Abstract

Qunjie Zhou and Zhenzhang Ye

December 5, 2016

Introduction

Artificial neural network

In the last couple of years, deep learning techniques have achieved huge success in various signal and information processing tasks including computer vision, automatic speech recognition and natural language processing, producing a great deal of state-of-art.

Especially, a variety of deep discriminative models(e.g., deep neural networks or DNN, recurrent neural networks or RNN, and convolutional neural networks or CNN, etc) have been applied to fields of vital importance to modern computer science, and have already brought exiting breakthrough.

Deep learning has such a breathtaking impact, which is not limited to computer science area but to almost every modern technology, that it is valuable to put more investigation and research into it.

Training a neural network

It has been widely agreed that the deeper these networks are, the better they can learn to tackle complicated real-world applications. However, deeper networks also result in highly non-convex optimization problems, which require very good training algorithms.

The traditional back propagation algorithm has the severe problem of being trapped in poor local optima especially when network gets deeper. Recently, stochastic gradient descent (SGD) has been extremely extensively applied for training of deep neural networks. Although SGD performs fairly well in finding a good approximation of global optima, it is difficult to parallelize across machines, which makes learning at large scale nontrivial. Even though there exists several improved versions(e.g., AdaGrad, Adam, Momentum, etc.), these gradient-based approaches still have the problem of scalability and other issues as well, such as vanishing gradients.

The immediate question comes: Is it the best we can do to efficiently learn a neural network? In fact, viewing training a neural network purely as an optimization problem, gradient descent is surely simple but also naive. In the family of optimization, a bunch of more sophisticated methods such as ADMM, PDHG are very popular as well. Therefore, we believe it's definitely worth trying others to see whether we can achieve either better accuracy or faster performance.

As we focus on applying new optimization methods instead of exploring deep network architectures, we decide to start simple with a not deep multi-layer perceptrons (MLP).

Approach

Alternating Direction Method of Multipliers

One promising option to solve a convex optimization problem would be using Alternating Direction Method of Multipliers (ADMM). The main idea of ADMM is to decompose the original problem into several smaller sub-problems which are easier to handle and solve. Recently, quite a few researches have applied ADMM to non-convex problems and achieved impressive results. Compared with back propagation, ADMM is more robust to being trapped in a local optima, since ADMM solves each sub-problem globally.(??) Meanwhile, the decomposition leads to easier parallel programming. Therefore, we decide to apply ADMM to neural network and hope to achieve a dramatic speedups compared with back propagation.

Plan of our project

We would like to start our project with a simple neural network and train it on MNIST database. The neural network consists of L layers. There is a linear operator W_l, which we call weight, between adjacent layers and each neuron is a non-linear neural activation function h_l. (??)Given an vector a_{l-1} and its label y, the goal of neural network is to find the best weight W which minimizes a loss function $\mathcal{L}$. To clarify, we define the problem as following:

$$\begin{aligned} \underset{\{W_l\},\{a_l\},\{z_l\}}{\text{minimize}} \quad & \mathcal{L}(z_L, y) \\ \text{s.t.} \quad & z_l = W_l a_{l-1}, for l = 1, 2, ..., L \\ & a_l = h_l(z_l), for l = 1, 2, ...L-1. \end{aligned} \tag{1}$$

Based on [1], we will try to analyze this problem and apply ADMM to it. Our first aim is to implement the algorithm from [1] and check its feasibility for this problem. If it works for above problem, we will try to analyze this algorithm and improve the performance. Otherwise, reviewing each part of this algorithm and figure out the correct solution will be included. Afterwards, comparing it with back propagation is one part of our project.
After above steps, we can conclude that ADMM is feasible for training neural network. To highlight the advantage of ADMM, we would like to apply it to a more complicated and deeper neural network such as CNN or RNN.
Since the problem in equation (1) is an nonlinear optimization one and we are doing our inter discipline project in Mathematics faculty. We would like to choose the course "Nonlinear Optimization: Advanced [MA3503]". The content of this course like optimality conditions will be helpful for analyzing our problem. Some numerical methods introduced in this course might give us some ideas about solving our problem.