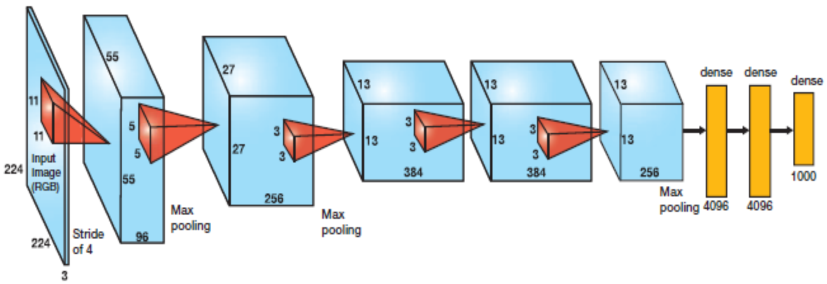
Slow and Stale Gradients Can Win the Race: Error-Runtime Trade-offs in Distributed SGD

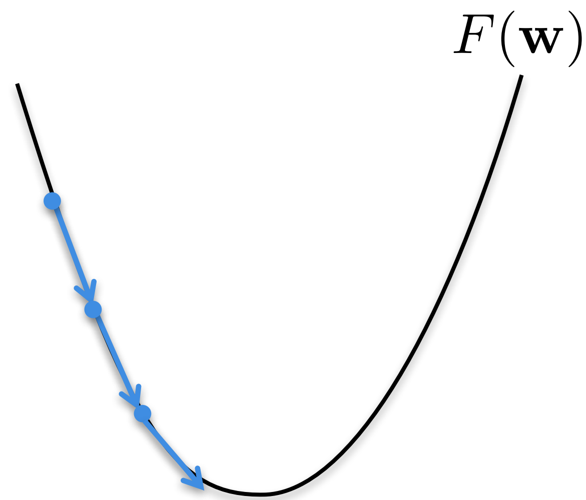
Sanghamitra Dutta, Gauri Joshi (Carnegie Mellon)

Soumyadip Ghosh, Parijat Dube, Priya Nagpurkar (IBM Research)

22th Oct 2018

Stochastic Gradient Descent is the backbone of ML





$$\mathbf{w_{n+1}} = \mathbf{w_n} - \eta \nabla F(\mathbf{w})$$

Speeding Up SGD convergence
is of critical importance!

Accelerating single-node SGD convergence

$$\mathbf{w}_{j+1} = \mathbf{w}_j - \frac{\eta}{m} \sum_{n=1}^{m} \nabla f(\mathbf{w}_j, \xi_n)$$

