An Investigation into HOGWILD! [NRRW11]

Abhijit Chowdhary
ac6361@nyu.edu

New York University — May 20, 2020

Contents

1 Introduction

Despite being the subject of much modern excitement, the ideas of gradient descent date all the way back to Cauchy in 1847. And although it's main application has been in the solution of recent big-data optimization problems, stochastic gradient has been around since the 1940s, formally by Robbins and Monro in 1951. In the last two decades, however, modern hardware has begun to see a tapering off of Moore's law, and has begun to expand out in a distributed fashion with multicore processors and GPUs; naturally the question becomes: In order to take advantage of the strengths of modern hardware, how can we parallelize a stochastic gradient method?

2 HOGWILD!

Prior to 2011, parallel stochastic gradient methods had been introduced, but most suffered from poor scaling due to the necessity of locks. A naive implementation could look like:

Algorithm 1 Very Naive Parallel Stochastic Gradient

Require: Number of data points N, seperable loss function $f = \sum_{e \in E} f_e(x_e)$, Initial x.

for epoch $= 1 \to$ MAX_EPOCHS **do**
    #pragma omp parallel for
    for $k = 1 \to N$ **do**
        Choose i uniformly from $\{1, \ldots, |E|\}$.
        #pragma omp critical
            Read current parameters x.
            Compute $\nabla f_i(x)$.
            $x \leftarrow x - \eta \nabla f_i(x)$.
    end for
end for

Note that the version presented above is one with a fixed number of iterations, as the discussion of stopping criteria seems to be similar to that of the stochastic gradient method, and for extremely large data sets, is often heuristic. But it's clear here that such an algorithm would only effectively be parallelizing the unform sample of i in $\{1, \ldots, |E|\}$, and it's overall parallel efficiency would likely be poor. Technically, you can improve the above by replacing the critical section with selective locks on components of x based on the sparsity pattern of $\nabla f_i(x)$, but because the process of acquiring locks is much more expensive than floating point arithmetic, this helps little.

However, in 2011, the article "HOGWILD!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent" by Niu et. al. [NRRW11] proposed a very simple solution to this problem. Remove the locks![1]

[1] Reportedly discovered by accident when *Feng Niu* commented out the locks on his SGD when debugging; I wish my troubleshooting was nearly as effective...

Algorithm 2 HOGWILD!: Asynchronous Stochastic Gradient with replacement

Require: Number of data points N, seperable loss function $f = \sum_{e \in E} f_e(x_e)$, Initial x.

for epoch $= 1 \rightarrow$ MAX_EPOCHS **do**
    #pragma omp parallel for
    for $k = 1 \rightarrow N$ **do**
        Choose i uniformly from $\{1, \ldots, |E|\}$.
        Read current parameters x.
        Compute $\nabla f_i(x)$.
        $x \leftarrow x - \eta \nabla f_i(x)$. ▷ Must be done atomically
    end for
end for

and should we want to sample without replacement[2] the algorithm is easily modified to:

