Taming the Wild: A Unified Analysis of HOGWILD!-Style Algorithms

Christopher De Sa, Ce Zhang, Kunle Olukotun, and Chris Ré

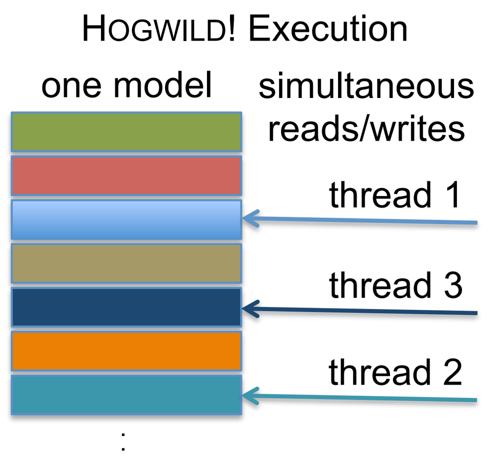
cdesa@stanford.edu, czhang@cs.stanford.edu, kunle@stanford.edu, chrismre@cs.stanford.edu

Departments of Electrical Engineering and Computer Science, Stanford University

Overview

Everyone uses stochastic gradient descent!

- ▷ De facto method for training models in machine learning.
- ▷ Important to run it fast on increasingly-parallel machines.



Common heuristic: asynchronous HOGWILD! execution

- ▷ Run multiple threads of SGD in parallel without locks.
- ▷ Scales very well on modern hardware.
  - – often almost linearly
- ▷ Very widely used.

Wide variety of applications and variants:

- ▷ PageRank approximations (FrogWild!)
- ▷ Deep learning (Dogwild!, DeepDive)
- ▷ Asynchronous stochastic coordinate descent (ASYSCD)
- ▷ Asynchronous stochastic proximal iteration (APPROX)

But it's hard to tell when a HOGWILD! algorithm will work.

- ▷ Can analyze each extension from scratch, but is cumbersome.

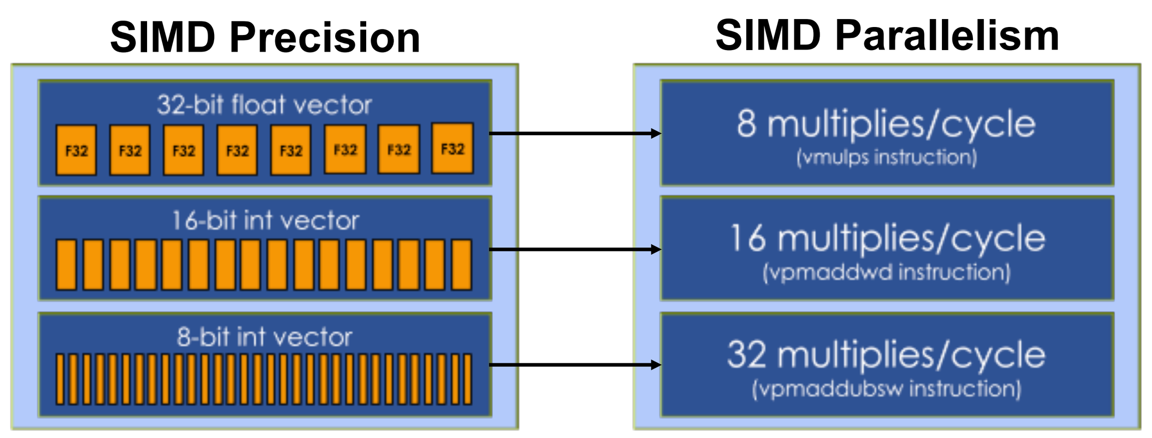
Our contribution: a unified analysis of HOGWILD!

- ▷ Introduce a new ***martingale-based*** result that handles each variant as a different form of noise within a unified model.
- ▷ ***Relax sparsity constraints*** of previous convex results.
- ▷ Derive ***first HOGWILD! convergence results for a non-convex problem***, matrix completion.

Beyond HOGWILD!: Low Precision

We propose BUCKWILD!, a fast heuristic for asynchronous SGD using *low-precision arithmetic*.

- ▷ Low-precision lowers the required memory bandwidth.
- ▷ Also lets us use high-throughput SIMD instructions.



Martingales and Stochastic Gradient Descent

Problem setup.

We're trying to solve stochastic optimization problems of the form

$$\text{minimize } \mathbf{E}[\tilde{f}(x)] \text{ over } x \in \mathbb{R}^n$$

by repeatedly running ***SGD updates***

$$x_{t+1} = x_t - \tilde{G}_t(x_t),$$

where $\tilde{G}_t$ is a random *sample* from some distribution. The goal of the algorithm is to produce, by some time T, a sample in some success region S close to the optimum; if we don't, we say the algorithm has *failed*.

