Elastic Averaging SGD in Distributed Deep Learning

Sixin Zhang, Anna Choromanska, Yann LeCun, **NIPS 2015**

Slides by: ***Jianyu Wang***

18-847F: Foundations of Cloud & ML Infrastructure
Oct 31, 2018

Background

Key Ideas

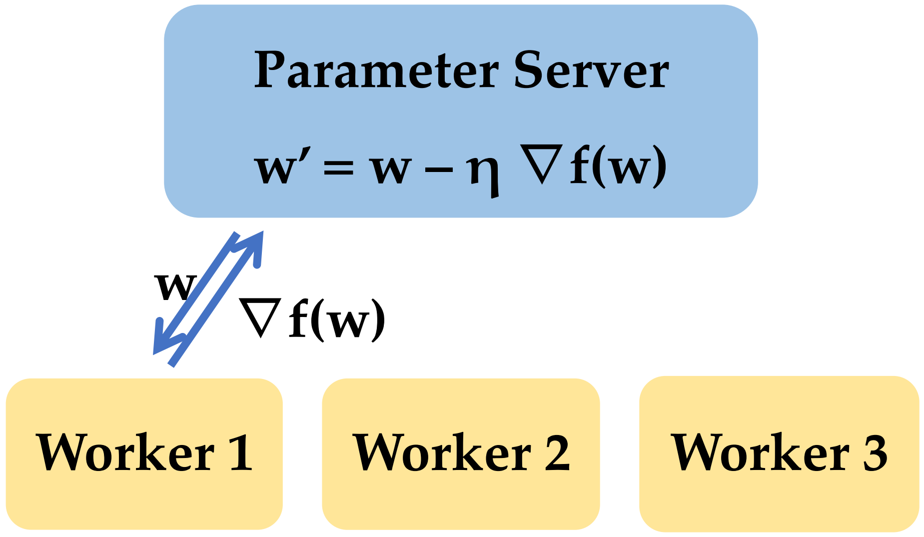
Update Rule: Sync. version

Async. & momentum variants

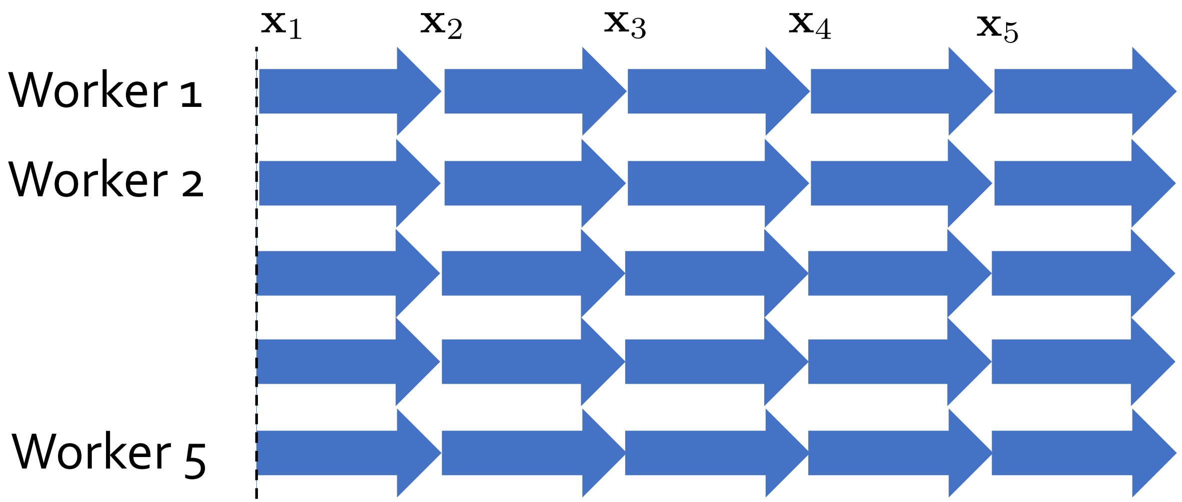
Theoretical Analysis

Experimental Results

Recap: Distributed SGD



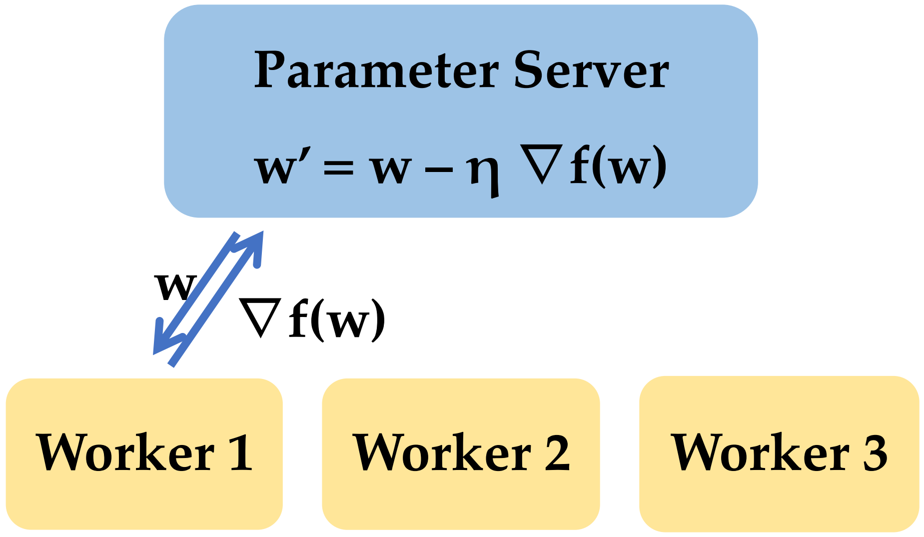
Parameter server framework



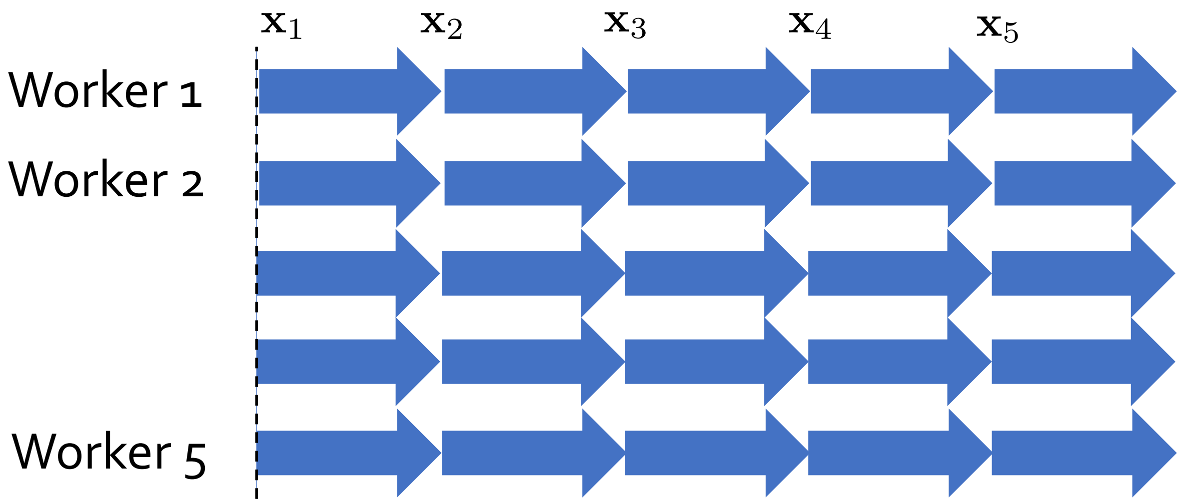
Execution pipeline **(Ideal case)**

Recap: Distributed SGD

What is the problem?



