Efficient Distributed Stochastic Dual Coordinate Ascent

Mingrui Liu, Jeff Hajewski
Department of Computer Science
University of Iowa
Iowa City, IA 52242
mingrui-liu@uiowa.edu, jeffery-hajewski@uiowa.edu

Abstract

We propose the design and analysis of an efficient, distributed, SDCA algorithm that uses GPUs to improve compute efficiency an asynchronous communication for model updates. This work builds off the previous work of T. Yang [23], M. Li [11], and A. Agarwal [1] creating a more scalable and efficient SDCA algorithm. Specifically, we propose to use CUDA to improve the runtime efficiency of the gradient and parameter update calculations along with the use of MPI for network communication. The asynchronous nature of the communication will be handled using standard MPI facilities combined with threading in C++.

1 Introduction

In recent years, the amount and size of available data has grown at an incredible rate. As the size of the data grows, the challenge of applying standard machine learning algorithms to the data has become increasingly complex. Two common countermeasures used to deal with this are employing stochastic optimization algorithms, and utilizing computational resources in a parallel or distributed manner [4].

In this paper, we consider a class of convex optimization problems with special structure, whose objective can be expressed as the sum of a finite sum of loss functions and a regularization function:

$$\min_{w\in\mathbb{R}^d} P(w), \text{where } P(w) = \frac{1}{n}\sum_{i=1}^{n}\phi(w^\top x_i, y_i) + \lambda g(w), \tag{1}$$

where $w \in \mathbb{R}^d$ denotes the weight vector, $(x_i, y_i), x_i \in \mathbb{R}^d, y_i \in \mathbb{R}$, $i = 1, \ldots, n$ are training data, $\lambda > 0$ is a regularization parameter, $\phi(z, y)$ is a convex function of z, and $g(w)$ is a convex function of w. We refer to the problem in (1) as Regularized Finite Sum Minimization (RFSM) problem. When $g(w) = 0$, the problem reduces to the Finite Sum Minimization (FSM) problem.

Both RFSM and FSM problems have been extensively studied in machine learning and optimization literature. When n is large, numerous sequential stochastic optimization algorithms have been proposed [3, 14, 17, 19, 18, 9, 15, 21, 20, 22, 6, 26, 13, 5, 2, 10], and there also exist several parallel or distributed stochastic algorithms [4, 16, 27, 23, 25, 27, 1]. Specifically, S. Shalv-Shwartz and T. Zhang [19] proposed the Stochastic Dual Coordinate Ascent (SDCA) which provided new analysis with strong theoretical guarantees regarding the duality gap. T. Yang [23, 24] developed two Distributed Stochastic Dual Coordinate Ascent (DisDCA) algorithms and analyzed the tradeoff between network communication between nodes and the difficulty of the performed computation (task). However, the problem of developing a more efficient distributed SDCA algorithm is still open. In this paper, we first provide a GPU implementation of the vanilla distributed SDCA [23], and then give an asynchronous distributed approach to SDCA to make full use of computational resources that scale well.

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

2 Related Work

First we review the related work of sequential stochastic convex optimization for solving FSM and RFSM problems. The first numerical scheme of stochastic optimization stems from stochastic gradient descent (SGD) [3, 14], which was designed to avoid the calculation of full gradient and gets faster convergence than full gradient descent (FGD). To improve the converge rate of SGD, many new algorithms were proposed by exploiting the finite sum structure, including the Stochastic average gradient (SAG) [17], stochastic dual coordinate ascent (SDCA) [19], stochastic variance reduced gradient (SVRG) [9], accelerated proximal coordinate gradient method (APCG) [13], SAGA [7], Prox-SDCA [20], Prox-SVRG [22], and stochastic primal-dual coordinate method (SPDC) [26]. Recently, the optimal first-order stochastic optimization method were developed [2, 10]. Although there exist rich literature studying sequential stochastic optimization with strong theoretical guarantee, less efforts have been devoted to considering them in a parallel or distributed manner. It constitutes a huge gap between theory and practice, since nowadays the size of data increases at a rapid speed, which makes one-core processor or one computer very difficult to handle it properly.

Then we review several related work of distributed optimization algorithms. In the existing literature, many distributed algorithms have been developed on top of stochastic gradient descent (SGD), alternating direction method of multipliers (ADMM), and stochastic dual coordinate ascent (SDCA). The two main approaches used in developing parallel algorithsm for SGD are based on shared memory and distributed memory architectures. Some work [12] looks at both settings, in addition to removing the syncrhonization requirement in the distributed memory setting. A number of approaches [27, 16, 1] consider the asynchronous or lock-free setting, making use of parameter servers [11], sparsity [16], as well as unique data-flow architectures for parameter updates [1]. ADMM stems from [8], which was developed to solve the equality constrained optimization problem. Recently, two independent works of stochastic ADMM were proposed [15, 21]. A standard reference for distributed ADMM is [4]. The advances of SDCA algorithms [19] and its variant [18, 20, 13] enjoy faster convergence than SGD and ADMM, and the distributed SDCA (DisDCA) [23, 24] was developed along with novel analysis of tradeoff between computation and communication. [23] serves as the starting point for our work.