Martingales: The sequential case.

A martingale-based proof for SGD starts with a ***rate supermartingale***, which is a function $W_t : \mathbb{R}^{n \times t} \to \mathbb{R}$ that satisfies the following conditions. First,

$$\mathbf{E}\left[W_{t+1}(x_t - \nabla \tilde{G}_t(x_t), x_t, \ldots, x_0)\right] \leq W_t(x_t, x_{t-1}, \ldots, x_0).$$

Second, if the algorithm hasn't succeeded yet, then

$$W_t(x_t, x_{t-1}, \ldots, x_0) \geq t.$$

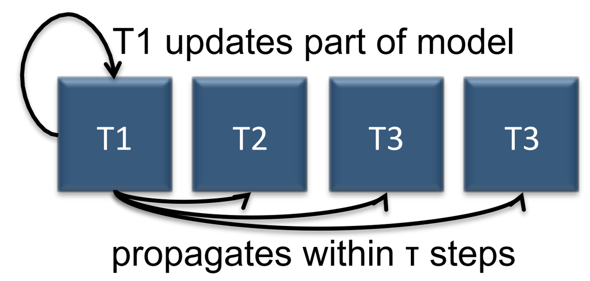
A rate supermartingale immediately lets us ***bound the probability of failure of sequential SGD***:

$$P\left(\text{sequential SGD doesn't succeed before } T\right) \leq \frac{\mathbf{E}\left[W_0(x_0)\right]}{T}.$$

Convergence Rates for Asynchronous SGD

Modeling the hardware.

- ▷ Behavior of HOGWILD! SGD will ***depend on the hardware***
  - – hardware affects rate of race conditions
- ▷ We ***use a parameter τ to abstract away unnecessary details*** about the machine:
  - – number of cores
  - – cache coherence protocol
- ▷ Roughly τ is the number of writes that can be "in flight" at a time.



Main Theorem: HOGWILD! Convergence

Assume some regularity conditions on the rate supermartingale. First, W must be Lipschitz continuous:

$$\left\| W_t(u, x_{t-1}, \ldots, x_0) - W_t(v, x_{t-1}, \ldots, x_0) \right\| \leq H \left\| u - v \right\|.$$

Second, $\tilde{G}$ must also be Lipschitz continuous:

$$\mathbf{E}\left[\left\|\tilde{G}(u) - \tilde{G}(v)\right\|\right] \leq R \left\| u - v \right\|_1.$$

Third, the expected magnitude of an update must be bounded:

$$\mathbf{E}\left[\left\|\tilde{G}(x)\right\|\right] \leq \xi.$$

Then the probability of failure is bounded by

$$P\left(\text{HOGWILD! doesn't succeed before } T\right) \leq \frac{\mathbf{E}\left[W(0, x_0)\right]}{(1 - HR\xi\tau)T}.$$

Example Analysis: Convex Case

Convex case. Assume that f is strongly convex with parameter c, that $\nabla \tilde{f}$ is Lipschitz continuous with constant L, and that $\mathbf{E}[\|\tilde{f}(x)\|^2] \leq M^2$. Let $S = \{x \| x - x^*\|^2 \leq \epsilon\}$. A ***rate supermartingale*** for this problem is

$$W_t(x_t, \ldots) = \frac{\epsilon}{2\alpha c\epsilon - \alpha^2 M^2} \log\left(e \left\| x_t - x^* \right\|^2 \epsilon^{-1}\right) + t.$$

Constructing this rate supermartingale follows from classic analysis of strongly-convex functions, and requires ***no additional work beyond proving sequential convergence***. If we choose step size

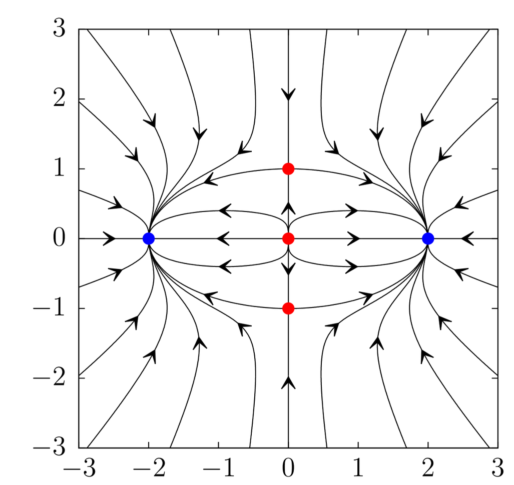
$$\alpha = \frac{c\epsilon\vartheta}{M^2 + 2LM\tau\sqrt{\epsilon}}.$$

we can ***bound HOGWILD!'s probability of failure***:

$$P\left(\text{fail}\right) \leq \frac{M^2 + 2LM\tau\sqrt{\epsilon}}{c^2\epsilon\vartheta T} \log\left(e \left\| x_0 - x^* \right\|^2 \epsilon^{-1}\right).$$

Non-Convex Case

Our analysis is general enough to apply to a non-convex problem.

