Adaptive Communication Strategy in Local-Update SGD

Jianyu Wang and Gauri Joshi

Carnegie Mellon University
ECE Department / Parallel Data Lab
Overview

Stochastic Gradient Descent (SGD) is the backbone of ML.

Many previous works:
- SGD w/ momentum
- AdaGrad, Adam
- SVRG, SAGA

Error-Convergence
Training Progress
Iterations
Overview

Extremely large datasets
[source: ImageNet]

Critical need to parallelize SGD!

Deeper and deeper models
[source: GoogleNet]

System Throughput

- Iterations
- Wall-clock time

Error-Convergence

- Training Progress
- Iterations
Overview

- There are many distributed SGD variants designed to increase system throughput [Local SGD, ICLR 19; Decentralized SGD, NeurIPS 17]
- But they also hurt model quality / slow down the convergence

This work: Method to balance error-runtime trade-off!

\[
\text{Training Progress} \quad \frac{\text{Wall-clock time}}{\text{Iterations}} \quad \times \quad \frac{\text{System Throughput}}{\text{Training Progress}} \quad \times \quad \frac{\text{Error-Convergence}}{\text{Iterations}}
\]
Background of Distributed SGD

Execution pipeline of distributed SGD (fully synchronous SGD):

1. Local stochastic gradients computation

Blue arrows: computation time
Background of Distributed SGD

Execution pipeline of distributed SGD (fully synchronous SGD):

1. Local stochastic gradients computation
2. Average gradients across all nodes
Background of Distributed SGD

Execution pipeline of distributed SGD (fully synchronous SGD):

1. Local stochastic gradients computation
2. Average gradients across all nodes
3. Repeat the above steps until convergence
Background of Distributed SGD

Execution pipeline of distributed SGD (fully synchronous SGD):

1. Local stochastic gradients computation
2. Average gradients across all nodes
3. Repeat the above steps until convergence
Communication is the Bottleneck

The communication time can be \textbf{even longer} than computation time. \textit{(No speedup at all!)}
A Promising Solution
Periodic Averaging SGD (PASGD)

Performing $\tau$ local steps before communication

(Special case of Federated Averaging)
A Promising Solution
Periodic Averaging SGD (PASGD)

Performing $\tau$ local steps before communication

✔ Communication delay is reduced by $\tau$ times
✔ Variations in computation delay are evened out

2 Benefits

Wall-clock time

Node 1

Node 2

... ...

Node m

Previous 6 iter.
Benefit of Periodic Averaging

Higher throughput

- **Computation time** $Y$: exponential random variable with mean 1
- **Communication delay** $D$: constant value, equal to 1
- **# nodes** $m = 16$

$Y \sim \exp(1), D = 1$

Much lighter tail

Higher throughput!
Limitation of Periodic Averaging

Worse convergence

Model parameter space

Larger communication period $\tau$, larger deviation between local models!

$\tau = 3$ local steps at each worker
Limitation of Periodic Averaging

Worse convergence

Theoretical Support of worse error-convergence:

\[
\text{Optimization error} \rightarrow \frac{\eta L \sigma^2}{m} + \eta^2 L^2 (\tau - 1) \sigma^2
\]

- \(\eta\) : Learning rate
- \(L\) : Lipschitz constant:
- \(\sigma^2\) : Stochastic gradient variance:

Higher optimization error!
Error-Runtime Trade-off

Large $\tau$ or sparse averaging reduces communication delay
Model discrepancies gives inferior error-convergence

Training Progress = \frac{\text{Training Progress}}{\text{Iterations}} \times \frac{\text{Iterations}}{\text{Wall-clock time}}

Error-Convergence

System Throughput

Large comm. period
Small comm. period

$Y \sim \exp(1), D = 1$
**Error-Runtime Trade-off**

**Key Observation**
Best communication period varies over time

**Goal**
Design an adaptive communication strategy

**Approach**
Use the error-runtime analysis to decide how to switch
Overview of Proposed Scheme

For brevity, first consider fixed learning rate

**Step 1:** Divide the training process into multiple intervals

**Step 2:** estimate the best communication period for each interval

![Diagram showing training loss over Wall clock time with intervals marked as $T_0$, $2T_0$, $lT_0$ and communication periods marked as $\tau_0$, $\tau_1$, $\tau_2$, $\tau_l$.](image-url)
Key Ingredient #1: Estimate the Best \( \tau \)

Combine runtime and convergence analyses

\[
\mathbb{E} \left[ \frac{1}{K} \sum_{k=1}^{K} \left\| \nabla F(\overline{x}_k) \right\|^2 \right] \leq \frac{2 [F(\overline{x}_1) - F_{\text{inf}}]}{\eta K} + \frac{\eta L \sigma^2}{m} + \eta^2 L^2 (\tau - 1) \sigma^2
\]