Parameter server framework



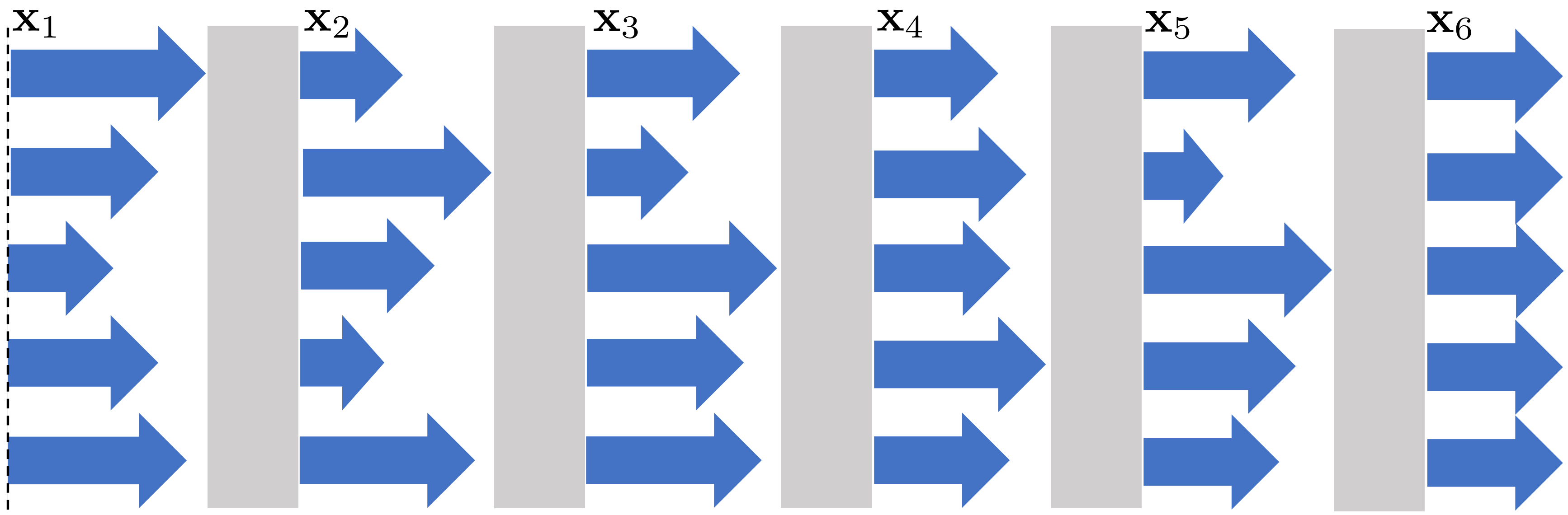
Execution pipeline **(Ideal case)**

Recap: Distributed SGD

What is the problem?

✓ Communication delay

✓ Straggler/Staleness



Background

Key Ideas

Update Rule: Sync. version

Async. & momentum variants

Theoretical Analysis

Experimental Results

Elastic Averaging SGD

Key Ideas:

✓ Workers maintain their local parameters

✓ Don't let local parameters go far away from central parameter

Dist. SGD:

Each worker

computes $g_i(\tilde{w}; \xi_i)$

EASGD:

Each worker

locally update w^i

Elastic Averaging SGD

Key Idea:

- ✓ Workers maintain their local parameters
- ✓ Don't let local parameters go far away from central parameter

Dist. SGD:

Minimize

$$\sum_{i=1}^{P} f_i(\tilde{w})$$

EASGD:

Minimize

$$\sum_{i=1}^{P} \left[f_i(w^i) + \frac{\rho}{2} \| w^i - \tilde{w} \|^2 \right]$$

Background

Key Ideas

Update Rule: Sync. version

Async. & momentum variants

Theoretical Analysis

Experimental Results

Update Rule

Sync. Version

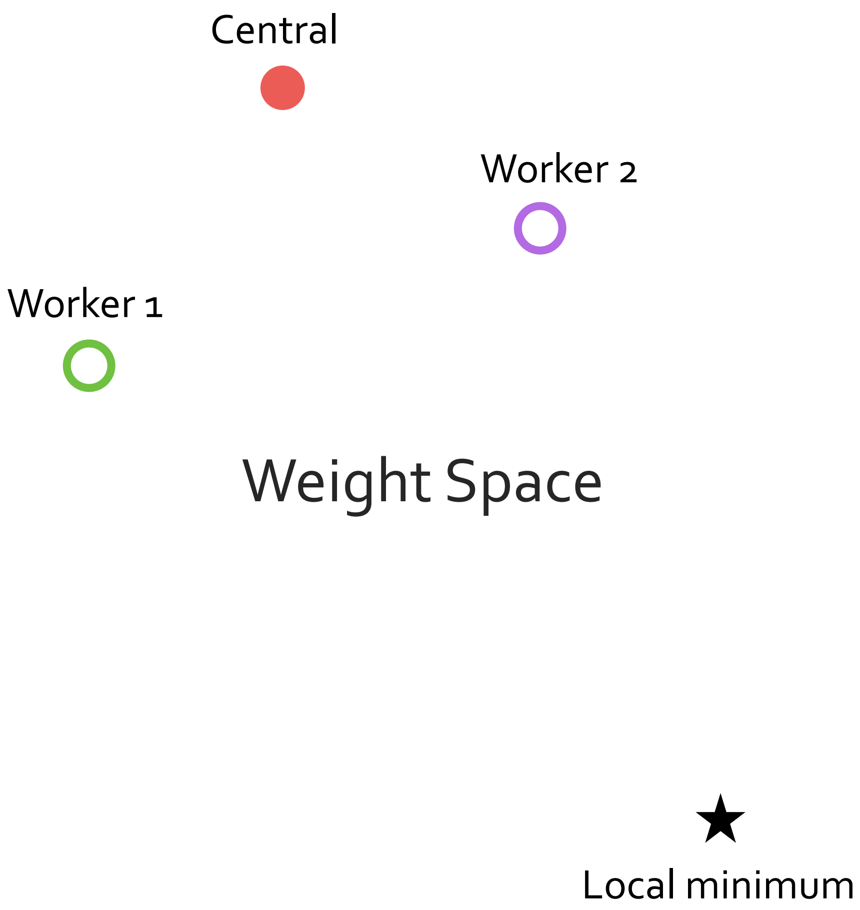
worker $w^i_+ = w^i - \eta g(w^i) - \text{Elastic Force}_i$