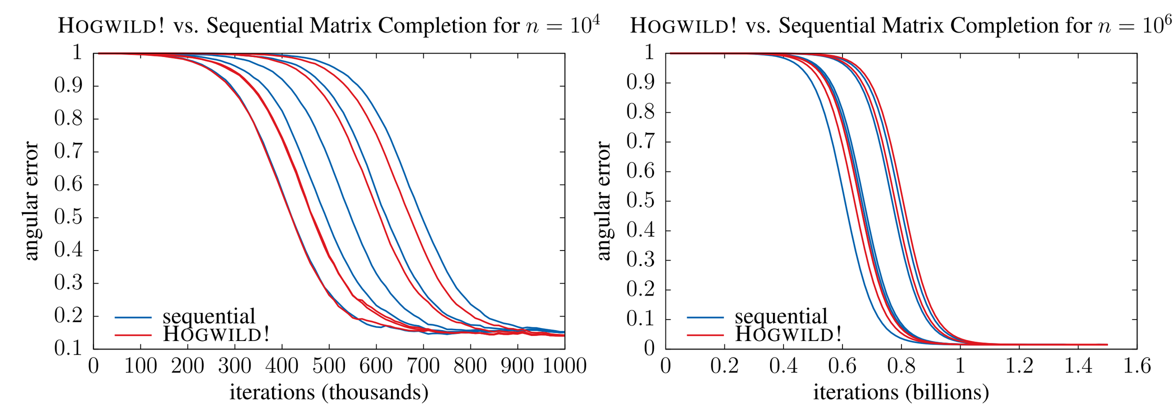


Matrix completion

- ▷ Non-convex because of unstable fixed points.
- ▷ Non-convexity means that ***standard convex analysis of HOGWILD! doesn't apply***.
- ▷ No existing HOGWILD! results.

Because there was an existing martingale-based result for the sequential case, ***our method easily extends it*** to show that ***HOGWILD! works for this problem.***

This is backed up by experiments. Here we compare some trajectories of 12-thread HOGWILD! and sequential SGD on matrix completion — notice that the ***dynamics are basically the same***.



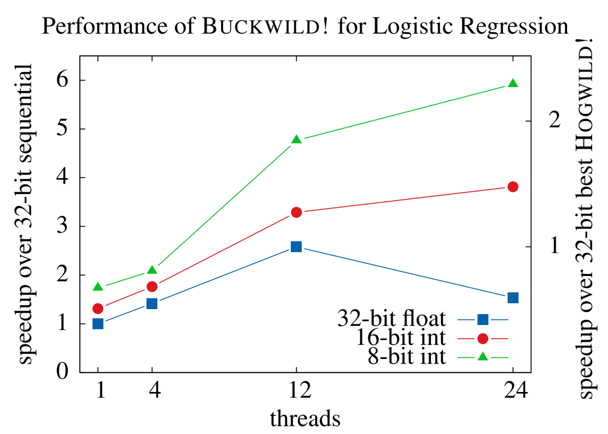
Low-Precision with BUCKWILD!

We ran BUCKWILD!, i.e. ***low-precision asynchronous SGD***, on logistic regression. This table shows the training loss as precision is changed — notice that ***low-precision has no effect on loss***.

Dataset	Rows	Columns	Size	32-bit float	16-bit int	8-bit int
Reuters	8K	18K	1.2GB	0.5700	0.5700	0.5709
Forest	581K	54	0.2GB	0.6463	0.6463	0.6447
RCV1	781K	47K	0.9GB	0.1888	0.1888	0.1879
Music	515K	91	0.7GB	0.8785	0.8785	0.8781

For convex functions with precision κ, our technique gets us

$$P\left(\text{fail}\right) \leq \frac{M^2(1 + \kappa^2) + LM\tau(2 + \kappa^2)\sqrt{\epsilon}}{c^2\epsilon\vartheta T} \log\left(e \left\| x_0 - x^* \right\|^2 \epsilon^{-1}\right).$$



- ▷ Speedup of BUCKWILD! running on dense RCV1 dataset.
- ▷ ***Significant speedup from low precision***.
- ▷ Up to ***2.3× as fast*** as the best HOGWILD!