Algorithm 3 HOGWILD!: Asynchronous Stochastic Gradient without replacement

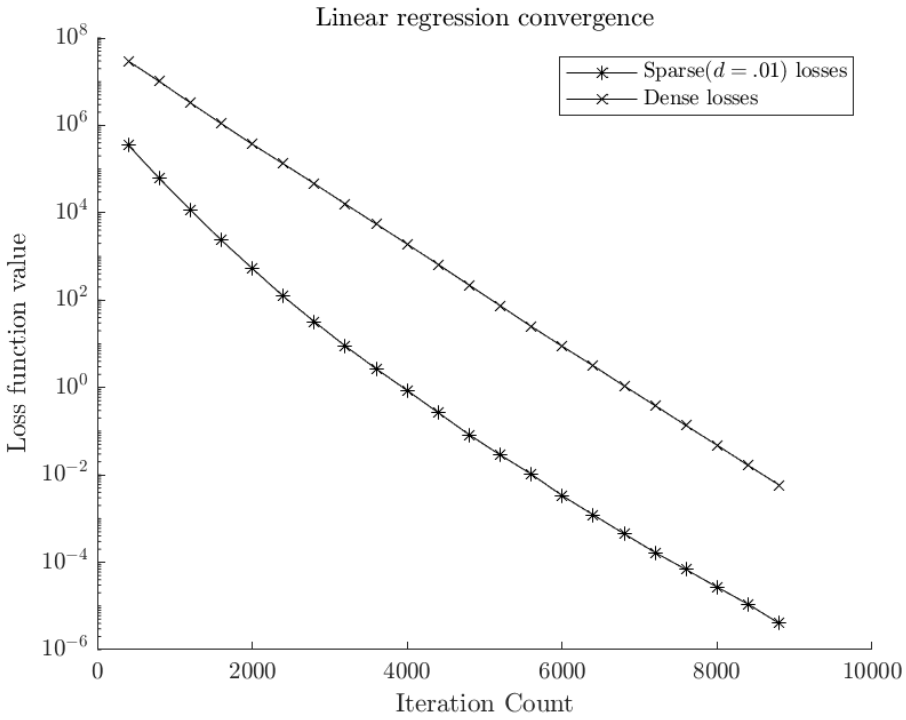
Require: Number of data points N, seperable loss function $f = \sum_{e \in E} f_e(x_e)$, Initial x.

for epoch $= 1 \rightarrow$ MAX_EPOCHS **do**
    Let P be a random permutation of $\{1, \ldots, |E|\}$. ▷ i.e. a Fisher-Yates Shuffle.
    #pragma omp parallel for
    for $k = 1 \rightarrow N$ **do**
        $i \leftarrow P[k]$.
        Read current parameters x.
        Compute $\nabla f_i(x)$.
        $x \leftarrow x - \eta \nabla f_i(x)$. ▷ Must be done atomically
    end for
end for

It should be noted that although the formal OMP locks have been removed, atomic operations are still required in order to prevent mutual exclusion. However, no guards have been placed to prevent a thread from overwriting another's computation midway through, and it's not obvious as to why such a race condition wouldn't destroy the performance of the Stochastic Gradient method. However, with certain guarantees on the sparsity of the function, one can show that this converges in general.