server $\tilde{w}_+ = \tilde{w} + \sum_i \text{Elastic Force}_i$

$$\text{Elastic Force}_i = \alpha(w^i - \tilde{w}) = \eta\rho(w^i - \tilde{w})$$

Update Rule

Sync. Version

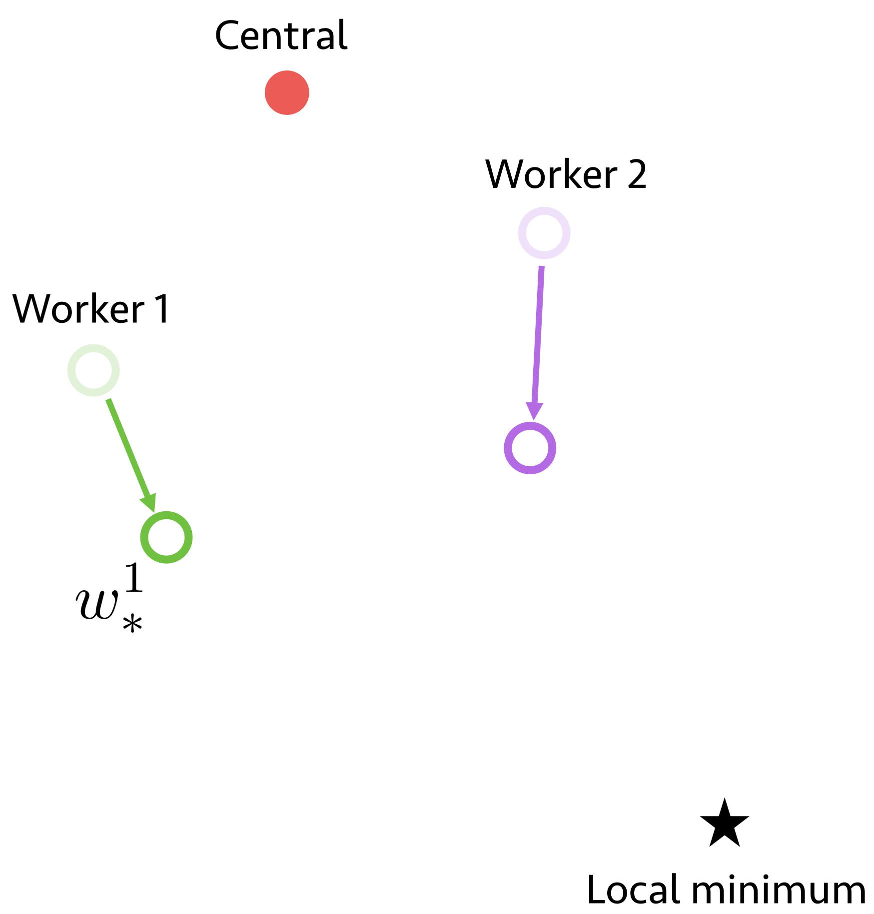


Global time = 0

Initialization

Update Rule

Sync. Version



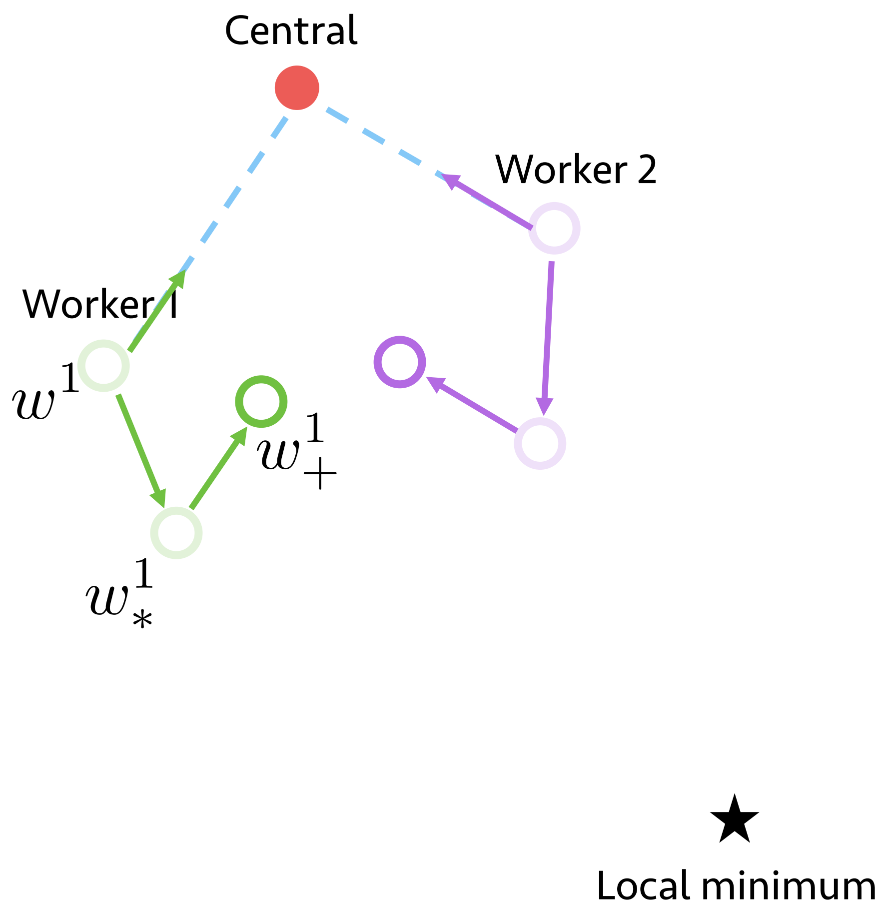
Global time = 0

Workers do **one** local **UPDATE**

$$w_*^i = w^i - \eta g(w^i; \xi^i)$$

Update Rule

Sync. Version



Global time = 0

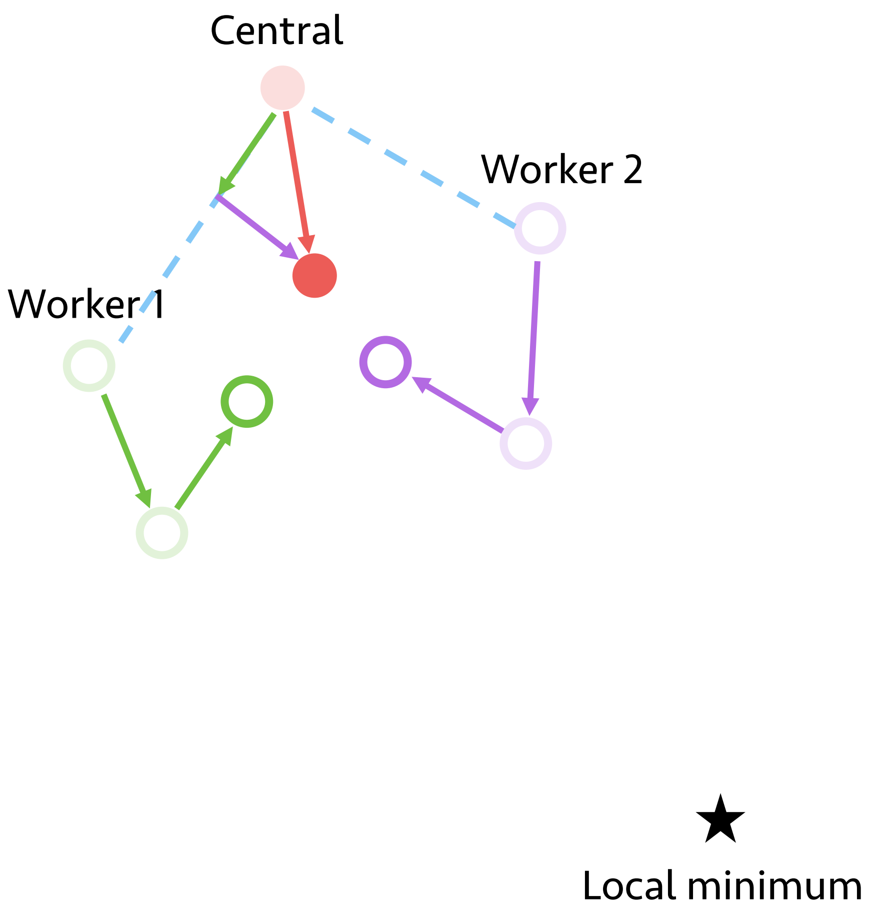
"Elastic Force"!

Workers go **BACK**.

$$w^i_+ = w^i_* - \alpha(w^i - \tilde{w})$$

Update Rule

Sync. Version



Global time = 1

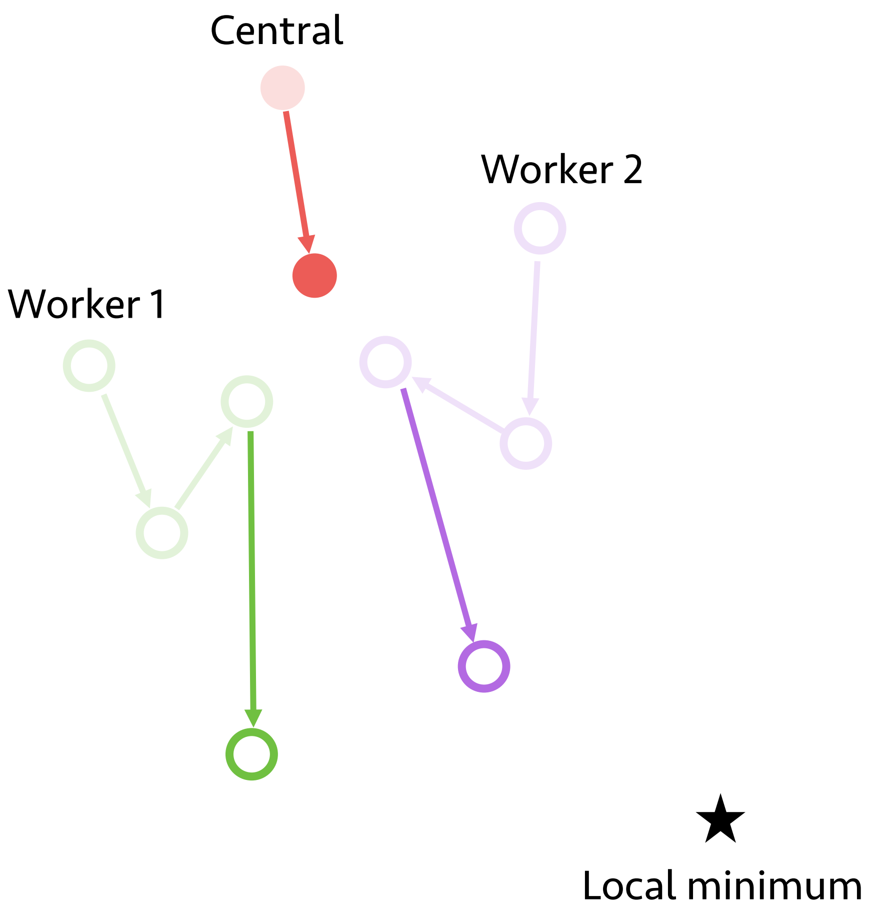
"Elastic force"!

