Newton-ADMM: A Distributed GPU-Accelerated Optimizer for Non-Convex Problems

K Ghosh
School of Electrical Engineering and Computer Science (EECS)
University of Ottawa
Ottawa, Canada
akademik.gk@gmail.com

October 5, 2021

1 Introduction

One of the critical steps of wide range of machine learning applications is estimation of parameters of a model from a given dataset. The problem of parameter estimation often is reduced to the finding a minima of an objective function, which is suitably formulated. The main challenges of today's machine learning models dealing with enormous amount of data directly relate to very large numbers of model parameters - which is as good as high dimensional optimization problems, low generalization errors, large training sets. Given the severity of these challenges a sizeable amount of research effort has been put into finding solutions towards these challenges. The most widely utilized optimization technique in machine learning is gradient descent and its variants, such as stochastic gradient descent(SGD). The gradient descent process, which mainly depend on gradient information are termed as first-order methods.

2 Literature Review

A key challenge in optimization for machine learning problems is the large, often, distributed nature of the training dataset. It may be infeasible to collect the entire training set at a single node and process it serially because of resource constraints (the training set may be too large for a single node, or that the associated data transfer overhead may be large), privacy (data may be constrained to specific locations), or the need for reducing optimization time. In each of these cases, there is a need for optimization methods that are suitably adapted to parallel and distributed computing environments. Distributed optimization solvers adopt one of two strategies – (i) executing each operation in conventional solvers (e.g., SGD or (quasi) Newton) in a distributed environment [8, 9, 11, 12, 15, 16, 22, 27, 30]; or (ii) executing an ensemble of local optimization procedures that operate on their own data, with a coordinating procedure that harmonizes the models over iterations [29]. The trade-offs between these two methods are relatively well understood in the context of existing solvers – namely that the communication overhead of methods in the first class is higher, whereas, the convergence rate of the second class of methods is compromised. For this reason, methods in the first class are generally preferred in tightly coupled data-center type environments, whereas methods in the latter class are preferred for wide area deployments.