Learning Rate Schedules: AdaGrad, Adam

Momentum Methods: Polyak, Nesterov

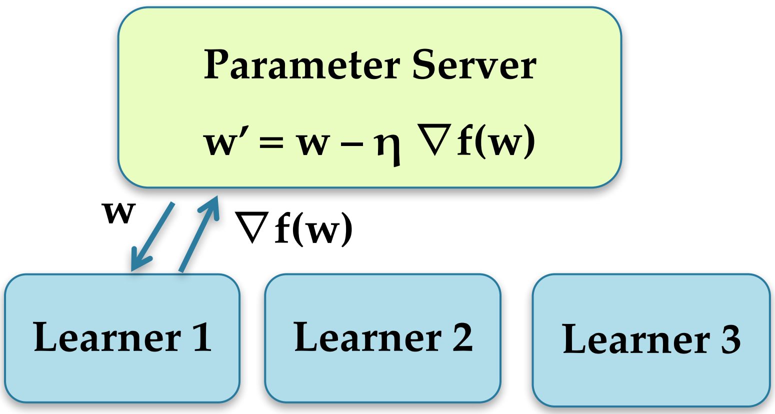
Variance Reduction Methods

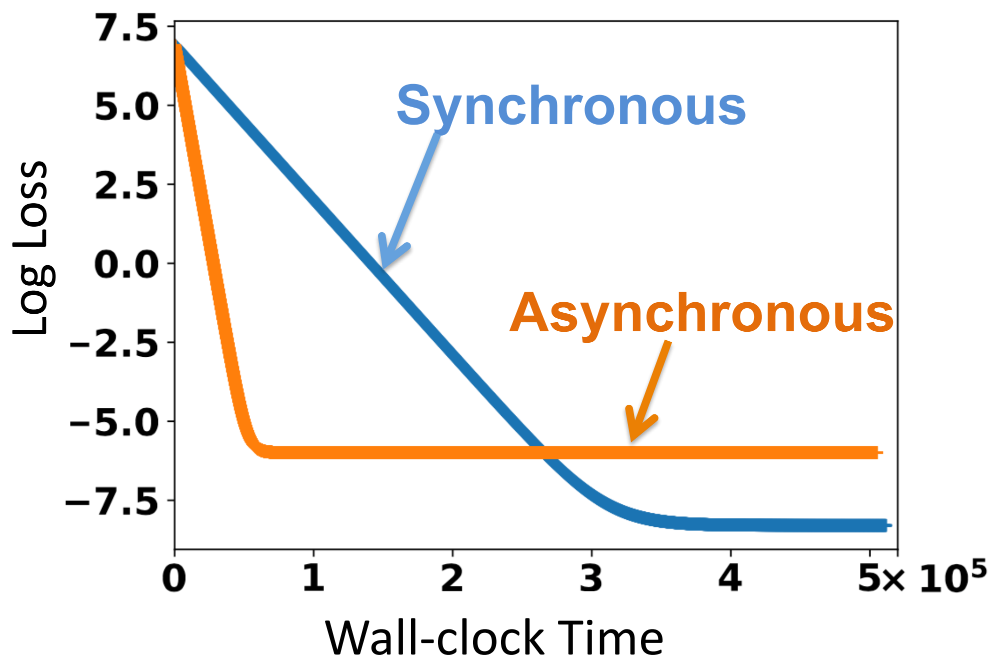
Second-Order Hessian Methods

For large training datasets single-node SGD can be prohibitively slow…

IMAGENET

This Work:
Speeding Up Distributed SGD via Scheduling + Algorithmic Techniques

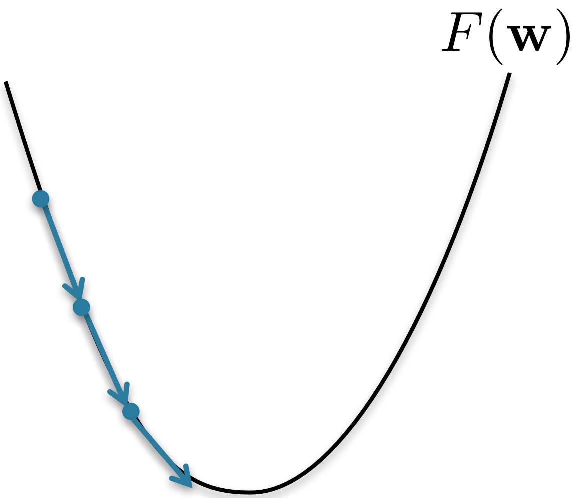




Key Issues

- Straggling Learners
- Gradient Staleness

Batch Gradient Descent



F(w) is the empirical risk function

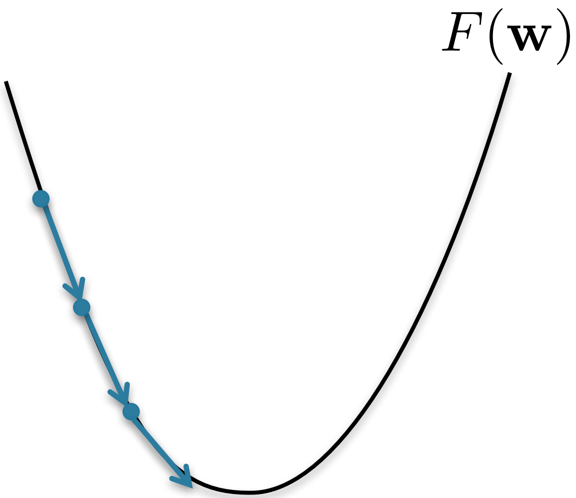
$$\min_{\mathbf{w}} \left\{ F(\mathbf{w}) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{w}, \xi_n) \right\}$$

ξ_n is the n-th labeled sample

$$\mathbf{w}_{j+1} = \mathbf{w}_j - \frac{\eta}{N} \sum_{i=1}^{N} \nabla f(\mathbf{w}_j, \xi_i)$$

Too expensive for large datasets

Stochastic Gradient Descent



F(w) is a function of the training dataset

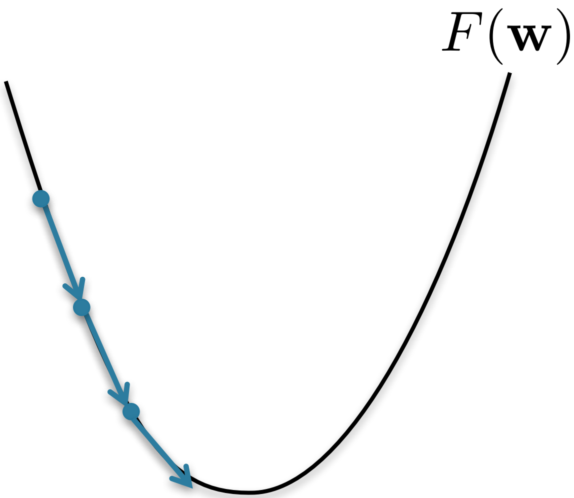
$$\min_{\mathbf{w}} \left\{ F(\mathbf{w}) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{w}, \xi_n) \right\}$$

ξ_n is the n-th labeled sample

$$\mathbf{w}_{j+1} = \mathbf{w}_j - \eta \nabla f(\mathbf{w}_j, \xi_n)$$

Stochastic gradient may be too noisy

Mini-batch SGD



F(w) is the empirical risk function

$$\min_{\mathbf{w}} \left\{ F(\mathbf{w}) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{w}, \xi_n) \right\}$$

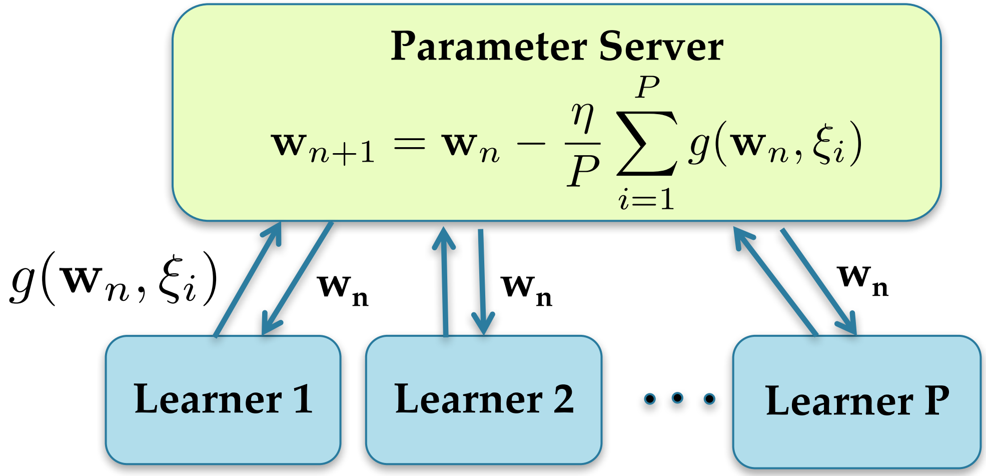
ξ_n is the n-th labeled sample

$$\mathbf{w}_{j+1} = \mathbf{w}_j - \frac{\eta}{m} \sum_{i=1}^{m} \nabla f(\mathbf{w}_j, \xi_i)$$