Server moves **FORWARD**!

$$\tilde{w}_+ = \tilde{w} + \alpha \sum_{i=1}^{P} (w^i - \tilde{w})$$

Update Rule

Sync. Version



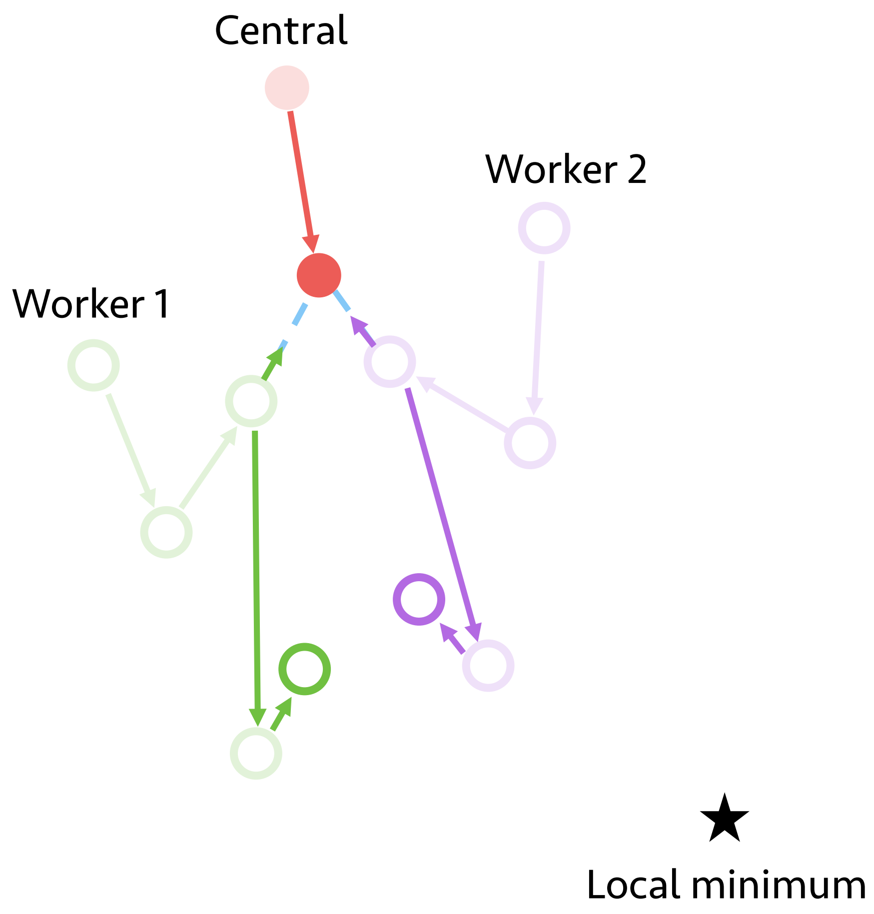
Global time = 1

Workers do **1** local **UPDATE**

$$w_*^i = w^i - \eta g(w^i; \xi^i)$$

Update Rule

Sync. Version



Global time = 1

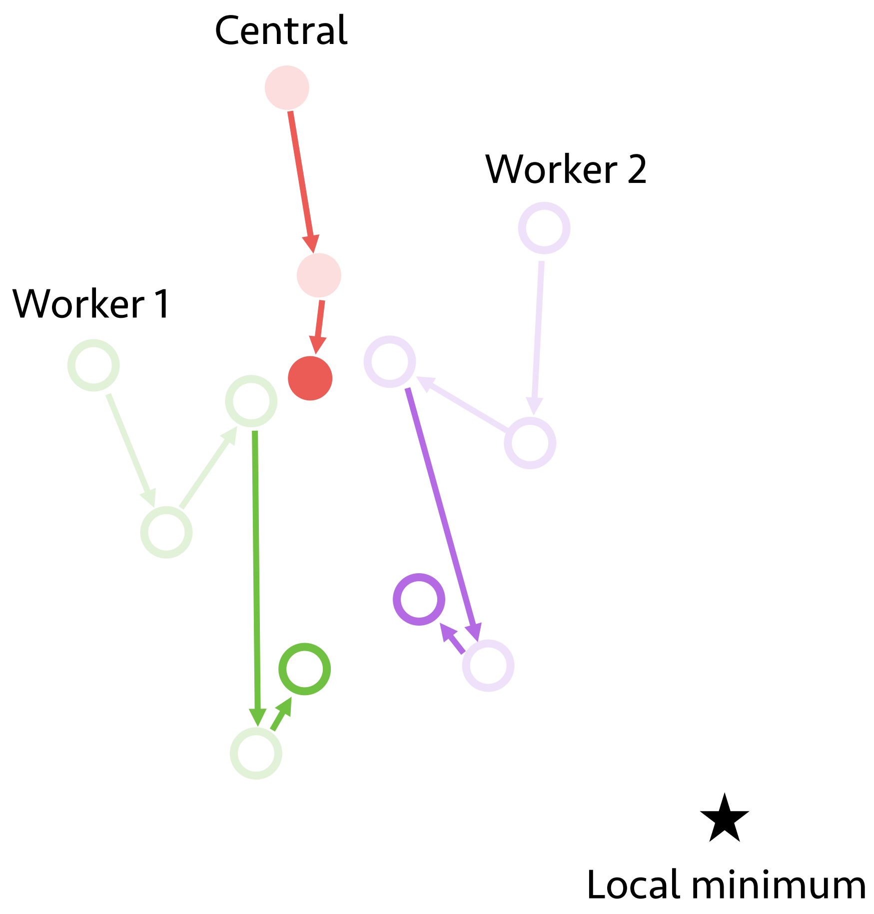
"Elastic Force"!

Workers go **BACK**.

$$w_+^i = w_*^i - \alpha(w^i - \tilde{w})$$

Update Rule

Sync. Version



Global time = 2

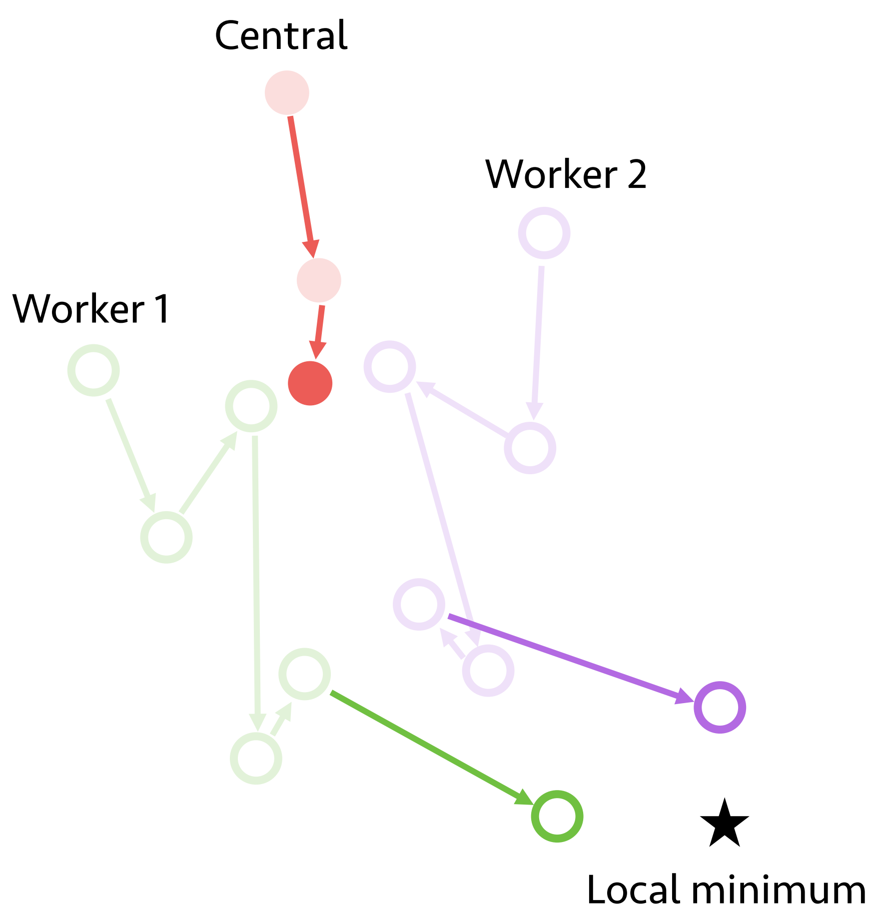
"Elastic force"!

