
\begin{aligned}
F(\boldsymbol{x}^{k+1}) - F(\boldsymbol{x}^k) &\le -\frac{\alpha}{2}\|\nabla F(\boldsymbol{x}^k)\|_2^2 + \frac{\alpha}{2}\Big\|\sum_{i\notin\mathcal{I}^k}\boldsymbol{\delta}_i^k\Big\|_2^2 \\
&\quad + \left(\frac{L}{2} - \frac{1}{2\alpha}\right)\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|_2^2 \\
&\overset{\alpha=\frac{1}{L}}{\Longrightarrow} \le -\frac{1}{2L}\|\nabla F(\boldsymbol{x}^k)\|_2^2 + \frac{1}{2L}\Big\|\sum_{i\notin\mathcal{I}^k}\boldsymbol{\delta}_i^k\Big\|_2^2 \triangleq \Delta_{CIAG}^k.
\end{aligned}
$$

Communication principle

Lemma (GD)

Under the same L-smooth assumption, the one-step GD update satisfies:

$$F(\boldsymbol{x}^{k+1}) - F(\boldsymbol{x}^k) \leq -\frac{1}{2L}\|\nabla F(\boldsymbol{x}^k)\|_2^2 \triangleq \Delta_{GD}^k.$$

Principle: Larger progress per communication:

$$\frac{\Delta_{\mathrm{IAG}}^k}{|\mathcal{I}^k|} \leq \frac{\Delta_{\mathrm{GD}}^k}{n}$$

Plugging Δ_{GD}^k and $\Delta^k{}_{\mathrm{IAG}}$ leads to:

$$\frac{-\frac{1}{2L}\|\nabla F(\boldsymbol{x}^k)\|^2 + \frac{1}{2L}\sum_{i \notin \mathcal{I}^k}\|\boldsymbol{\delta}_i^k\|^2}{|\mathcal{I}^k|} \leq \frac{-\frac{1}{2L}\|\nabla F(\boldsymbol{x}^k)\|^2}{n}$$

Deriving the sufficient condition for the principle

$$\frac{-\frac{1}{2L}\|\nabla F(\boldsymbol{x}^k)\|^2 + \frac{1}{2L}\sum_{i\notin\mathcal{I}^k}\|\boldsymbol{\delta}_i^k\|^2}{|\mathcal{I}^k|} \leq \frac{-\frac{1}{2L}\|\nabla F(\boldsymbol{x}^k)\|^2}{n}$$

$$\iff \Big\|\sum_{i\notin\mathcal{I}^k}\boldsymbol{\delta}_i^k\Big\|^2 \leq \Big(1-\frac{|\mathcal{I}^k|}{n}\Big)\|\nabla F(\boldsymbol{x}^k)\|^2.$$

By Cauchy–Schwarz inequality, $\|\boldsymbol{a}_1+\boldsymbol{a}_2+\cdots+\boldsymbol{a}_n\|^2 \leq n\sum_{i=1}^n\|\boldsymbol{a}_i\|^2$, it holds that:

$$\Big\|\sum_{i\notin\mathcal{I}^k}\boldsymbol{\delta}_i^k\Big\|^2 \leq \Big(n-|\mathcal{I}^k|\Big)\sum_{i\notin\mathcal{I}^k}\|\boldsymbol{\delta}_i^k\|^2 \leq \Big(n-|\mathcal{I}^k|\Big)n\max_{i\notin\mathcal{I}^k}\|\boldsymbol{\delta}_i^k\|^2.$$

Deriving the sufficient condition for progress principle

Sufficient Condition for the Principle:

$$\Big(n - |\mathcal{I}^k|\Big) n \max_{i \notin \mathcal{I}^k} \|\boldsymbol{\delta}_i^k\|^2 \le \frac{n - |\mathcal{I}^k|}{n} \|\nabla F(\boldsymbol{x}^k)\|^2.$$

$$\iff \|\boldsymbol{\delta}_i^k\|^2 \le \frac{1}{\alpha^2 n^2} \|\nabla F(\boldsymbol{x}^k)\|^2, \quad \text{for all } i \in \{1, \dots, n\}.$$

Q: How can we check this condition either at the server or at worker?

$$\|\nabla F(\boldsymbol{x}^k)\|^2 = \Big\| \sum_{i=1}^{n} \nabla F_i(\boldsymbol{x}^k) \Big\|^2$$

This cannot be computed locally.