Noisier, but computationally tractable

For large training datasets single-node SGD can be prohibitively slow…

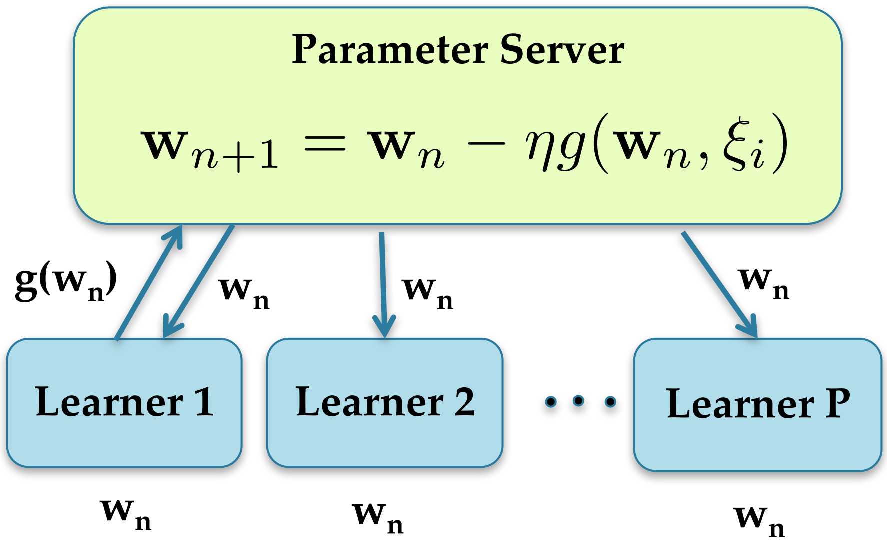
Parameter Server Model: Synchronous SGD



Can process a P-times larger mini-batch in each iteration

Bottlenecked by one or more slow learners

Parameter Server Model: Asynchronous SGD

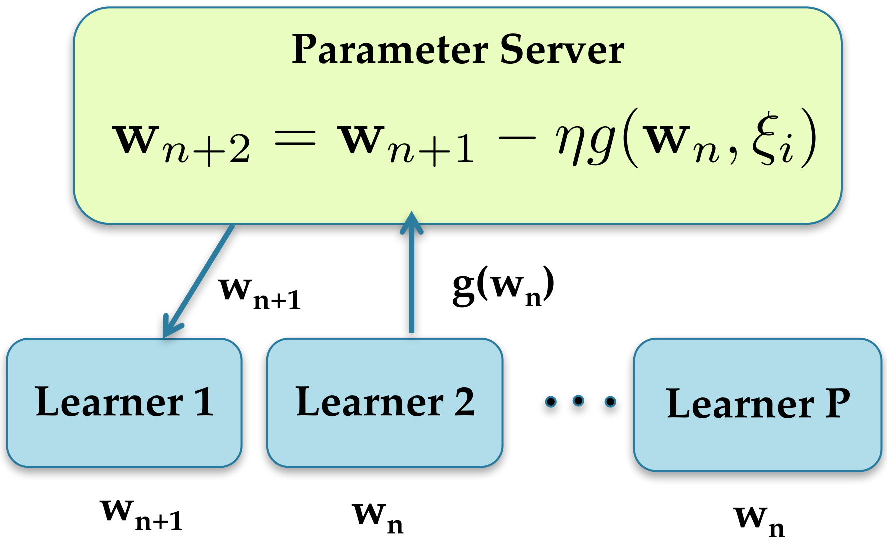


[Recht 2011, Dean 2012, Cipar 2013 …]

Don't have to wait for straggling learners

Gradient Staleness can increase error

Parameter Server Model: Asynchronous SGD

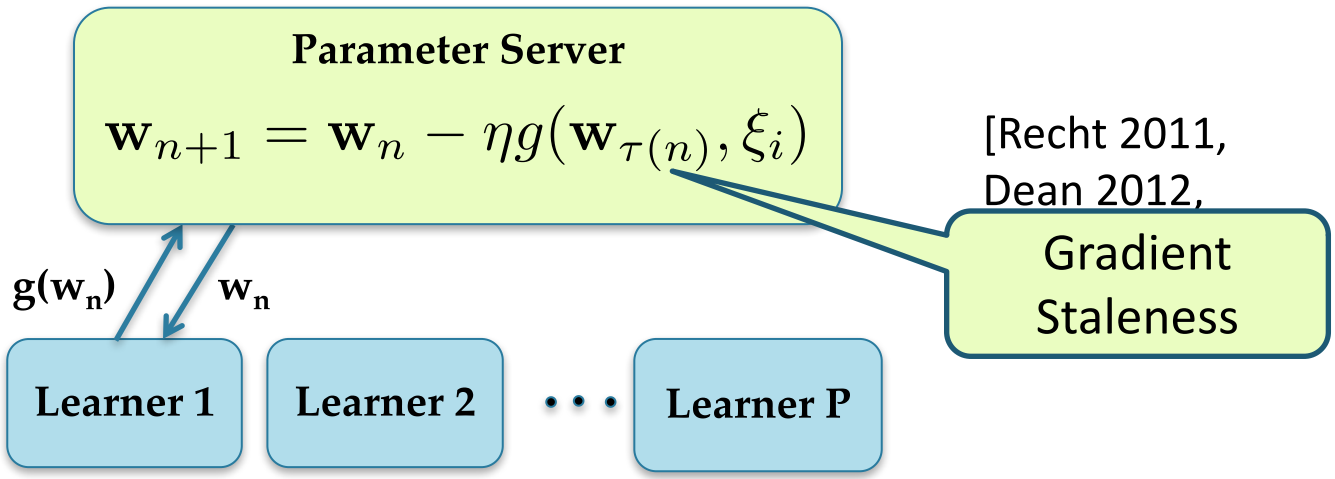


[Recht 2011, Dean 2012, Cipar 2013 …]

Don’t have to wait for straggling learners

Gradient Staleness can increase error

Parameter Server Model: Asynchronous SGD



Don't have to wait for straggling learners

Gradient Staleness can increase error

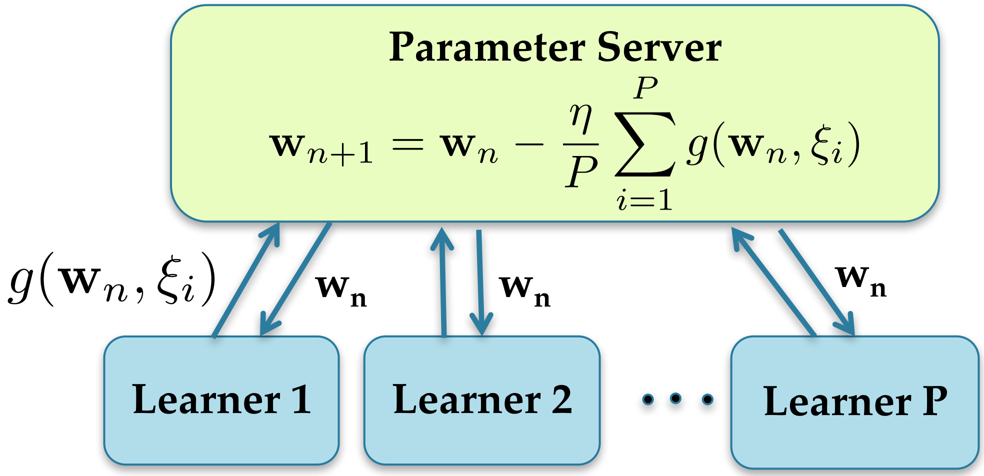
Main Results

Runtime & Error Analysis of Sync, Async SGD

Straggler Mitigation via SGD variants

Staleness Compensation in Async SGD

Expected Time Per Iteration

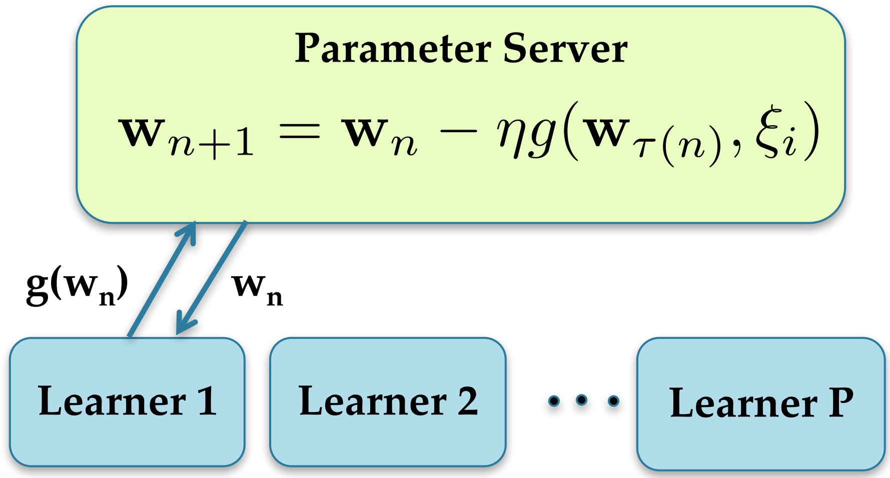


Each learner takes time X~ exp(μ)

Synchronous SGD

$$\mathbb{E}[T] = \mathbb{E}[X_{P:P}]$$
$$\approx \frac{1}{\mu}\log P$$

Expected Time Per Iteration



Each learner takes time X~ exp(μ)

Synchronous SGD

$$\mathbb{E}[T] = \mathbb{E}[X_{P:P}]$$
$$\approx \frac{1}{\mu} \log P$$

Asynchronous SGD

$$\mathbb{E}[T] = \frac{1}{\mu P}$$

P log P times smaller!

