Final Project: Proposal

Alex Renda (adr74), Matthew Li (ml927), Matt Gharrity (mjg355)

November 7, 2017

Our team would like to focus on parallelizing optimization algorithms for machine learning. Our initial proposal is to implement HOGWILD! [1], but our project idea lends itself to extensions for other algorithms (such as extensions like BUCKWILD! [2]). Our interest in these two algorithms is inspired by Professor Chris De Sa.

HOGWILD! is an algorithm for optimizing stochastic gradient descent. SGD is notoriously difficult to parallel due to it's inherently sequential nature. The HOGWILD! paper has a simple solution to this issue: run SGD in parallel without locks. This scheme relies on the assumption that weight vector updates are sparse so that updates don't constantly overwrite each other. Using exponential backoff in gradient step size, it is provable that there is an upper bound on the convergence rate.

BUCKWILD! is another algorithm similar to HOGWILD! that farther improves performance by using lower-precision arithmetic. BUCKWILD! shows using lower precision arithmetic in the form of round off error introduces additional noise into the system, but does not affect convergence rate. Using lower precision arithmetic by rounding to an integer allows BUCKWILD! to take advantage of SIMD instructions for integers on CPUs, farther improving performance.

Our proposal is to implement both algorithms and explore their performance trade-offs in different problem classes. We are interested in adjusting for sparsity as well as determining the quality of the result in the form of test error. We would also like to see if we are able to replicate the performance improvements specified in their respective papers.

To evaluate our results, we will implement HOGWILD! and BUCKWILD! in several steps. First, we'll implement a fast serial SGD, as a baseline to compare our parallelism against. Next, we'll implement HOGWILD! and BUCKWILD!, and compare all three on a variety of tasks, checking both iteration run time and convergence rate, to evaluate which optimizations resulted in the highest speedup at the lowest accuracy cost. Further, we will run benchmarks on each of our implementations, testing different dataset size, feature vector size, etc., to detect and resolve/trade off bottlenecks in our algorithms.

References

[1] Feng Niu, Benjamin Recht, Christopher Ré, and Stephen J. Wright. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, 2011.

[2] Christopher De Sa, Ce Zhang, Kunle Olukotun, and Christopher Ré. Taming the wild: A unified analysis of hogwild!-style algorithms. In *NIPS*, 2015.