We will build off of work from T. Yang's work on distributed SDCA [23], first add GPU capabilities and then incorporating M. Li's work on parameter servers [11] to handle communication updates in the distributed setting. In [23], SDCA is implemented in a distributed, synchronized fashion. While the achieved results were quite promising, they did not take advantage of hardware acceleration or asynchronous communication. [11] builds an asynchronous communication framework using the concept of parameter servers, which are central data stores for model parameters, and distributed workers working in an asynchronous fashion (i.e., communication and parameter updates are non-blocking operations). Additionally, [1] explores the theoretical ramifications of delayed parameter updates in general distributed stochastic optimization setting. By combining the work of T. Yang, M. Li, and A. Agarwal, we hope to develop a robust, distributed SDCA solution.

3 The Proposed Work

We will approach this problem from both theoretical and implementation perspectives. On the theoretical side we hope to make guarantees on convergence of our proposed approach, while on the implementation side we hope to build a scalable system that can take advantage of its hardware as well as work in a distributed setting.

3.1 Theory

We will try to establish the convergence result of the proposed asynchronous distributed SDCA and analyze the tradeoff between computation and asynchronous communication.

3.2 Implementation

There are two key aspects to the implementation: taking advantage of the GPU and handling the asynchronous communication.

3.2.1 GPU

When available, GPU acceleration will be used via CUDA. We will start by using CUDA libraries such as `cuBLAS` for efficient math operations. Additionally, we will also add CUDA kernels for portions of our algorithms that are easily parallelized in a SIMD fashion.

3.2.2 Asynchronous Communication

Once the hardware acceleration is established, we will turn our focus to building a distributed, asynchronous communication framework. Each comptuer in the cluster will work on a small subproblem whose solution will be used in the parameter update. This result will be sent to a parameter server, which will handle parameter updates and synchronoization across workers. The planned architecture will be similar to that of [11], using a central parameter server that communicates asynchronously with distributed workers. This asyncrhonous communication framework will be built using MPI and C++ threading facilities. Additional libraries may be used such as Eigen for linear algebra.

4 Plan

We will work in parallel, making progress on both the theory and implementation aspects of the project.

4.1 Theory

We will first study the theories and analysis techniques in the existing literature and then try to prove the correctness of naive asynchronous approach. If it works, we will try to establish the convergence rate and analyze the computation and communication cost respectively.

4.2 Implementation

The first part of the implementation approach will be adding CUDA support for the core math operations. The focus is on correctness, rather than optimizing runtime. This is also the simpler portion of the project and will require less time that the asynchronous communciation frmaework.

The second part of the implementation is building the asynchronous communication framework. This will be the most complex part of the implementation and require the most time and effort.

4.3 Comparisions with Other Methods

We plan to test our implementation using two suites of tests: a GPU test suite and a communication test suite. The GPU test suite will compare a CPU-only build with a GPU based build. The communication test suite will compare a single machine model (with GPU acceleration) against a distributed, asynchronous model on a large data set. While it would be ideal to compare the asynchronous framework against a synchronous framework, the technical requirements of adding a synchronous communication framework are beyond the scope of this work.

References

[1] A. Agarwal and J. C. Duchi. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 873–881, 2011.

[2] Z. Allen-Zhu. Katyusha: Accelerated variance reduction for faster sgd. *ArXiv e-prints, abs/1603.05953*, 2016.

[3] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[5] A. Defazio. A simple practical accelerated method for finite sums. In *Advances In Neural Information Processing Systems*, pages 676–684, 2016.

[6] A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.

[7] A. Defazio, F. Bach, and S. Lacoste-Julien. SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. *Nips*, pages 1–12, 2014.

[8] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.

[9] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

[10] G. Lan and Y. Zhou. An optimal randomized incremental gradient method. *arXiv preprint arXiv:1507.02000*, 2015.

[11] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su. Scaling distributed machine learning with the parameter server. In *OSDI*, volume 14, pages 583–598, 2014.

[12] X. Lian, Y. Huang, Y. Li, and J. Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 2737–2745, 2015.

[13] Q. Lin, Z. Lu, and L. Xiao. An accelerated proximal coordinate gradient method. In *Advances in Neural Information Processing Systems*, pages 3059–3067, 2014.

[14] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

[15] H. Ouyang, N. He, L. Tran, and A. G. Gray. Stochastic alternating direction method of multipliers. *ICML (1)*, 28:80–88, 2013.

[16] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

[17] N. L. Roux, M. Schmidt, and F. R. Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. In *Advances in Neural Information Processing Systems*, pages 2663–2671, 2012.

[18] S. Shalev-Shwartz and T. Zhang. Accelerated mini-batch stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 378–385, 2013.

[19] S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013.

[20] S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *ICML*, pages 64–72, 2014.

[21] T. Suzuki et al. Dual averaging and proximal gradient descent for online alternating direction multiplier method. In *ICML (1)*, pages 392–400, 2013.

[22] L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

[23] T. Yang. Trading computation for communication: Distributed stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 629–637, 2013.

[24] T. Yang, S. Zhu, R. Jin, and Y. Lin. Analysis of distributed stochastic dual coordinate ascent. *arXiv preprint arXiv:1312.1031*, 2013.

[25] R. Zhang and J. T. Kwok. Asynchronous distributed admm for consensus optimization. In *ICML*, pages 1701–1709, 2014.

[26] Y. Zhang and X. Lin. Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *ICML*, pages 353–361, 2015.

[27] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.