Server moves **FORWARD**!

$$\tilde{w}_+ = \tilde{w} + \alpha \sum_{i=1}^{P} (w^i - \tilde{w})$$

Update Rule

Sync. Version

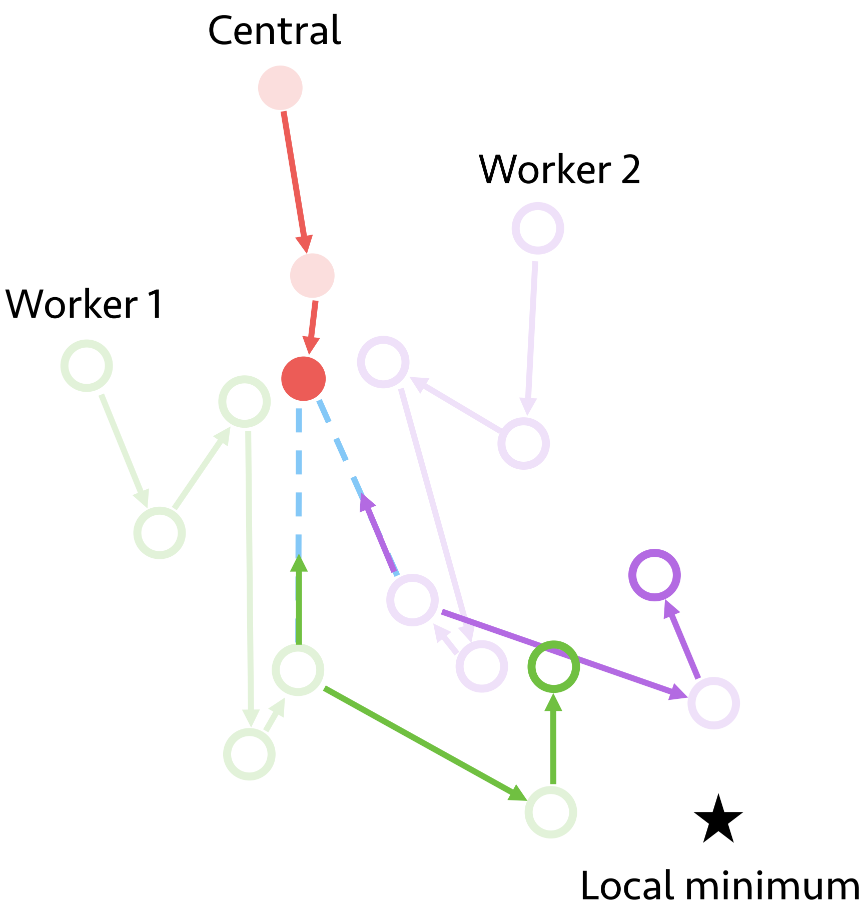


Global time = 2

LOCAL UPDATES

Update Rule

Sync. Version

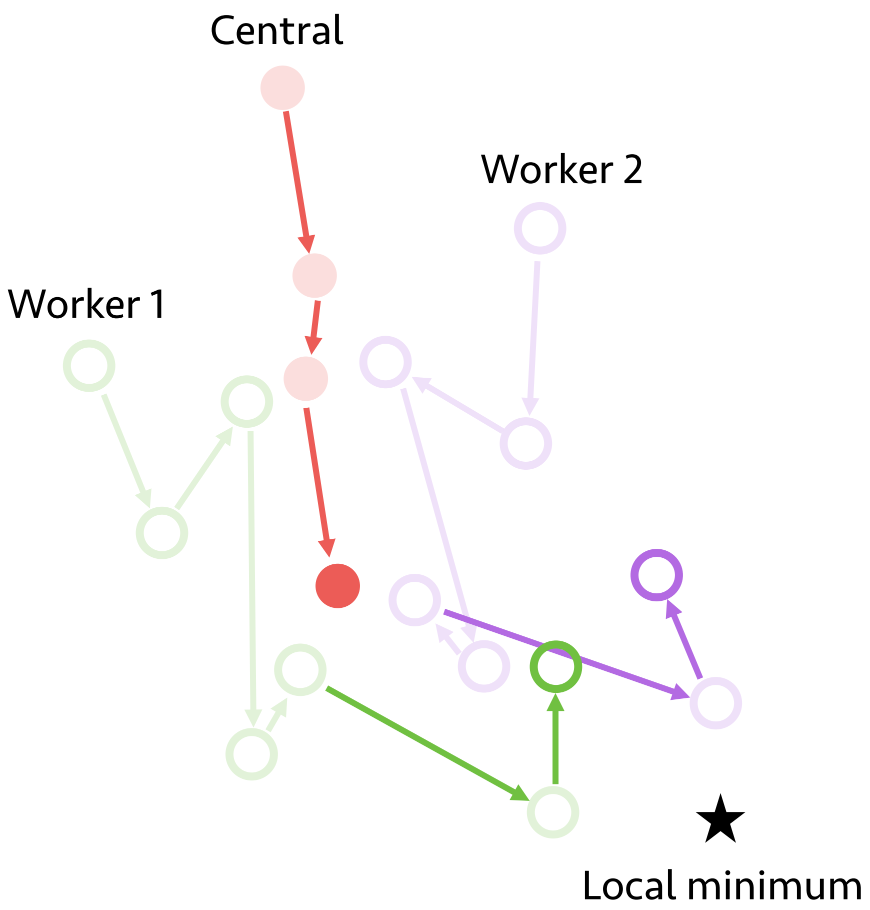


Global time = 2

BACK

Update Rule

Sync. Version



Global time = 3

FORWARD

Background

Key Ideas

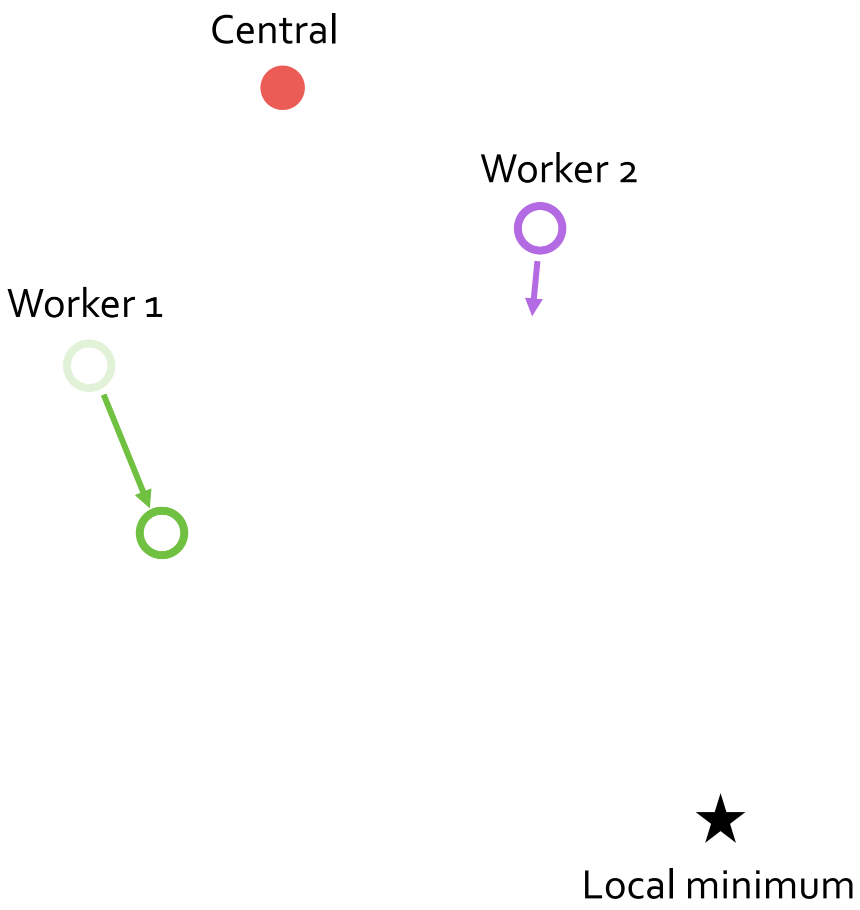
Update Rule: Sync. version

Async. & momentum variants