Checking the sufficient condition

Approximation:

$$\|\nabla F(\boldsymbol{x}^k)\|^2 \approx \frac{1}{\alpha^2}\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|^2$$

so that each worker can check condition locally by:

$$\boxed{\|\boldsymbol{\delta}_i^k\|^2 \le \frac{1}{\alpha^2 n^2}\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|^2 \quad (\textit{Worker side})}$$

Q: What if we find an upper bound on the left-hand side?

$$\|\nabla F_i(\boldsymbol{x}^k) - \nabla F_i(\boldsymbol{x}_i^k)\| \le L_i\|\boldsymbol{x}^k - \boldsymbol{x}_i^k\|$$

A sufficient condition rule is:

$$\boxed{L_i^2\|\boldsymbol{x}^k - \boldsymbol{x}_i^k\|^2 \le \frac{1}{\alpha^2 n^2}\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|^2 \quad (\textit{Server side})}$$

Implementation of adaptive selection rule (LAG)*

Worker side:
For iteration $k = 1, 2, \ldots, K$:

1. **Server broadcasts** the current model parameter $\boldsymbol{x}^k$ to all workers.
2. **For each worker** $i = 1, 2, \ldots, n$ **(in parallel):**
   - Worker i computes the local gradient $\nabla F_i(\boldsymbol{x}^k)$.
   - Worker i checks the upload condition:

$$\|\boldsymbol{\delta}_i^k\|^2 \leq \frac{1}{\alpha^2 n^2} \|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|^2.$$

   - If the condition is satisfied $\Rightarrow$ Do not upload.
   - Otherwise $\Rightarrow$ Upload.
3. **Server updates** the global model via the generic IAG update rule.

Chen, T., Giannakis, G., Sun, T., and Yin, W. LAG: Lazily Aggregated Gradient for Communication-efficient Distributed Learning, *NeurIPS 2018*

Implementation of adaptive selection rule (LAG)*

Server side:
For iteration $k = 1, 2, \ldots, K$:

1. **Server checks** the condition for each worker $i = 1, 2, \ldots, n$:

$$L_i^2 \|\boldsymbol{x}^k - \boldsymbol{x}_i^k\|^2 \leq \frac{1}{\alpha^2 n^2} \|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|^2.$$

2. Collect all violating workers into the set $\mathcal{I}^k$.
3. **Server sends** the current model $\boldsymbol{x}^k$ to all $i \in \mathcal{I}^k$.
4. **For each worker** $i \in \mathcal{I}^k$:
   - Worker i computes and uploads $\nabla F_i(\boldsymbol{x}^k)$ to the server.
5. **Server updates** the global parameter $\boldsymbol{x}^{k+1}$ via the generic IAG update rule.

Chen, T., Giannakis, G., Sun, T., and Yin, W. LAG: Lazily Aggregated Gradient for Communication-efficient Distributed Learning, *NeurIPS 2018*

Theoretical guarantee of LAG

Theorem 4 (Convergence of LAG)

1. Under the L-smooth assumption of $F_i(\boldsymbol{x})$, we have:

$$\frac{1}{K}\sum_{k=1}^{K}\|\nabla F(\boldsymbol{x}^k)\|^2 = \mathcal{O}\left(\frac{1}{K}\right) \qquad \text{(Same as GD)}$$

2. Under the additional convex assumption, we have:

$$F(\boldsymbol{x}^k) - F(\boldsymbol{x}^*) = \mathcal{O}\left(\frac{1}{K}\right) \qquad \text{(Same as GD)}$$

3. Under the additional μ-strong convexity assumption, we have:

$$F(\boldsymbol{x}^k) - F(\boldsymbol{x}^*) = \mathcal{O}\left((1-\tfrac{\mu}{L})^k\right) \qquad \text{(Same as GD)}$$

Empirical performance of LAG

- **Faster convergence per iteration:** LAG achieves similar or faster convergence compared with IAG and GD in terms of iteration complexity.
- **Significantly reduced communication cost:** LAG requires fewer communication rounds while maintaining accuracy.

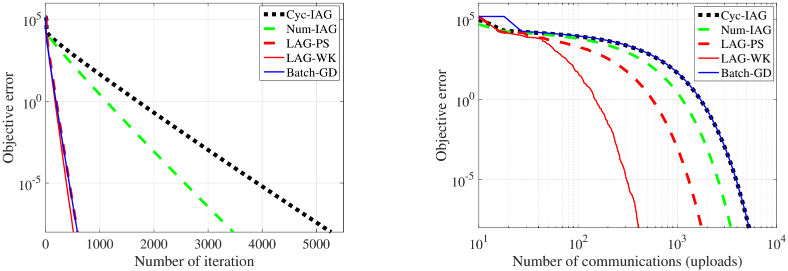


Figure: Iteration and communication complexity for linear regression.

Worker-side condition in LAG (setup)

Goal: Show that Lazy Aggregated Gradient (LAG) still guarantees descent even when workers communicate intermittently.

