ID5130: Parallel Scientific Computing

Course Project

Monish Kumar V (CE18B118)

Abstract:

In a typical machine learning problem, given the training data, the goal is to learn the parameters of a function that best fits the given data. Learning this function is useful as it allows us to make future predictions about the data. To achieve this goal, generally, we train a model to minimize the error function — a function that captures how far away the model predictions are from the ground truth.

A popular algorithm used to minimize error functions is the gradient descent algorithm. In gradient descent, we begin from a random estimate of the solution and then update the estimate by taking a finite step along the slope of the function (i.e., the gradient). This process is repeated iteratively until the minimum of the function is reached. Gradient descent is especially useful for convex functions, as it is guaranteed to minimize the error function for a convex function after a finite number of iterations. However, for more complex functions, gradient descent requires many iterations before reaching a good solution.

The stochastic-gradient descent (SGD) algorithm is a variation of gradient descent that addresses some of the limitations of the gradient descent algorithm. In SGD, during each iteration, a random point from the training data is sampled and then a gradient-descent like update is performed with respect to that single point. One of the biggest advantages of SGD is that it has a very small memory footprint and a rapid learning rate thereby making SGD extremely popular for a variety of machine learning tasks.

However, SGD is inherently a serial algorithm. It updates the parameters after seeing every example. Thus, SGD's scalability is limited by its inherently sequential nature and as a result it is difficult to parallelize as each thread must wait for another thread to update before the next iteration. Zinkevich et al. (2010) in their paper "Parallelized Stochastic Gradient Descent" addresses this problem and comes up a with a way to parallelize SGD. So, as part of this course project, I will be implementing the technique mentioned in Zinkevich et al. (2010) using C/FOTRAN.