Theoretical Analysis

Experimental Results

Variant 01: Asynchronous EASGD



Global time = 0

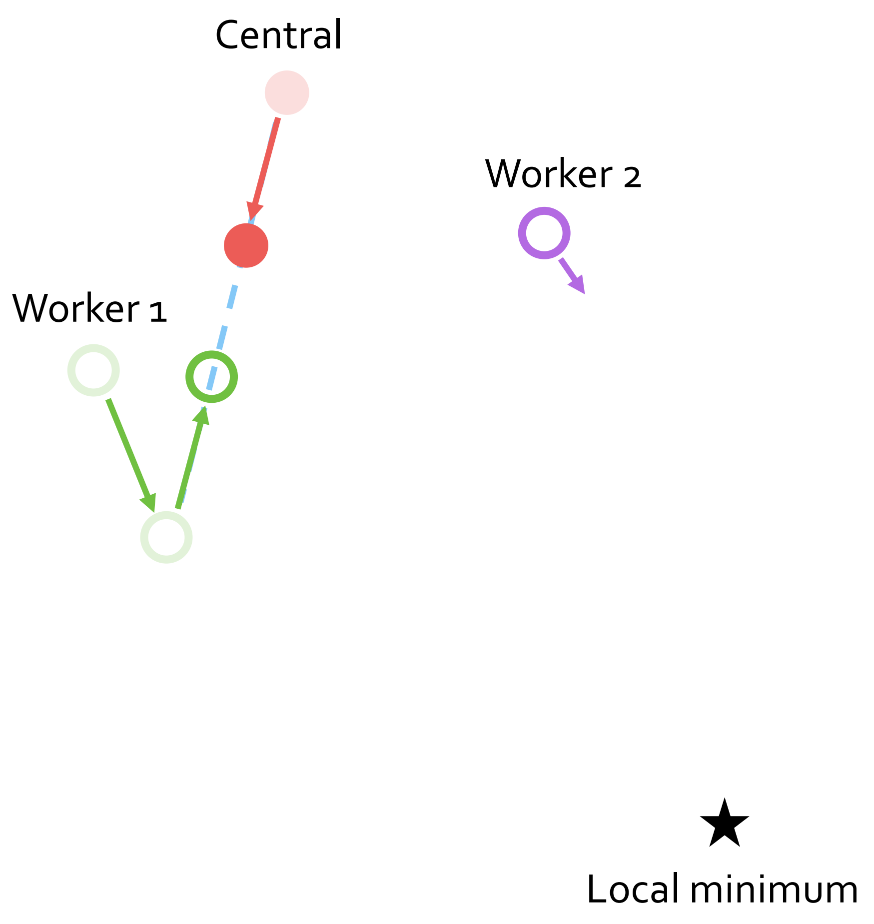
Configurable commun. period

Worker 1 finishes its **T** local updates

Worker 2 doesn't

$$w_T^i = w_0^i - \sum_{j=0}^{T-1} \eta_j g(w_j^i; \xi_j^i)$$

Variant 01: Asynchronous EASGD



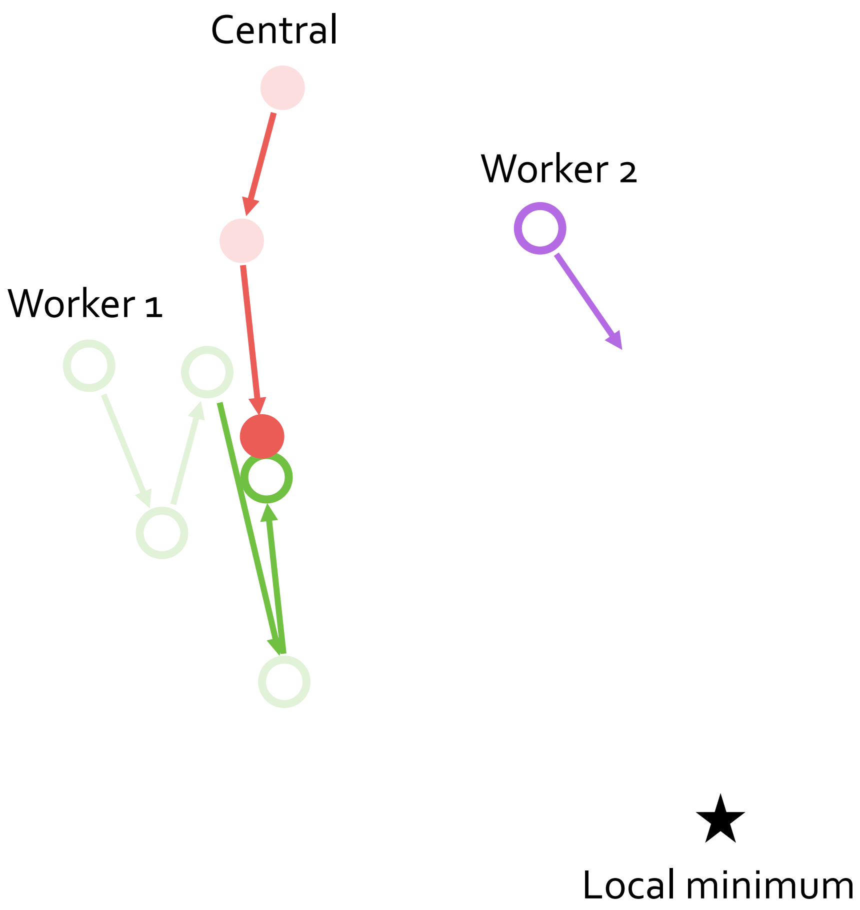
Global time = 1

Elastic Force!

Move **back** and move **forward**.

Only worker 1 communicates with parameter server.