Sync SGD: Error Analysis

Update Rule: Equivalent to mini-batch SGD with batch size Pm

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \frac{\eta}{P} \sum_{i=1}^{P} g(\mathbf{w}_n, \xi_i)$$

For c-strongly convex, L-smooth functions [Bottou, 2016]

$$\mathbb{E}[F(\mathbf{w}_J) - F^*] \leq \frac{\eta L \sigma^2}{2c(Pm)} + (1 - \eta c)^J \left(F(\mathbf{w}_0) - F^* - \frac{\eta L \sigma^2}{2c(Pm)} \right)$$

Error Floor

Decay Rate

Async SGD: Error Analysis

Update Rule $\mathbf{w}_{n+1} = \mathbf{w}_n - \eta g(\mathbf{w}_{\tau(n)}, \xi_i)$

Hard to analyze due to stale gradients

Assumptions in Previous works

- Upper Bound on Staleness $\tau(n) \leq B$ [Lian et al 2015]
- Geometric staleness distribution

  $P(\tau(n) = j) = p(1-p)^{j-1}$ [Mitiliagkas et al 2016]
- Independently drawn gradient staleness

We remove these assumptions, and instead consider

$$\mathbb{E}[||\nabla F(\mathbf{w}_j) - \nabla F(\mathbf{w}_{\tau(j)})||_2^2] \leq \gamma \mathbb{E}[||\nabla F(\mathbf{w}_j)||_2^2] \qquad \gamma \leq 1$$

Async SGD: Error Analysis

For c-strongly convex, L-smooth functions,

$$\mathbb{E}[F(\mathbf{w}_J) - F^*] \leq \frac{\eta L \sigma^2}{2c\gamma' m} + (1 - \eta c \gamma')^J \left(\mathbb{E}[F(\mathbf{w}_0) - F^*] - \frac{\eta L \sigma^2}{2c\gamma' m} \right)$$

Larger than Sync-SGD

Can be faster than Sync SGD if $p_o / 2 > \gamma$

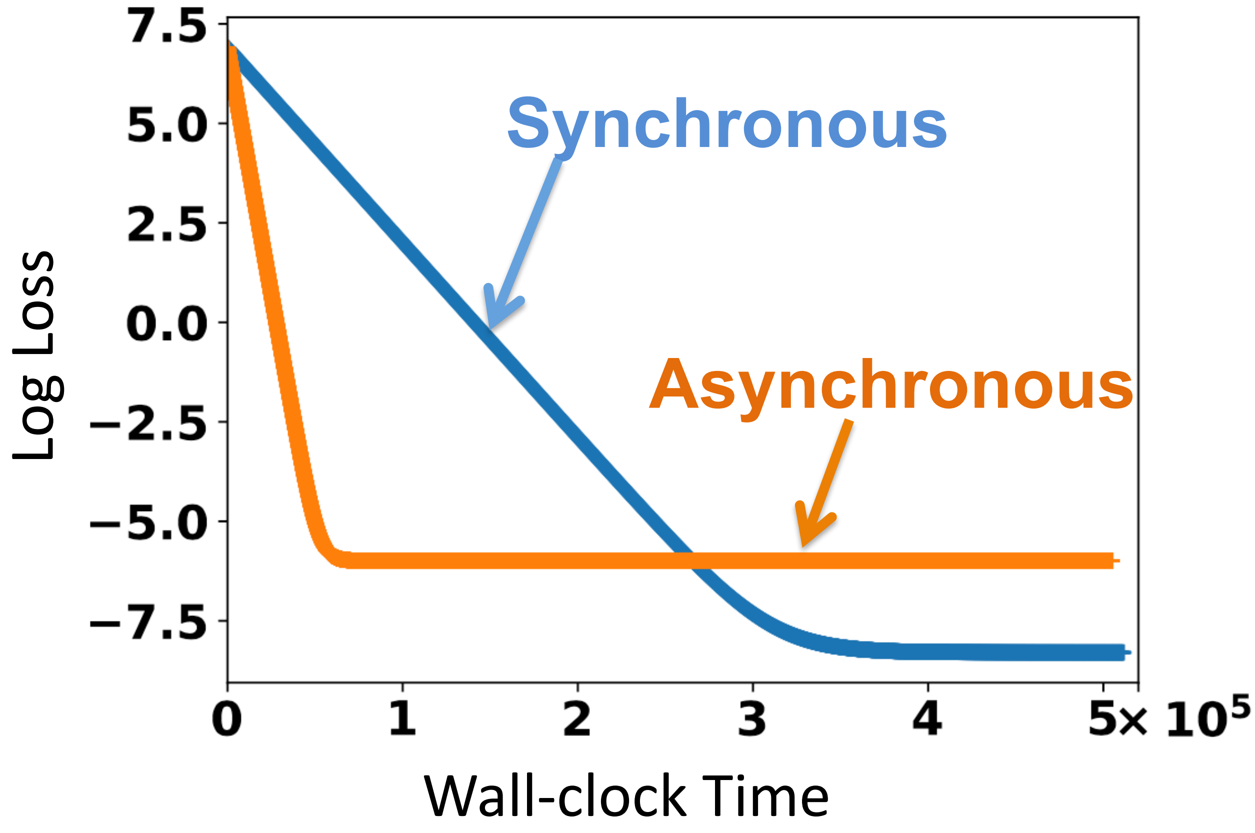
where $\gamma' = 1 - \gamma + p_0/2$

γ is the staleness bound,

and p_0 is the probability of getting a fresh gradient

Analysis can be generalized to non-convex objectives

Need to compare convergence w.r.t. *wall-clock time* instead of iterations



Main Results

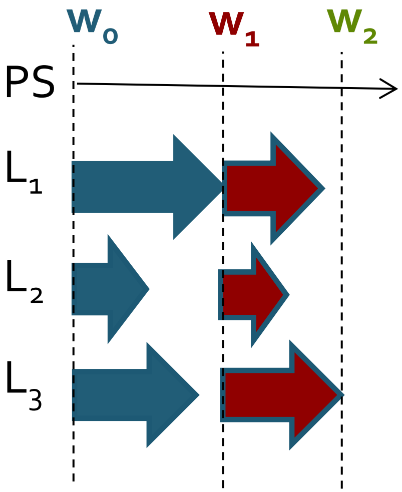
Runtime & Error Analysis of Sync, Async SGD

