Project Proposal

David Hall Alex Kantchelian

March 1, 2012

1 Introduction

We propose to study the effect of explicitly fault-prone systems on machine learning workloads, in particular focusing on the optimization of convex objective functions commonly used in machine learning using stochastic gradient descent (SGD). Specifically, we will introduce plausible changes into a standard architecture that relax some of the usual correctness guarantees of the processor. Moreover, we will do this while trying to preserve correctness guarantees and convergence rates for the overall algorithm.

We are particularly interested in whether or not commonplace protections against read-after-write, write-after-read, and write-after-write hazards can be (at least) partially relaxed. Protection against these kinds of hazards limit the amount of instruction-level parallelism available, but allowing these hazards may not be fatal to a fault-tolerant machine learning algorithm.

2 Intuitions and Background

Before going further, we quickly review the basic stochastic gradient algorithm. Given a (sub)differentiable function f, SGD seeks to optimize

$$\min_{x \in \mathcal{X}} f(x) + \phi(x) \tag{1}$$

where $f(x)$ is usually called a "loss" function. Typically, in the literature, we are only presented with successive noisy approximations to f (which we label $\tilde{f}$), usually arising from the data itself. Therefore, we introduce $\phi(x)$ as a regularization function that intuitively "smooths" x, making it more resistant to noise in the data.

To actually optimize this function, we take small steps along the (noisy) approximation to the gradient:

$$x_{t+1} \leftarrow x_t - \alpha_t \left(\nabla \tilde{f}(x) + \nabla \phi(x) \right) \tag{2}$$

α_t is a series of step sizes. Because the steps are small, and because the overall objective function is well-behaved, SGD is guaranteed to converge to the optimum, even in the presence of noise in the training data. We believe this noise-tolerance can be used to guarantee convergence under mild to moderate levels of error, even at the hardware level.

3 Previous work

There has been a recent uptick of interest in the area of optimization under error-prone conditions, both from the machine learning community and the systems community.

On the systems side, Sartori et al. [2011] argued empirically for what they term "stochastic processors," which are classical processors that have been designed with the assumption that a certain amount of imprecision is acceptable. In particular, they provide experimental evidence that a variety of algorithms—including certain varieties of SGD—can withstand random bit flips in the floating point units. Moreover, the distribution of errors they induced was consistent with the distribution of errors observed from voltage overscaling. Similarly, Oboril et al. [2011] investigated variations on the conjugate gradient algorithm that were robust to random bit flips.

On the machine learning side, the most relevant work is Hogwild! [Niu et al., 2011]. Essentially, they proposed and analyzed a variant of parallelized SGD where no locking was done in the updates to the weight vector x. For sparse problems where the majority of the components of the vector x are not in use at any particular time, this lock-free approach does not suffer much degradation over locking SGD, and it is of course much faster. Interestingly, however, they are able to specifically prove that their algorithm is guaranteed to converge at rates that are essentially no different from "correct" SGD algorithms.

4 Approach

Our proposed approach consists of three parts. First, we will—in a high-level language—simulate the effect of certain kinds of hardware errors on the convergence rate and performance of different variants of SGD for a few different machine learning workloads. For instance, we might simulate a read-after-write hazard by randomly stalling the actual commit of a value to the weight vector.

Second, depending on what kinds of hazards and what kinds of error levels seem to be empirically feasible, we will investigate modifications to a standard architecture that might increase its error rate to allow the architecture to be faster or more power efficient. We will verify our proposals empirically by modifying a standard architecture in a simulator like Gem5 [Binkert et al., 2011].

Finally, we will try to deploy the theoretical machinery from Niu et al. [2011] or other related papers to prove that the version of SGD we propose is indeed fault tolerant.

References

Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39:1–7, August 2011. ISSN 0163-5964.

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

Fabian Oboril, Mehdi B. Tahoori, Vincent Heuveline, Dimitar Lukarski, and Jan-Philipp Weiss. Numerical defect correction as an algorithm-based fault tolerance technique for iterative solvers. In *Pacific Rim International Symposium on Dependable Computing*, pages 144–153, Los Alamitos, CA, USA, 2011. IEEE Computer Society.

John Sartori, Joseph Sloan, and Rakesh Kumar. Stochastic computing: Embracing errors in architecture and design of processors and applications. In *International Conference on Compilers, Architecture, and Synthesis of Embedded System*, 2011.