Asynchronous Distributed Machine Learning

Hagit Attiya and Noa Schiller (Technion)
Distributed Optimization

Processes access the same data distribution $D$ and loss function $\ell: \mathbb{R}^d \times D \to \mathbb{R}$

Cost function at $x \in \mathbb{R}^d$ is $Q(x) \triangleq \mathbb{E}_{z \sim D}[\ell(x, z)]$

Minimize $Q$ to find $x^* \in \arg\min_{x \in \mathbb{R}^d} Q(x)$

$Q$ is differentiable and smooth

Can be either (strongly) convex or non-convex
Stochastic Gradient Descent (SGD)

Stochastic gradient $G(x, z)$ estimates the true gradient $\nabla Q(x)$ using a random data point $z \in D$

For $t = 1, 2, ...$
1. Draw a random data point $z \sim i.i.d. D$
2. Update $x_{t+1} = x_t - \eta_t G(x_t, z)$

Estimates are unbiased: $\mathbb{E}_{z \sim D}[G(x, z)] = \nabla Q(x)$

with bounded variance: $\mathbb{E}_{z \sim D}[\|G(x, z) - \nabla Q(x)\|_2] \leq \sigma$
Mini-Batch SGD

Faster convergence by sampling several data points

For \( t = 1, 2, \ldots \)
1. Draw \( M \) random data points \( z_1, \ldots, z_M \) \( i.i.d \sim D \)
2. Update \( x_{t+1} = x_t - \frac{\eta_t}{M} \sum_{i=1}^{M} G(x_t, z_i) \)
Simple Distributed Mini-Batch SGD

Centralized scheme requires synchronization
Parameter server is a single point-of-failure

For $t = 1, 2, ...$

1. Draw $M$ random data points $z_1, ..., z_M \sim D$
2. Update $x_{t+1} = x_t - \frac{\eta_t}{M} \sum_{i=1}^{M} G(x_t, z_i)$
Decentralized SGD

Fully-connected set of $n$ nodes

For $t = 1, 2, ...$
1. A node draws a random data point $i.i.d \sim D$
   & computes new gradient
2. Get gradients from $M$ nodes & update

How good is this?
**Strongly Convex Q**

\[ \Rightarrow \text{“External” Convergence} \]

Single minimum \( x^* \) obtained at a unique point

For any \( M \), any round \( T \), and any node \( i \)

\[
\mathbb{E} \left[ \| x^i_T - x^* \|_2^2 \right] \leq O \left( \frac{\| x^1 - x^* \|_2^2}{MT} + \frac{\sigma^2}{MT} \right)
\]

Same convergence rate as sequential mini-batch SGD, with batch size \( M \)

Implies external convergence \[ \mathbb{E} \left[ \| x^i - x^* \|_2^2 \right] \leq \varepsilon \]
Strongly Convex Q
\( \Rightarrow \) “External” Convergence

Single minimum \( x^* \) obtained at a unique point

For any \( M \) \( \Rightarrow \) the algorithm withstands partitioning

I.e., converges even when communicating with only a minority

No “split-brain”
Non Strongly Convex Functions Require a Majority

For some non-convex cost function $Q$, algorithm does not converge if a majority of processes fail

$$\max_{i,j} \mathbb{E} \left[ \| \text{output}^i - \text{output}^j \|_2 \right] > \delta$$

(No internal convergence)

Proof more complicated than expected and relies on probabilistic indistinguishability

[Goren, Moses & Spiegelman, DISC 2021]
General Algorithm

Convergence rate similar to sequential algorithm

[Ghadimi,Lan, 2013]

Analysis is a simplified version of

[ElMhamdi,Farhadkhani,Guerraoui,Guirguis,Hoang,Rouault,NeuroIPS 2021]

For iteration $t = 1, \ldots, T$:
1. Compute the stochastic gradient $g_t^i$ at $x_t^i$
2. $x_t^i \leftarrow x_t^i - \eta_t \cdot g_t^i$
3. Send $\langle t, x_t^i \rangle$
4. Wait to receive $\geq M$ iteration-$t$ messages
5. $x_{t+1}^i \leftarrow \text{Avg}(\text{received models})$
But with an additive factor of $\Delta$, which is reduced using multi-dimensional approximate agreement

This algorithm needs communication with a majority

For iteration $t = 1, \ldots, T$:
1. Compute the stochastic gradient $g_t^i$ at $x_t^i$
2. $x_t^i \leftarrow x_t^i - \eta_t \cdot g_t^i$
3. Send $\langle t, x_t^i \rangle$
4. Wait to receive $\geq M$ iteration-$t$ messages
5. $x_{t+1}^i \leftarrow \text{Avg}(\text{received models})$
Multi-Dimensional Approximate Agreement

A process starts with input $x^i \in \mathbb{R}^d$ and returns $y^i \in \mathbb{R}^d$, such that the outputs are

- In the convex hull of the inputs
- Contracted by a factor of $q$ relative to the inputs

\[ \max_{i,j} \| y^i - y^j \|_2^2 \leq q \max_{i,j} \| x^i - x^j \|_2^2 \]

[Mendes, Herlihy, Vaidya, Garg, DC 2015]
[Fugger, Nowak, DISC 2018]
Convergence of General Algorithm