Straggler Mitigation via SGD variants

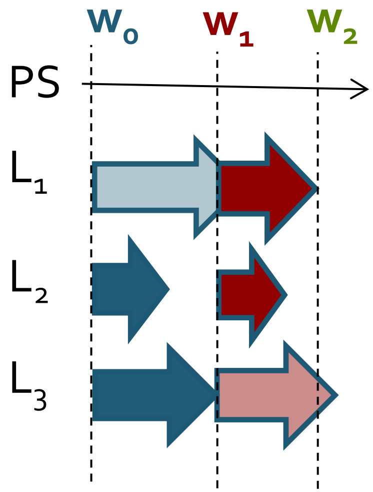
Staleness Compensation in Async SGD

Sync SGD Variants

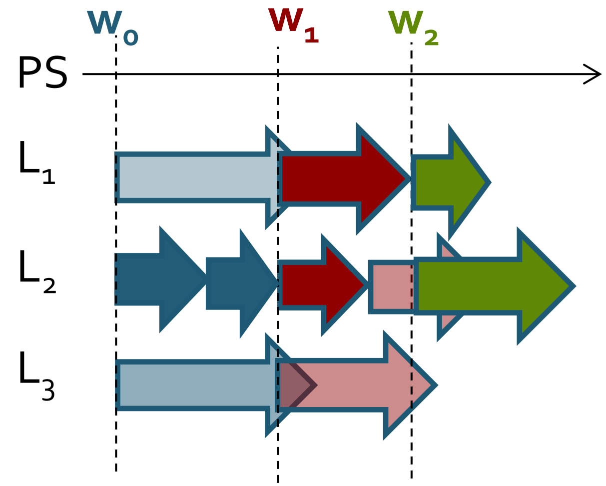
Fully Sync-SGD



K-Sync SGD



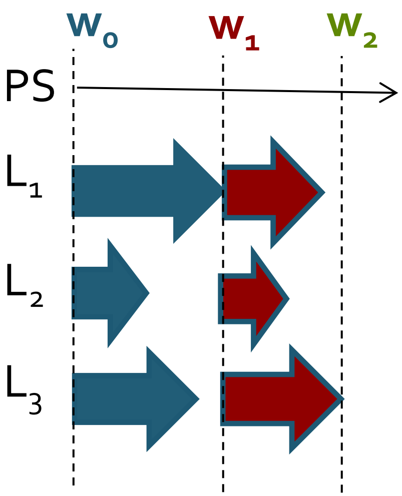
K-Batch Sync SGD



Instead of using coding, we are utilizing the inherent redundancy in data

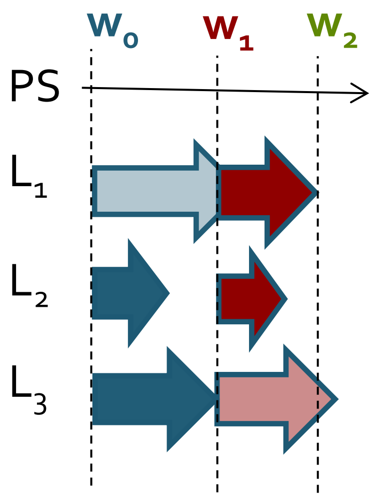
Sync SGD: Expected Time Per Iteration

Fully Sync-SGD



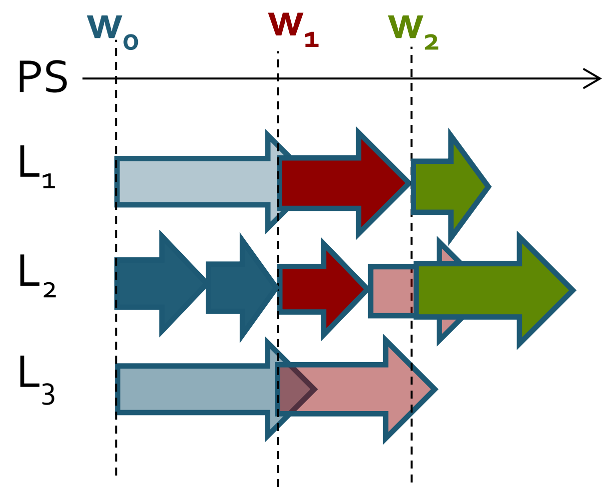
$$\mathbb{E}[T] = \mathbb{E}[X_{P:P}]$$
$$\approx \frac{1}{\mu} \log P$$

K-Sync SGD



$$\mathbb{E}[T] = \mathbb{E}[X_{K:P}]$$
$$\approx \frac{1}{\mu} \log \frac{P}{P-K}$$