3 Convergence Analysis

a



3.1 Theoretical Results

3.2 Numerical Validation

4 Efficiency Analysis

4.1 Theoretical Results

4.2 Numerical Validation

5 Asynchronous Noise Analysis

5.1 Quantifying the Error

5.2 Banded Matrix Regression

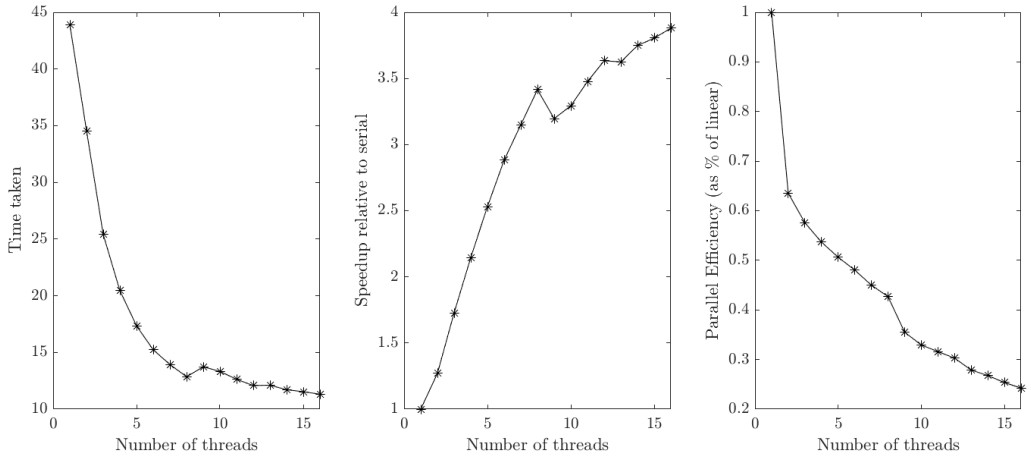
5.3 Impacts of Noise

5.4 Modern Solutions

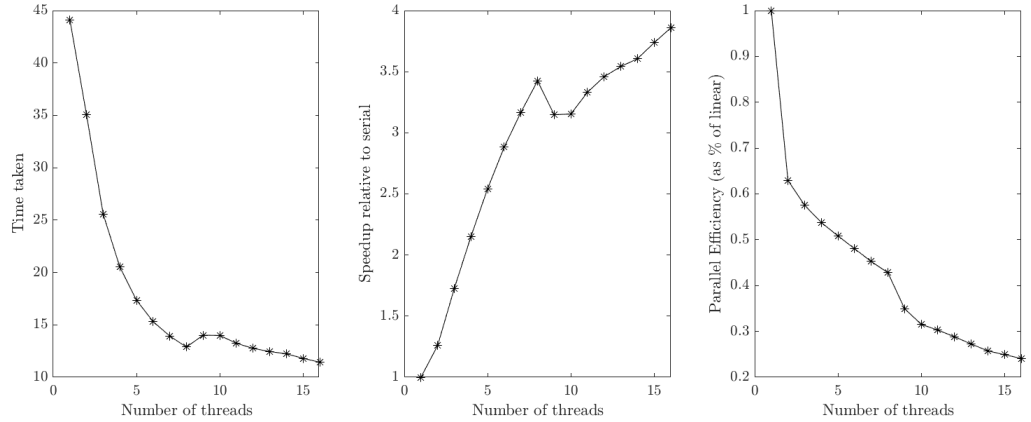
6 Conclusions and Future Work

[2]As it turns out, sampling without replacement provides small amounts of benefit with respect to the asynchronous noise generated, see section 5 for further analysis on the topic.

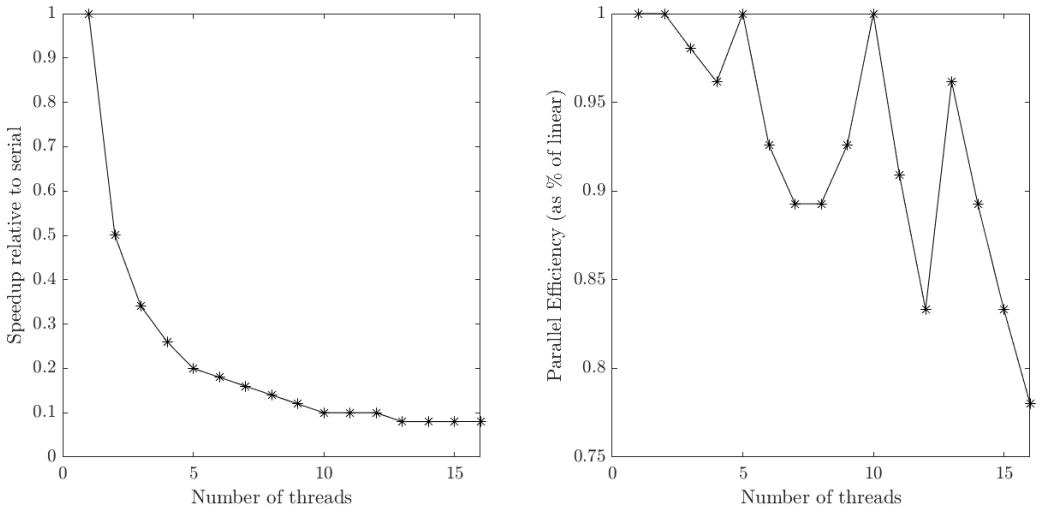
Large regression scaling study with replacement



Large regression scaling study w/o replacement



'Heavy Gradient' Efficiency Study



References

[NRRW11] Feng Niu, Benjamin Recht, Christopher Re, and Stephen J. Wright. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent, 2011.