With an appropriate $q$, we get internal convergence

$$\max_{i,j} E\|x^i - x^j\|_2 < \delta$$

Can also show external convergence

Better (1-dimension) AA when shared-memory is used

[A, Kumari, Schiller, OPODIS 2020]
Cluster-Based Model

- Disjoint clusters, each with shared read/write registers
- All processes can send asynchronous messages to each other

[Raynal, Cao, ICDCS 2019]
Cluster-Based Model for HPC

- **Cluster 1**: Multiple compute nodes + RDMA
- **Cluster 2**: Multiple compute nodes + RDMA
- **Cluster 3**: Multiple compute nodes + RDMA
- **Cluster m**: Multiple compute nodes + RDMA

Asynchronous message-passing network

Threads of the same process
Multi-Dimensional AA in the Cluster-Based Model

For round $r = 1 \ldots R$:
1. $z_r = \text{ClusterApproximateAgreement}(y_r)$
2. Send $\langle r, z_r \rangle$ to all processes
3. Wait to receive round $r$ messages representing a majority of processes
4. $y_{r+1} = \text{Aggregate}(\text{received values})$
Multi-Dimensional AA in the Cluster-Based Model

A process represents all processes in its cluster

For round $r = 1 \ldots R$:
1. $z_r = \text{ClusterApproximateAgreement}(y_r)$
2. Send $(r, z_r)$ to all processes
3. Wait to receive round $r$ messages representing a majority of processes
4. $y_{r+1} = \text{Aggregate}(\text{received values})$

Better contraction inside a cluster
Tune to get $q$-contraction in $O(\log q)$ rounds
MDAA within a Cluster

An array $\mathbf{A}$ of $\langle \text{value, round}\# \rangle$ for each cluster

$$r \leftarrow 1; \ A[i] \leftarrow \langle x, r \rangle$$

while $r < \text{constant}$ do

   let $r_{\text{max}}$ be the largest round number in $\mathbf{A}$

   if $r = r_{\text{max}}$ then
      $X \leftarrow \text{values in } \mathbf{A} \text{ with round } r_{\text{max}}$
      $A[i] \leftarrow \langle \text{MidExtremes}(X), r+1 \rangle$
      $r \leftarrow r + 1$
   else $r \leftarrow r_{\text{max}}$

return some $x_j$ s.t. $A[j] = \langle x_j, r+1 \rangle$

\textbf{MidExtremes} returns the average of the two values realizing the maximum Euclidean distance
An array $A$ of $\langle$value, round$\rangle$ for each cluster

$r ← 1 ; A[i] ← \langle x, r \rangle$
while $r < \text{constant}$ do
  let $r_{\text{max}}$ be the largest round number in $A$
  if $r = r_{\text{max}}$ then
    $X ←$ values in $A$ with round $r_{\text{max}}$
    $A[i] ← \langle \text{MidExtremes}(X), r+1 \rangle$
    $r ← r + 1$
  else $r ← r_{\text{max}}$
return some $x_j$ s.t. $A[j] = \langle x_j, r+1 \rangle$

Ensures constant contraction within $O(1)$ rounds
Skipping

\[ r \leftarrow 1 ; A[i] \leftarrow \langle x, r \rangle \]
while \( r < \text{constant} \) do
  \[ \text{let } r_{\text{max}} \text{ be the largest round number in } A \]
  if \( r = r_{\text{max}} \) then
    \[ X \leftarrow \text{values in } A \text{ with round } r_{\text{max}} \]
    \[ A[i] \leftarrow \langle \text{MidExtremes}(X), r+1 \rangle \]
    \[ r \leftarrow r + 1 \]
  else \( r \leftarrow r_{\text{max}} \)
return some \( x_j \) s.t. \( A[j] = \langle x_j, r+1 \rangle \)
Skipping

A process can skip to the most advanced iteration instead of going through intermediate iterations

For round $r = 1 \ldots R$:
1. $z_r = \text{ClusterApproximateAgreement}(y_r)$
2. Send $\langle r, z_r \rangle$ to all processes
3. Wait to receive round $r$ message from a majority of clusters
4. $y_{r+1} = \text{AggregationRule}(\text{received values})$
5. If received round $r'$ messages, $r' > r$, then skip to round $r'$
Recovery through Skipping

Allows recovering process to rejoin the computation

Non-volatile memory can be used to checkpoint the current status

For round $r = 1 \ldots R$:

1. $z_r = \text{ClusterApproximateAgreement}(y_r)$
2. Send $\langle r, z_r \rangle$ to all processes
3. Wait to receive round $r$ message from a majority of clusters
4. $y_{r+1} = \text{AggregationRule}(\text{received values})$
5. If received round $r'$ messages, $r' > r$, then skip to round $r'$
Some Related Work

Other work does not ignore (stale) parameters from previous iterations

[Li, Ben-Nun, Di Girolamo, Alistarh, Torsten Hoefler, PPoPP 2020]
[Li, Ben-Nun, Di Girolamo, Dryden, Alistarh, Torsten Hoefler, TPDS 2021]

Elastic consistency bounds the staleness

[Nadiradze, Markov, Chatterjee, Kungurtsev, Alistarh, AAAI 2021]

Our MDAA algorithm “beats” a $f(d + 2)$-redundancy lower bound for Byzantine failures

[Mendes, Herlihy, Vaidya, Garg, DC 2015]

$2f$-redundancy is a necessary and sufficient condition for $f$-resilient Byzantine optimization

[Su, Vaidya, PODC 2016]