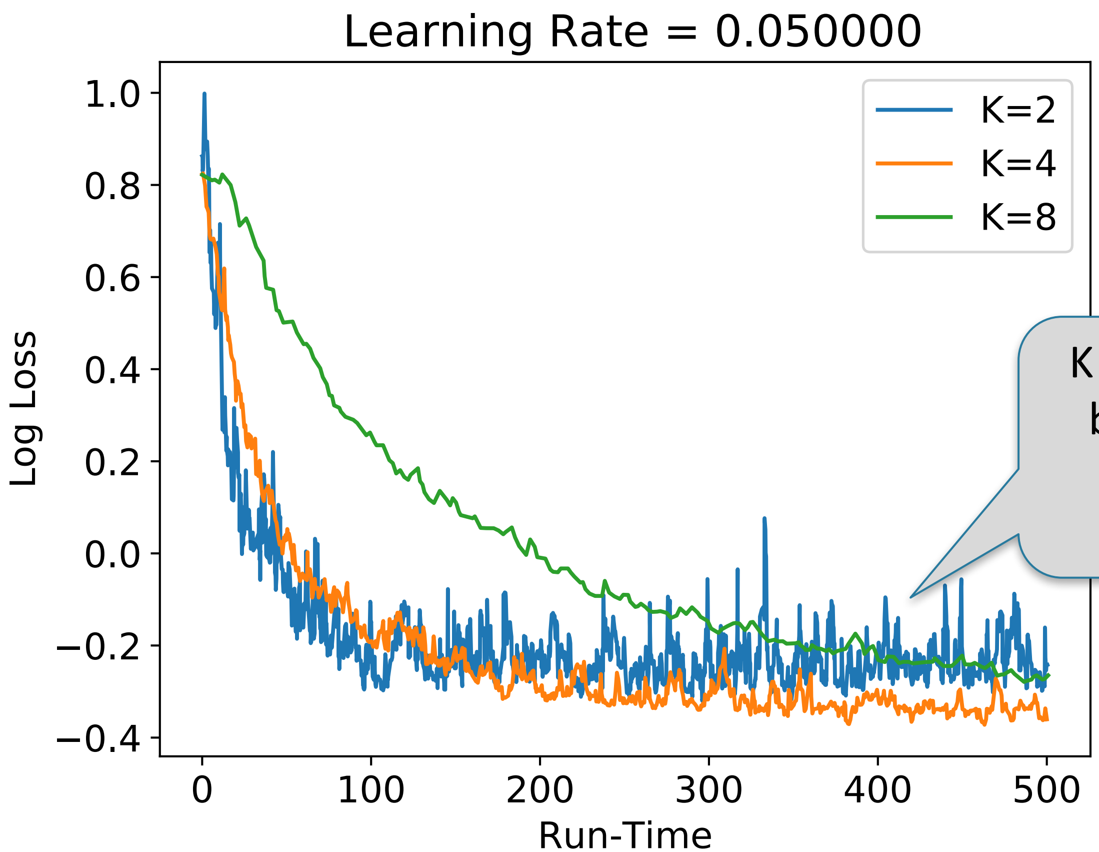
K-Batch Sync SGD



$$\mathbb{E}[T] = \frac{K}{\mu P}$$

Sync SGD: Choosing the best K

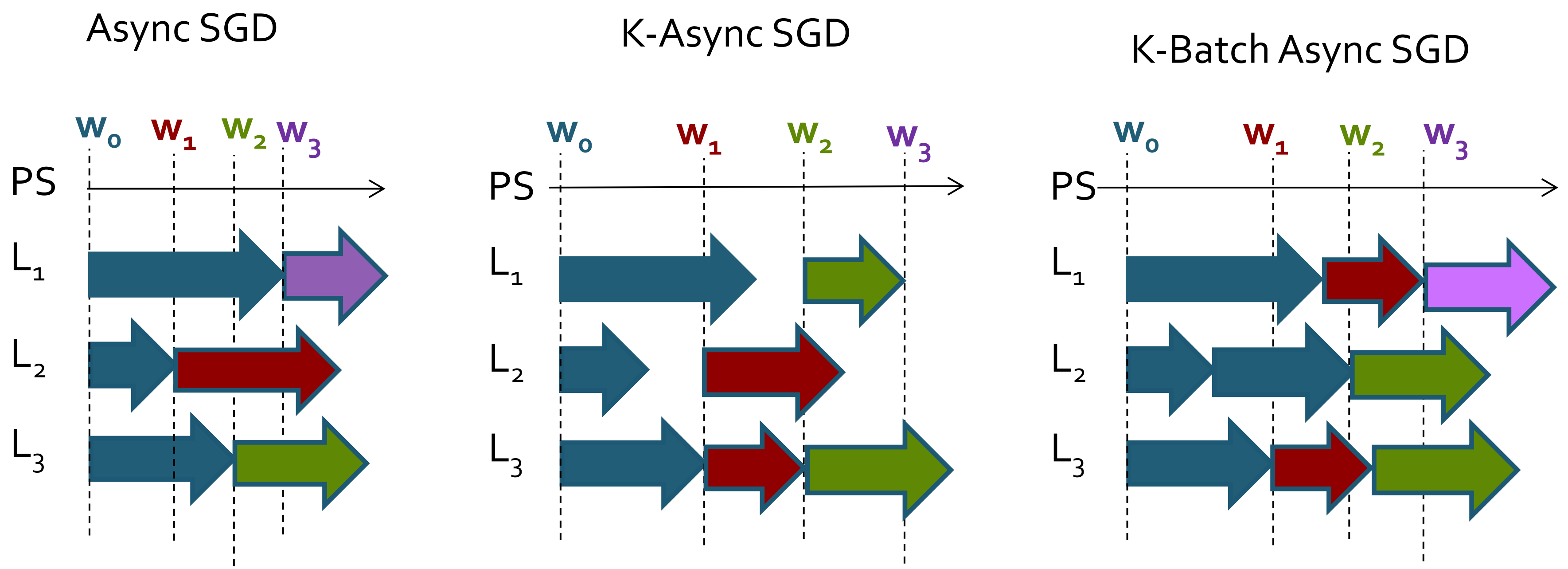
Error is equivalent to mini-batch SGD with batch size Km