Worker update condition:

$$\|\boldsymbol{\delta}_i^k\|^2 \leq \frac{\zeta}{\alpha^2 n^2}\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|^2,$$

where $\boldsymbol{\delta}_i^k = \nabla F_i(\boldsymbol{x}_i^k) - \nabla F_i(\boldsymbol{x}^k)$.

Intuition:

- $\boldsymbol{\delta}_i^k$ measures how “stale" a worker's gradient is.
- Larger $\zeta \Rightarrow$ easier condition $\Rightarrow$ fewer communications.
- When local changes are small, workers can skip communication.

Step 1: One-step progress of LAG

Under L-smoothness of F, the descent lemma gives:

$$F(\boldsymbol{x}^{k+1}) - F(\boldsymbol{x}^k) \leq \langle \nabla F(\boldsymbol{x}^k), \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \rangle + \frac{L}{2} \|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2.$$

Substitute the LAG update rule:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha \sum_{i \in \mathcal{I}^k} \nabla F_i(\boldsymbol{x}^k) - \alpha \sum_{i \notin \mathcal{I}^k} \nabla F_i(\boldsymbol{x}_i^k).$$

This introduces gradient mismatch terms $\boldsymbol{\delta}_i^k$ from lazy workers.

Bounding the inner product term (sketch)

Plugging the update into the inner product:

$$\left\langle \nabla F(\boldsymbol{x}^k),\, \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \right\rangle = -\alpha \|\nabla F(\boldsymbol{x}^k)\|^2 - \alpha \Big\langle \nabla F(\boldsymbol{x}^k),\, \sum_{i \notin \mathcal{I}^k} \boldsymbol{\delta}_i^k \Big\rangle.$$

Using the identity $2a^\top b = \|a\|^2 + \|b\|^2 - \|a - b\|^2$, we bound the cross term:

$$\Big\langle \nabla F(\boldsymbol{x}^k), \sum_{i \notin \mathcal{I}^k} \boldsymbol{\delta}_i^k \Big\rangle \le \frac{1}{2}\|\nabla F(\boldsymbol{x}^k)\|^2 + \frac{1}{2}\Big\| \sum_{i \notin \mathcal{I}^k} \boldsymbol{\delta}_i^k \Big\|^2.$$

Result:

$$F(\boldsymbol{x}^{k+1}) - F(\boldsymbol{x}^k) \le -\frac{\alpha}{2}\|\nabla F(\boldsymbol{x}^k)\|^2 + \frac{\alpha}{2} \sum_{i \notin \mathcal{I}^k} \|\boldsymbol{\delta}_i^k\|^2 + \frac{L}{2}\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2.$$

Step 2: Apply worker-side condition

Applying the worker condition to bound the error term gives:

$$F(\boldsymbol{x}^{k+1}) - F(\boldsymbol{x}^k) \leq -\frac{\alpha}{2}\|\nabla F(\boldsymbol{x}^k)\|^2 + \frac{\zeta}{2\alpha \boldsymbol{n}}\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|^2.$$

Takeaway:

- The first term guarantees descent.
- The second term captures accumulated gradient staleness.
- Properly tuning ζ keeps LAG stable and communication-efficient.

Understanding the tradeoff

Key tradeoff:

Descent rate vs. Error from delayed gradients.

If ζ is too small $\Rightarrow$ frequent communication but stable. If ζ is too large $\Rightarrow$ fewer communications but possible instability.

Balance point: choose $\zeta = 1 - \alpha L$ so that:

Error term $\sim$ Descent term.

This ensures that each step still decreases $F(\boldsymbol{x})$ in expectation.

Step 3: Convergence sketch

Choose: $\zeta = 1 - \alpha L$ to balance descent and error.

Telescoping over $k = 1, \ldots, K$**:**

$$\frac{1}{K}\sum_{k=1}^{K} \|\nabla F(\boldsymbol{x}^k)\|^2 \leq \frac{F(\boldsymbol{x}^1) - F(\boldsymbol{x}^*)}{\alpha K}.$$

Result:

$$\boxed{\text{LAG achieves } \mathcal{O}(1/K) \text{ convergence}}$$

- the same as parallel GD, but with significantly fewer communications.

(Idea: smoothness $\Rightarrow$ gradients evolve slowly $\Rightarrow$ lazy communication.)

Recap and fine-tuning

- What we have talked about today?

  ⇒ **Sparsification** reduces communication cost by transmitting gradients with fewer entries.

  ⇒ **Worker selection** reduces communication cost by letting only a subset of workers upload gradients adaptively.

Welcome anonymous survey!