Variant 01: Asynchronous EASGD

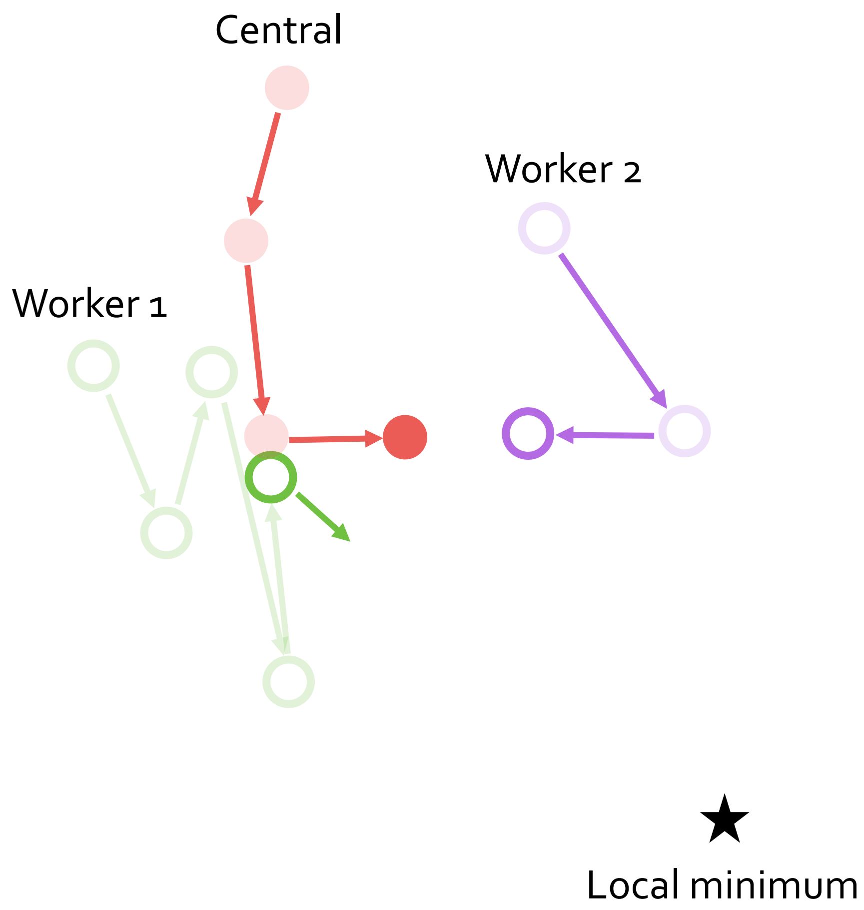


Global time = 2

Worker 1 finishes another T updates

Worker 2 doesn't

Variant 01: Asynchronous EASGD

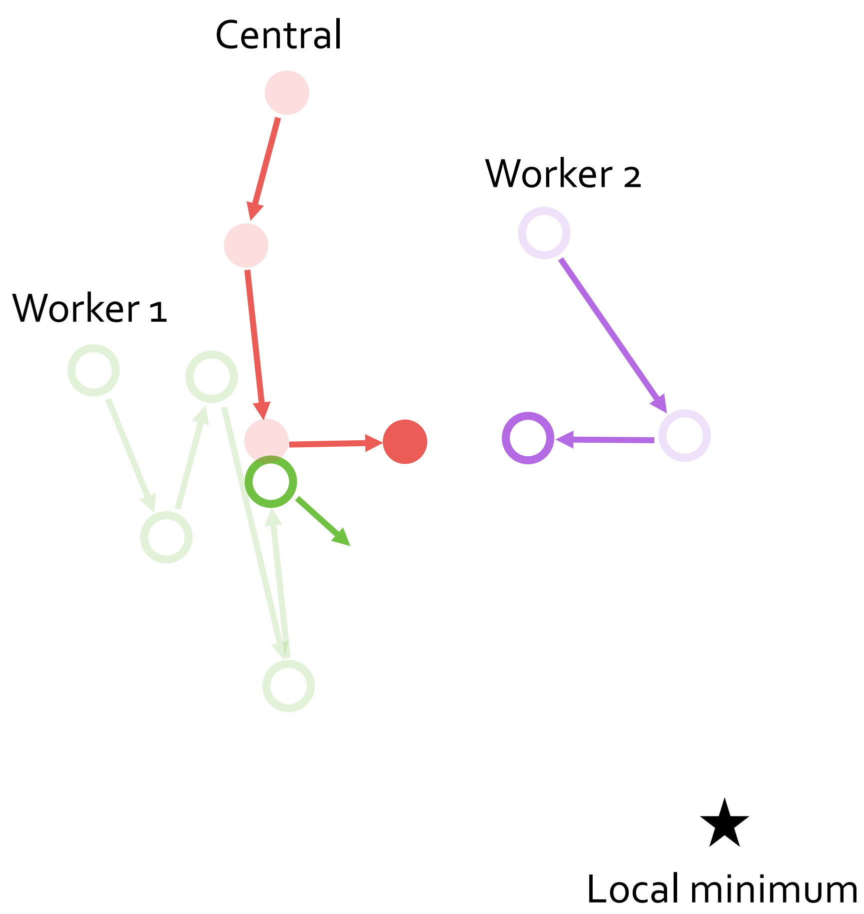


Global time = 3

Worker 2 finishes its **first T** updates

Worker 1 doesn't finish its **third T** updates

Variant 01: Asynchronous EASGD



Global time = 3

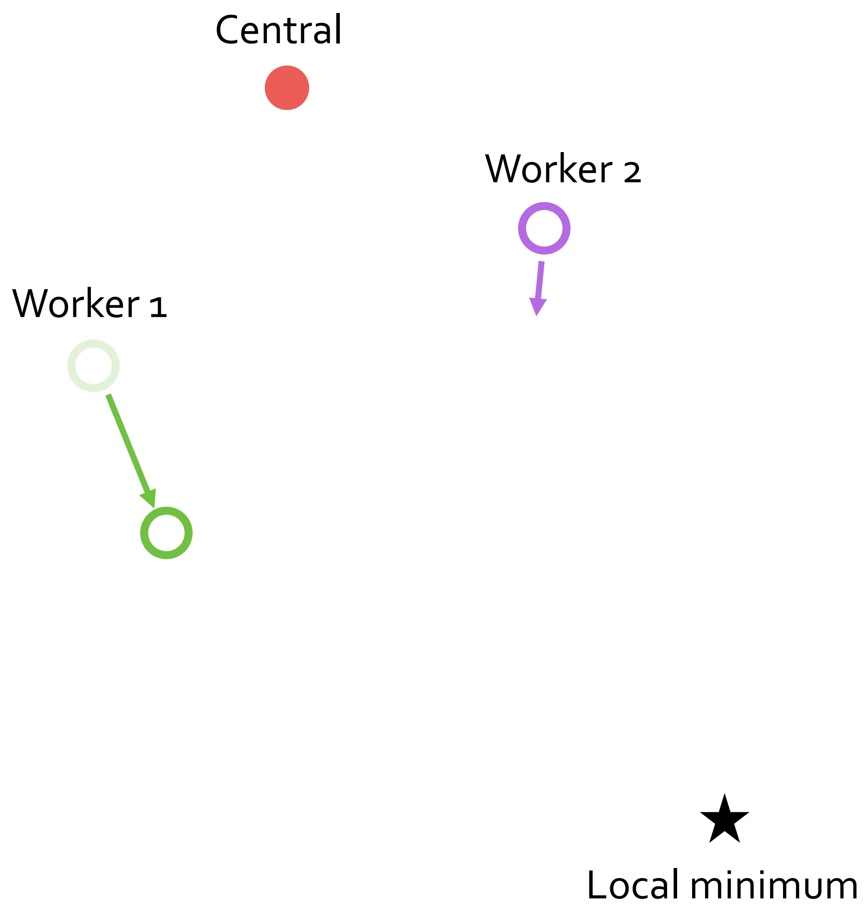
Worker 2 finishes its **first T** updates

Worker 1 doesn't finish its **third T** updates

This algorithm is robust w.r.t. the communication period T.

Increase T, reduce comm. overhead!

Variant 02: Momentum EASGD



Global time = 0

Local worker uses Nesterov momentum SGD.

$$\cancel{w_T^i = w_0^i - \sum_{j=0}^{T-1} \eta_j g(w_j^i; \xi_j^i)}$$

$$\boxed{\begin{aligned} v_{t+1}^i &= \delta v_t^i - \eta g(w_t^i + \delta v_t^i) \\ w_{t+1}^i &= w_t^i + v_t^i \end{aligned}}$$

Local workers converge faster!

Background

Key Ideas

Update Rule: Sync. version

Async. & momentum variants

Theoretical Analysis

Experimental Results

Stability Analysis

The objective we want to optimize in each iteration can be formulized as:

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{n} F(w^i) \\ &\text{subject to} \quad w^i - \tilde{w} = 0 \end{aligned}$$

Stability Analysis

Alternating **D**irection **M**ethods for **M**ultipliers (ADMM)

$$\begin{aligned}&\text{minimize} \quad \sum_{i=1}^{n} F(w^i) \\ &\text{subject to} \quad w^i - \tilde{w} = 0\end{aligned}$$

$$\Rightarrow$$

$$\text{minimize} \quad \sum_{i=1}^{n} \left[F(w^i) - \lambda^i (w^i - \tilde{w}) + \frac{\rho}{2} (w^i - \tilde{w})^2 \right]$$

Stability Analysis

Alternating **D**irection **M**ethods for **M**ultipliers (ADMM)

$$\text{minimize} \quad \sum_{i=1}^{n} F(w^i)$$
$$\text{subject to} \quad w^i - \tilde{w} = 0$$

$$\text{minimize} \quad \sum_{i=1}^{n} \left[F(w^i) - \lambda^i (w^i - \tilde{w}) + \frac{\rho}{2} (w^i - \tilde{w})^2 \right]$$

One dimensional quadratic case + Round-Robin scheme

Stability Analysis

Alternating **D**irection **M**ethods for **M**ultipliers (ADMM)

$$\text{minimize} \quad \sum_{i=1}^{n} F(w^i)$$
$$\text{subject to} \quad w^i - \tilde{w} = 0$$

$$\Rightarrow$$

$$\text{minimize} \quad \sum_{i=1}^{n} \left[F(w^i) - \lambda^i (w^i - \tilde{w}) + \frac{\rho}{2} (w^i - \tilde{w})^2 \right]$$

Key takeaway

- ✓ In **1-D quadratic case**, ADMM algorithm can exhibit chaotic behavior, leading to exponential divergence.
- ✓ The analytic condition for ADMM to be stable is still unknown, while for EASGD it is very simple.

Stability Analysis

Some basic convergence analysis in:

- ✓ One dimensional quadratic case
- ✓ Multi-dimensional quadratic case
- ✓ Strongly convex case

Hasn't been studied sufficiently!

Key takeaway

- ✓ In **1-D quadratic case**, ADMM algorithm can exhibit chaotic behavior, leading to exponential divergence.
- ✓ The analytic condition for ADMM to be stable is still unknown, while for EASGD it is very simple.