MNIST
dataset

K = 4 strikes a good balance between conv.speed and error floor

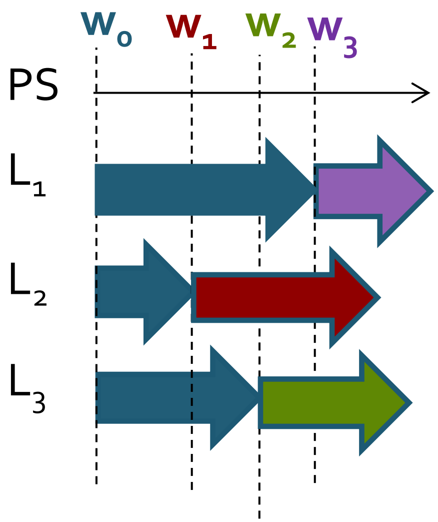
Async SGD Variants



Our error analysis for Async SGD can be generalized to these variants

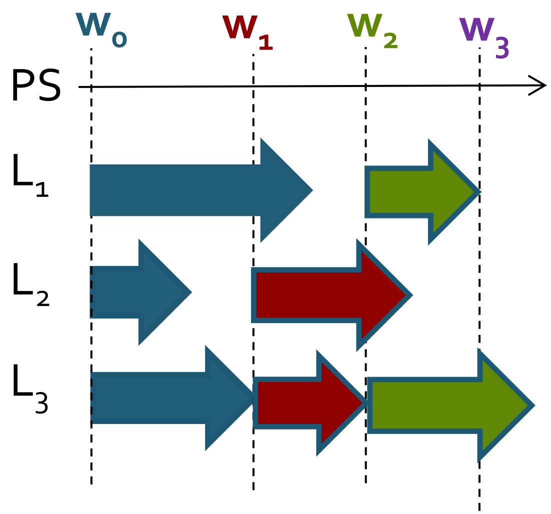
Async SGD: Expected Time Per Iteration

Async SGD



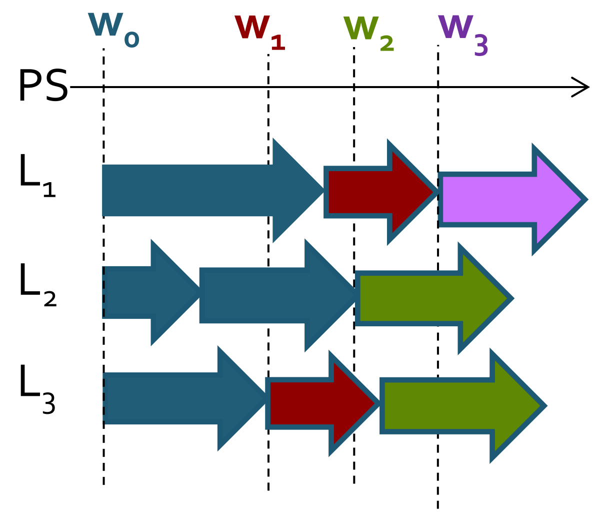
$$\mathbb{E}[T] = \frac{1}{\mu P}$$

K-Async SGD



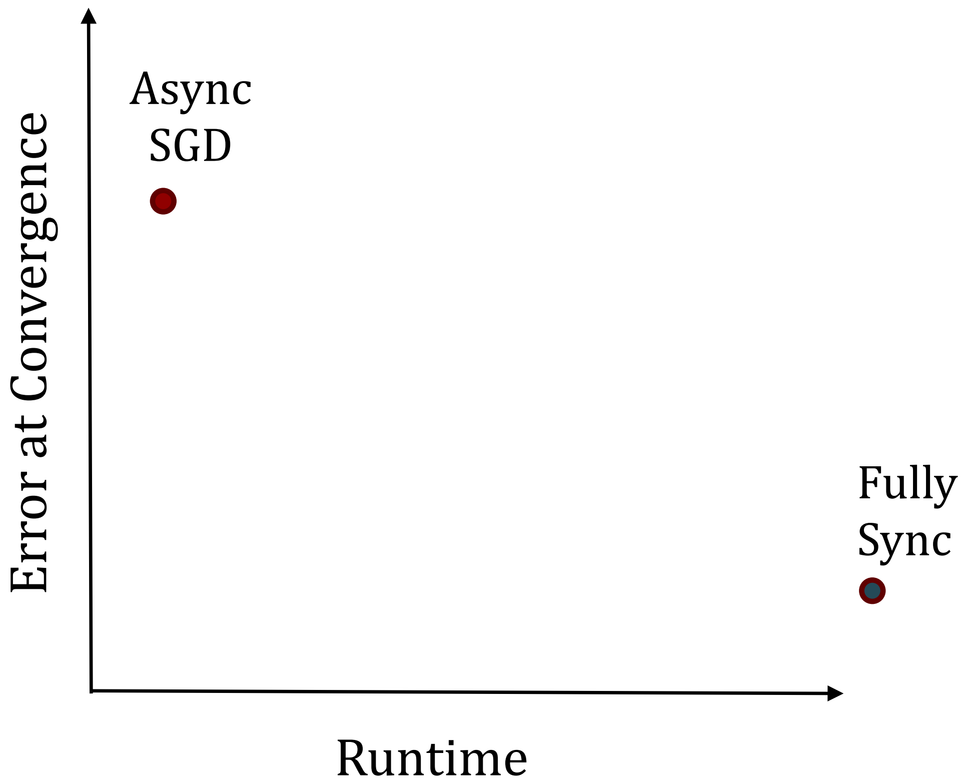
$$\mathbb{E}[T] = \mathbb{E}[X_{K:P}]$$
$$\approx \frac{1}{\mu} \log \frac{P}{P-K}$$

K-Batch Async SGD

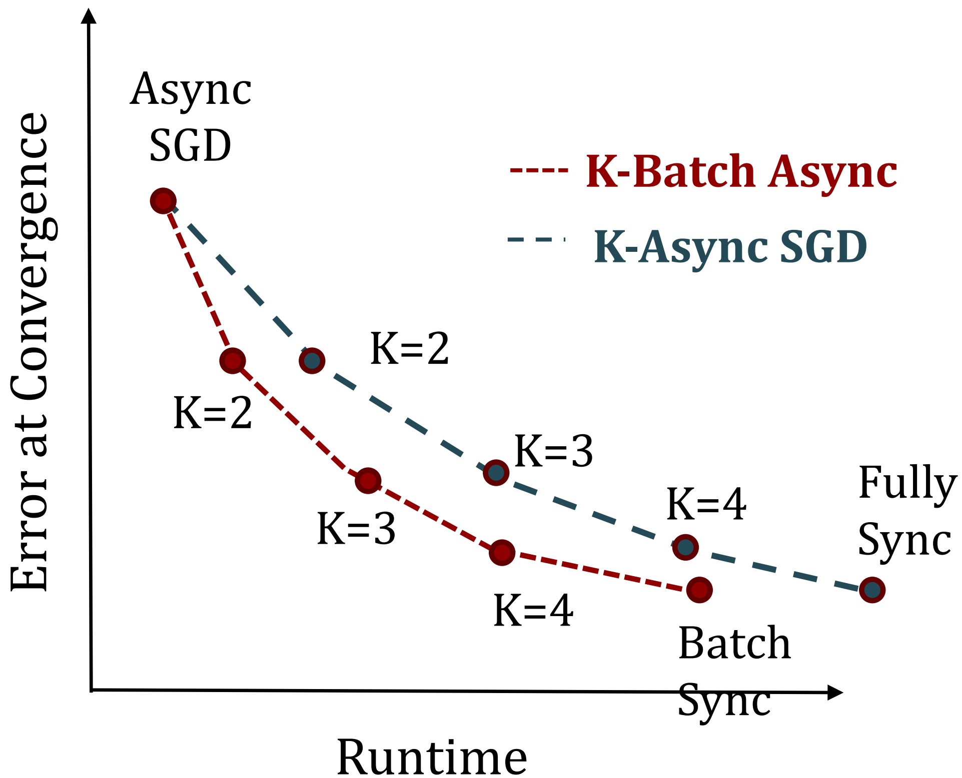


$$\mathbb{E}[T] = \frac{K}{\mu P}$$

Spanning the spectrum between Synchronous and Asynchronous SGD



Spanning the spectrum between Synchronous and Asynchronous SGD



Main Results

Runtime & Error Analysis of Sync, Async SGD

Straggler Mitigation via SGD variants

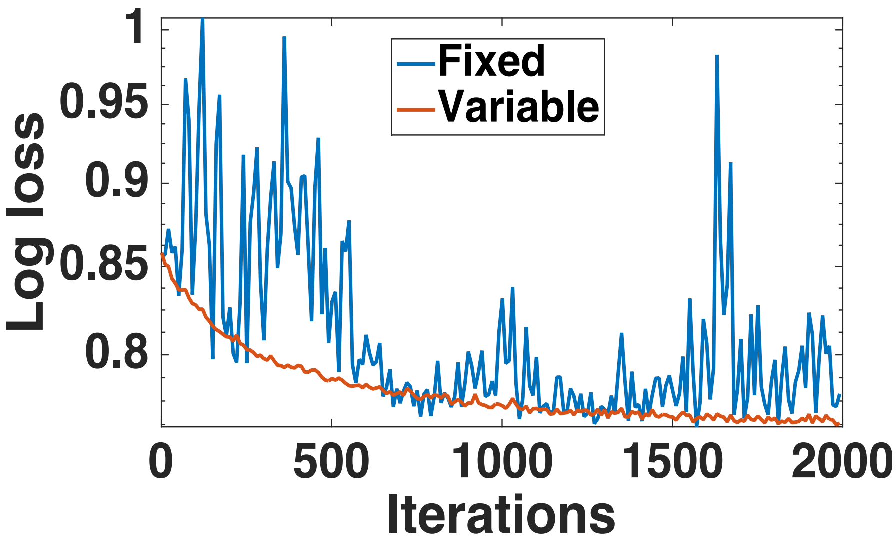
Staleness Compensation in Async SGD

Adapting the Learning Rate to Tame Gradient Staleness

Proposed Learning Rate Schedule

$$\eta_j = \min \left\{ \frac{C}{||\mathbf{w}_j - \mathbf{w}_{\tau(j)}||_2^2}, \eta_{max} \right\}$$