Optimization error

Replace iteration index by wall-clock time
**Key Ingredient #1: Estimate the Best $\tau$**

Combine runtime and convergence analyses

\[
\mathbb{E} \left[ \frac{1}{K} \sum_{k=1}^{K} \| \nabla F(\overline{x}_k) \|^2 \right] \leq \frac{2 [F(\overline{x}_1) - F_{\text{inf}}]}{\eta K} + \frac{\eta L \sigma^2}{m} + \eta^2 L^2 (\tau - 1) \sigma^2
\]

Optimization error

Replace iteration index by wall-clock time

\[
K = \frac{T}{Y + \frac{D}{\tau}}
\]

Runtime / iteration

Optimization error after time $T$

\[
\leq \frac{2 [F(\overline{x}_1) - F_{\text{inf}}]}{\eta T} \left( Y + \frac{D}{\tau} \right) + \frac{\eta L \sigma^2}{m} + \eta^2 L^2 (\tau - 1) \sigma^2
\]
Key Ingredient #1: Estimate the Best $\tau$

Combine runtime and convergence analyses

Optimization error after time $T$

\[ \leq \frac{2 [F(x_1) - F_{\text{inf}}]}{\eta T} \left( Y + \frac{D}{\tau} \right) + \frac{\eta L \sigma^2}{m} + \eta^2 L^2 (\tau - 1) \sigma^2 \]

A heuristic choice of $\tau$ is to take the derivative and set it to 0

\[ \tau^* = \sqrt{\frac{2(F(x_1) - F_{\text{inf}})D}{\eta^3 L^2 \sigma^2 T}} \]

AdaComm Strategy

Cannot use in practice due to unknown constants!
Key Ingredient #2: update rule

How to use in practice?

The only difference is loss value

\[ \tau_0^* = \sqrt{\frac{2(F(\bar{x}_{t=0}) - F_{\text{inf}})D}{\eta^3 L^2 \sigma^2 T_0}} \]

\[ \tau_l^* = \sqrt{\frac{2(F(\bar{x}_{t=lT_0}) - F_{\text{inf}})D}{\eta^3 L^2 \sigma^2 T_0}} \]

Training loss

Wall clock time

Large comm. period

Small comm. period
Key Ingredient #2: update rule

How to use in practice?

Compute the ratio

\[
\frac{\tau_l^*}{\tau_0^*} = \sqrt{\frac{F(\bar{x}_{t=lT_0}) - F_{inf}}{F(\bar{x}_{t=0}) - F_{inf}}} \approx \sqrt{\frac{F(\bar{x}_{t=lT_0})}{F(\bar{x}_{t=0})}}
\]

Estimated via a grid search

Training loss

Wall clock time
Key Ingredient #2: update rule

How to use in practice?

Update rule in practice

\[
\tau_l = \left[ \sqrt{\frac{F(x_{t=lT_0})}{F(x_{t=0})}} \right] \tau_0
\]

Observation: \( \tau \) gradually decreases

Match well with the intuition
Connection to Learning Rate Decay

Same trade-off, same motivation

- Stage-wise decay
- Exponential decay
- AdaGrad
- Adam
- Etc..

- AdaComm
Connection to Learning Rate Decay

Same trade-off, same motivation

Insight: Decaying comm. period is similar to decaying LR in spirit!
Refinement: Variable learning rate

So far, only consider fixed learning rate

Problem: It’s common practice in DL to use stage-wise decayed learning rate

Solution: rescale $\tau$ when learning rate changes

$$
\tau_l = \sqrt{\frac{\eta_0}{\eta_l}} \frac{F(x_t=lT_0)}{F(x_t=0)} \tau_0
$$
Results on CIFAR-10 with 8 Nodes

**Communication** Intense task

**Computation** Intense task

Test accuracy: **92.52, 91.85, 91.15, 92.72**

**91.93, 91.15, 90.46, 91.77**
Conclusion & Future Directions

Adaptive communication strategy can work well in practice

- achieving up to **3 times** speedup over synchronous SGD
Conclusion & Future Directions

Adaptive communication strategy can work well in practice

- achieving up to 3 times speedup over synchronous SGD

Conceptually similar to adaptive learning rate

- AdaGrad
- Adam
- Stage-wise decay
- Exponential decay
- Etc..

Adaptive communication strategy can work well in practice

- achieving up to 3 times speedup over synchronous SGD

Conceptually similar to adaptive learning rate

- AdaGrad
- Adam
- Stage-wise decay
- Exponential decay
- Etc..
Conclusion & Future Directions

Adaptive communication strategy can work well in practice

- achieving up to 3 times speedup over synchronous SGD

Conceptually similar to adaptive learning rate

Similar trade-off also exists in other distributed SGD variants

- Decentralized parallel SGD: control connectivity of worker network
- K-synchronous SGD: control #nodes to wait at each iteration
Thanks for attention!

Questions?