Background

Key Ideas

Update Rule: Sync. version

Async. & momentum variants

Theoretical Analysis

Experimental Results

Experimental Setup

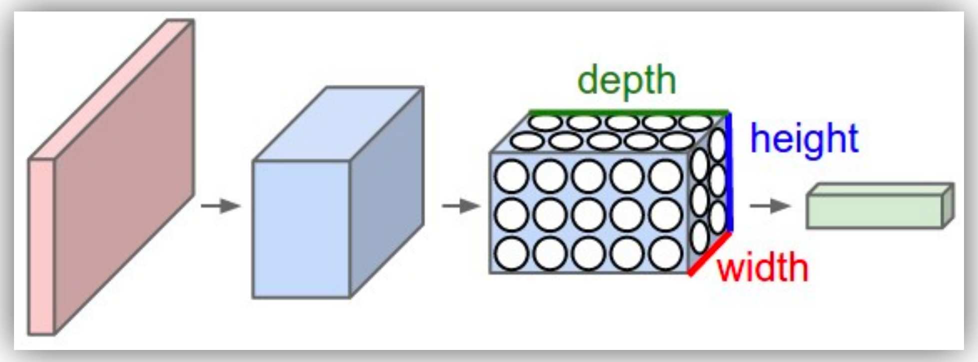
Hardware

- ✓ Running on a GPU-cluster
- ✓ Parameter-sever framework

(https://zh.gluon.ai/chapter_gluon-advances/multiple-gpus-scratch.html)

ML Model

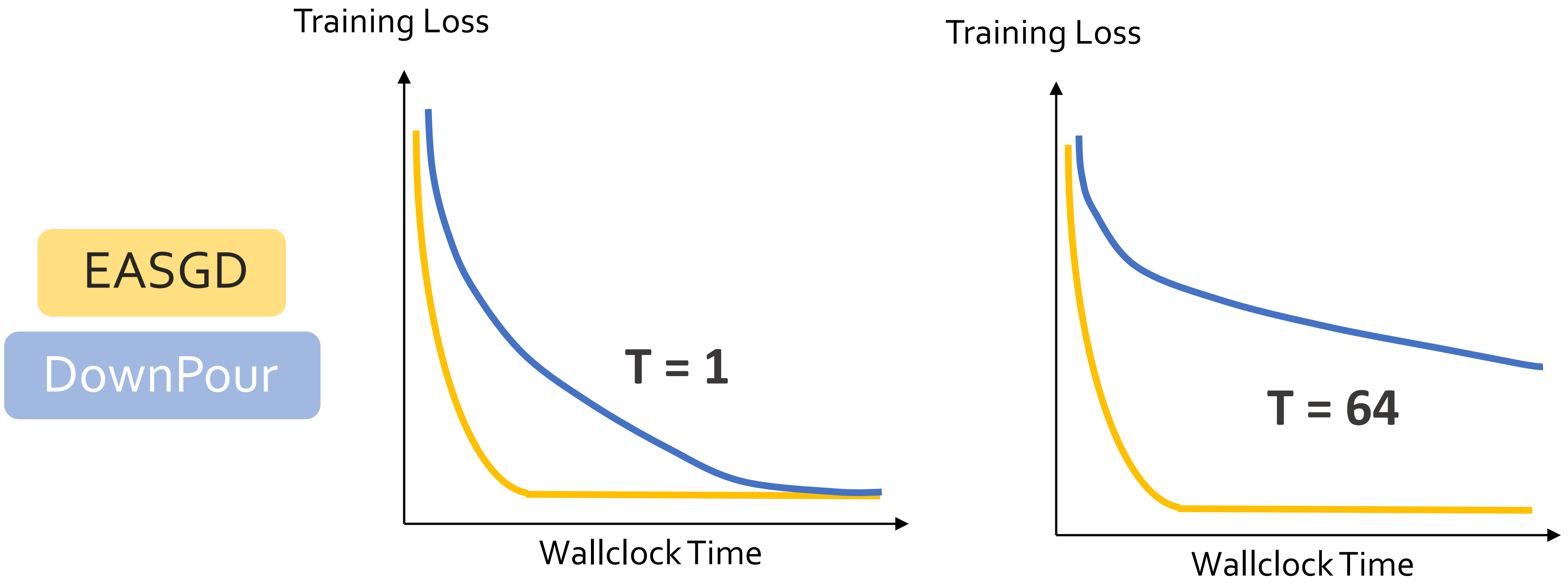
- ✓ 7(or 11)-layer CNN
- ✓ Tested on CIFAR-10 and IMAGENET



(http://cs231n.github.io/convolutional-networks/)

Results on CIFAR-10

P = 4
$\beta = \alpha \times P = 0.9$

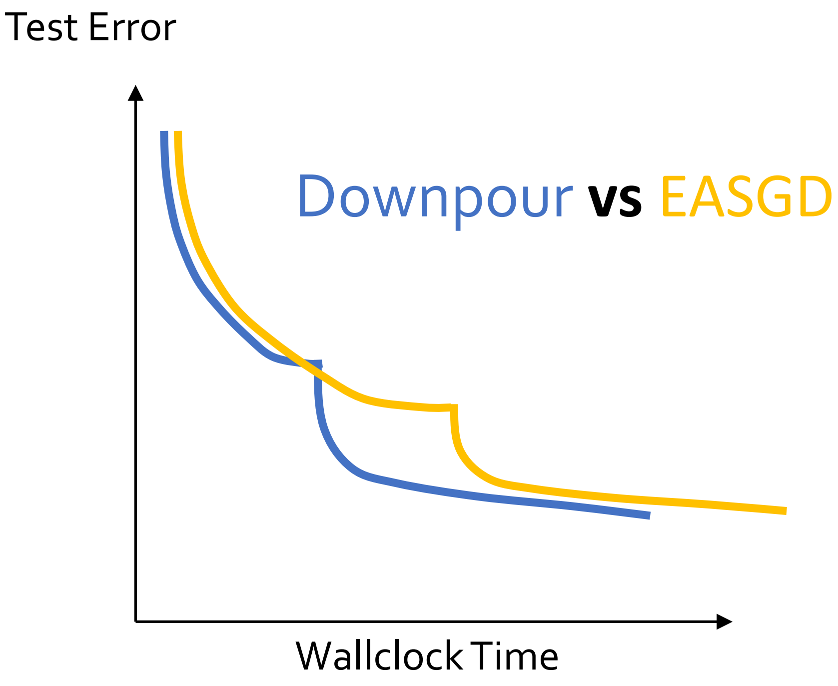


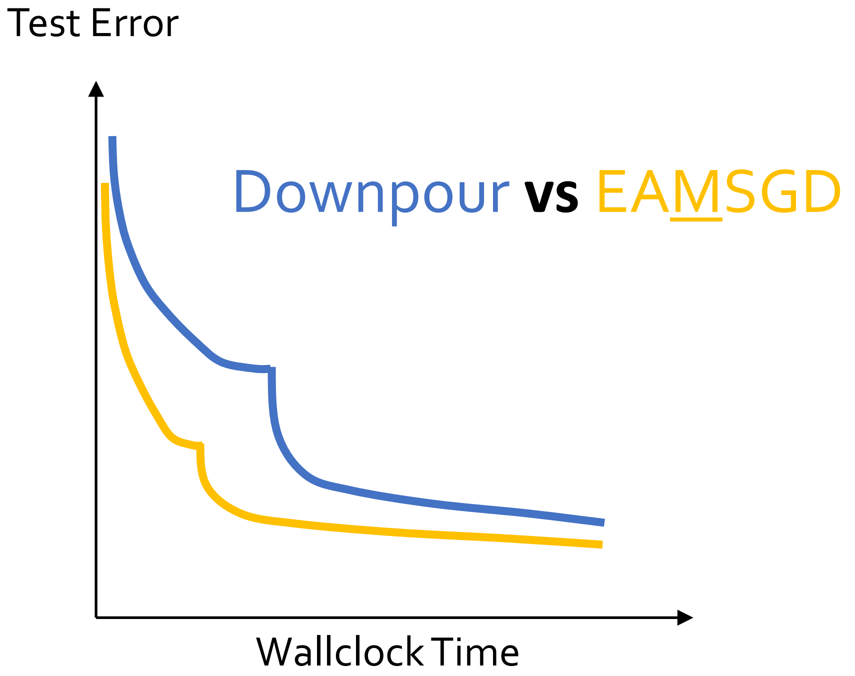
Key Takeaway:

- ✓ EAMSGD significantly outperforms comparator methods for all values of T
- ✓ EASGD can work well even when T = 1000.

Similar Results on ImageNet

P = 4
$\beta = \alpha \times P$ = 0.9
T = 10





Key Takeaway:

✓ EAMSGD significantly outperforms comparator methods.

EASGD is a special case of *Cooperative SGD.*

- ✓ Provided a convergence analysis for non-convex objectives (sync. version)
- ✓ Identified the best choice of the elasticity parameter
- ✓ Generalized the idea of elastic force and developed new comm. efficient SGD variants

"Cooperative SGD: A Unified Framework for the Design and Analysis of Communication-Efficient SGD Algorithms"
Jianyu Wang and Gauri Joshi. arXiv preprint.