helps eliminate the bounded staleness assumption in our analysis

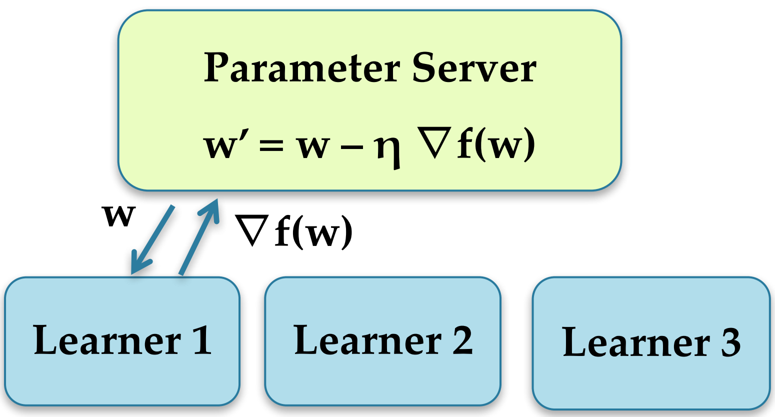


$\eta = 0.01$,
CIFAR10 dataset

Related to momentum tuning in [Mitliagkas 2016]

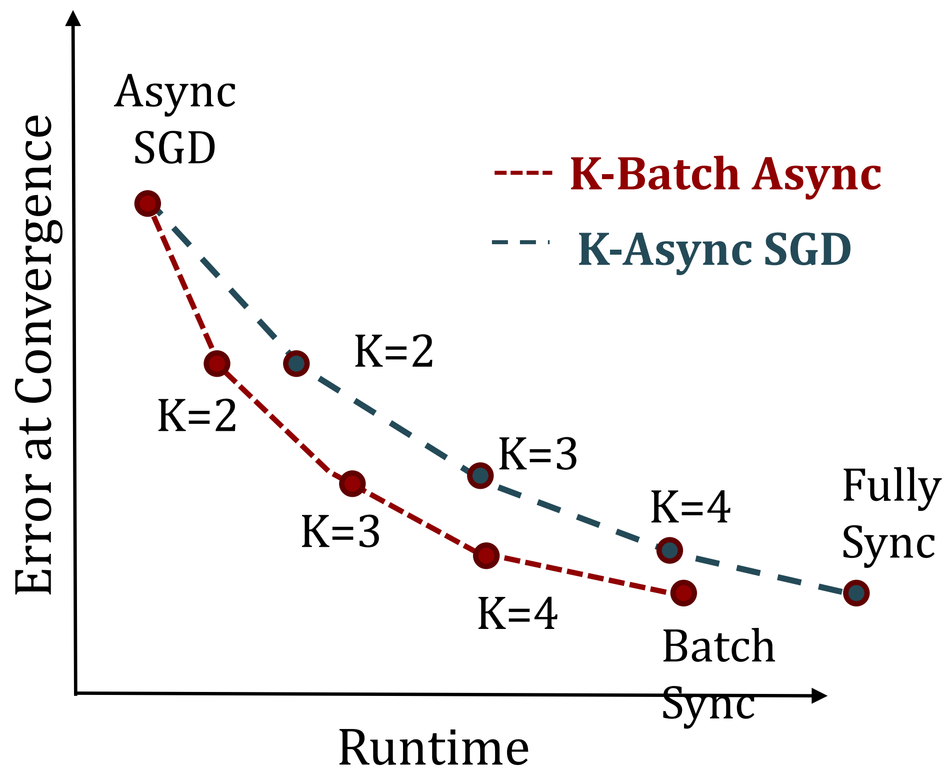
Key Takeaways

- True SGD convergence is w.r.t. the wall-clock time
- Integration scheduling & algorithmic techniques



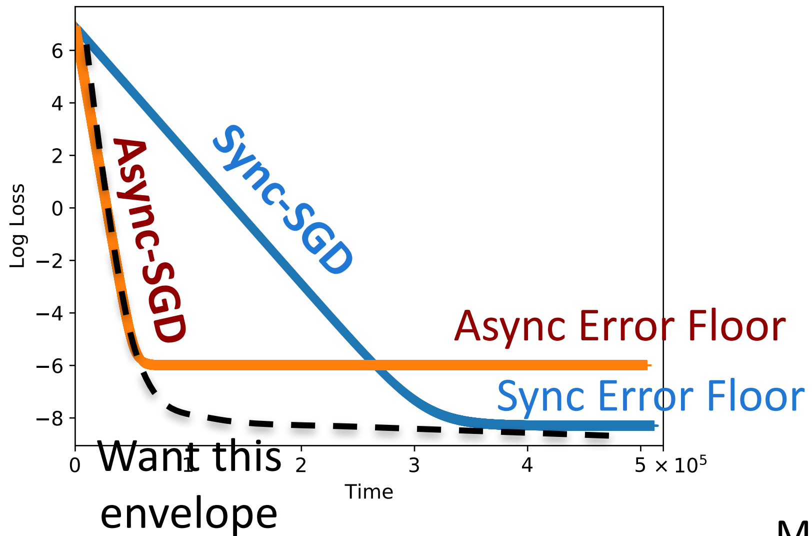
Key Issues

- Straggling Learners
- Gradient Staleness

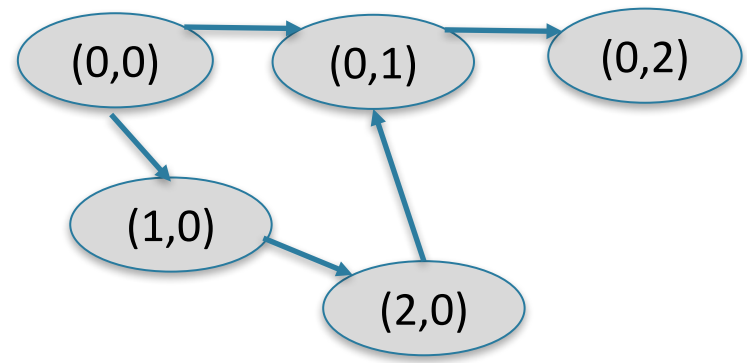


Ongoing & Future Directions

Gradually increasing synchrony



Stochastic Staleness Analysis



Minimizing Communication via Local Updates