EE 105 Distributed Optimizations Report

Olive Garst

November 2020

1 Introduction

This document summarizes the lecture that was recorded for class on November 10, 2020, on distributed optimizations. This lecture explores the idea of distributed optimizations for strongly convex systems, and the ability for these systems to potentially outperform centralized algorithms. In addition, this idea of distributed systems optimization are especially useful in communications systems where is common to not have all nodes/information in one place.

Distributed optimization takes a cluster of data points and creates a line of best fit, minimizing error, among different nodes. This distributed algorithm does best in convex problems, where control is applicable.

2 Applications of Distributed Optimization

One way to create these unified lines of best fit, or optimization solution, is to perform gradient tracking on every node. Like most optimization problems, distributed optimization algorithms are searching for the global minimum in an loss function curve. This is why strongly-convex problems often yield better for optimization algorithms - the algorithm can iteratively track the gradient, or slope, of this optimization curve until it reaches an area where slope is 0. With deep learning problems, where the optimization curve is not convex (aka doesn't has well defined global extrema based on changes in gradients or slope), distributed algorithms often fail. For the above reasons, when using a distributed optimization algorithm, the best problems to apply these algorithms to have strongly convex optimization curves and are smooth.

3 Basics of Gradient Tracking

One way to change the performance of distributed optimization algorithms is to apply an idea of acceleration. Intuitively, acceleration in optimization problems is this idea of having inertia towards the extrema of the optimization curve.

One equation to track the gradient of a function is as follows:

$$x_{k+1} = x_k - \alpha f(x_k)$$

This equation tracks the gradient and slowly steps towards the minimum, where slope will be close to 0. When the loss function has high slope, less steps will be needed. One can increase the efficient of the gradient tracking by adding some momentum to the system using the equation, which makes the step size closer to the minimum:

$$x_{k+1} = x_k - \alpha f(x_k) + \beta * (x_k - x_{k-1})$$

This method makes searching for the minimum much faster when compared to simple gradient descent algorithms.

4 Distributed Optimization

The application of optimization algorithms to distributed networks adds complexity to the problem. There are many nodes with certain information. The distributed system can be imagined as a directed graph with n nodes. The loss function of a distributed network is to minimize some global loss function, which can be broken down into each nodes' local loss function. Think of this method as finding the gradient at each node (which will have various data points), and then finding the gradient fro the whole system by taking into account each node's gradient equation. This is taking a linear regression of each node's linear regression.

This leads to a problem since there will very likely not be linear regression agreement between each node - leading to muddling of the optimal solution. The minimum of the loss function's gradient will not be 0 when there is not agreement. Therefore, to solve for the minimum of the loss function for a sum of distributed nodes, the optimal solution to the loss function will be when the gradient of the sum of each linear regression, or the sum gradient. However, Khan points out that the value of the distributed system at the point where the sum gradient is 0 will not be 0. This discrepancy is called the local-vs-global dissimilarity bias.

Therefore, the idea of the optimum solution equation is as follows, where F is the global gradient estimate, x star is the estimated solution, and wights are the linear weights of each node:

$$x = \sum weight * x^* - \alpha * F$$

This will move down the global gradient until it approaches 0, when the algorithm will have found the global minimum for convex problems. This algorithm can again be sped up using the heavy ball technique outlined earlier. This algorithm has been proven to converge linearly for some cases (a range of step sizes, for randomly connected graphs, etc).

These ideas can be applied to batch problems, whose goal is to find some classifier to distinguish between nodes. Using distributed optimization algorithms by sampling one value in each node and applying the weighted global gradient descent algorithm, then finding the gradient tracking from the graph created by sampling from each graph (for more info, look up GT-SARAH).

5 Conclusion

As discussed above, gradient descent is a common algorithm to solve for convex loss functions, and can be utilized to find the minimum loss in distributed systems using the right algorithms and approaches. In addition, gradient descent algorithms can be sped up using various methods such as heavy-ball algorithms. It's also important to note that when there is constant step size alpha for convex problems, there is a linear speed up when compared to the centralized solution. The convergence rate is also independent of the problem set up, such as the

graph nodes and connection directions. This means that after some time, the network set up doesn't matter, which I believe cannot be said of centralized solutions.