Parallel Stochastic Gradient Descent

Rishabh Ganesh (rg3478), Yealin Park (yp2611), Yue Sun (ys3535)

What is the problem?

Classification is a canonical machine learning problem that enables machines to predict the category of some query data. The problem is formalized as finding an optimal boundary between data points to decide that data points on one side of the boundary belong to one category, and data points on the other side of the boundary belong to another category. Mathematically, the boundary is represented by a line in two dimensions, a plane in 3-dimensions, and a hyperplane in higher dimensions. Since most real-world classification problems are in higher dimensions, the computation that it takes to determine the best boundary can be taxing, which presents a lot of opportunities for parallelizing the problem. In this project, we plan to classify the Red Wine Quality Dataset (https://www.kaggle.com/datasets/ryanholbrook/dl-course-data?select=red-wine.csv), which classifies wines into different quality categories based on their physicochemical properties, using the stochastic gradient descent algorithm with the supervised machine learning model known as the support vector machine. We are working with an existing sequential implementation of support vector machines using stochastic gradient descent.

Where can we parallelize?

Gradient Descent is used in the support vector machine to find the best parameters of the hyperplane. However, when using all the points in the calculation of loss and derivatives, it requires a large number of vector calculations, so we use stochastic gradient descent to only calculate a mini-batch of the training data, and we can also optimize the speed of training using parallelization. There are two ways of parallelization. One is synchronous, and the other is asynchronous, also known as hogwild. We will compare the performance of the two algorithms and how much speedup they can achieve compared to the non-parallel stochastic gradient descent and find their limitations.

Distributed SGD – Averaging Estimates

Algorithm 3 SimuParallelSGD(Examples $\{c^1, \dots c^m\}$, Learning Rate η, Machines k)

Define $T = \lfloor m/k \rfloor$
Randomly partition the examples, giving T examples to each machine.
for all $i \in \{1, \dots k\}$ **parallel do**
  Randomly shuffle the data on machine i.
  Initialize $w_{i,0} = 0$.
  for all $t \in \{1, \dots T\}$: **do**
    Get the tth example on the ith machine (this machine), $c^{i,t}$
    $w_{i,t} \leftarrow w_{i,t-1} - \eta \partial_w c^i(w_{i,t-1})$
  end for
end for
Aggregate from all computers $v = \frac{1}{k} \sum_{i=1}^{k} w_{i,t}$ and **return** v.

Algorithm 1 HOGWILD! update for individual processors

1: **loop**
2: Sample e uniformly at random from E
3: Read current state x_e and evaluate $G_e(x_e)$
4: **for** $v \in e$ **do** $x_v \leftarrow x_v - \gamma G_{ev}(x_e)$
5: **end loop**