CSCI5451 Assignment 1 Report
Haonan Tian 5229178

For this assignment, we are required to apply parallel program to execute a linear classification model. Specifically, we are trying to find a vector w which if left multiplied by the test data matrix can generated a result that best match the classification label vector. The program applies two iteration to establish the model. The outer iteration is the general iteration for the vector w. For every inner iteration, every vector entry in vector w is updated to new results. Because of the setting of this algorithm, trying to applied parallel algorithm to either outer or inner iteration will be worthless. Since for every update of vector entry in vector w, we need to use the previously updated vector. In other words, every update depends on the previous updates, therefore, one essential feature for this program is that the both outer and inner iterations are all serial and cannot be executed by parallel threads. But this does not mean we cannot apply parallel program to execute this algorithm. If we carefully observe the formula to update the new w entry, we can see that actually for every update, we need to recompute Xi^T * y, Xi^T * X-i and Xi^T * Xi. Therefore, the key for this project is that we can apply parallel methods to first compute all of these three parts mentioned above and stored the results. Then it will be very fast that we can extract the results and directly compute the new value for certain item in vector w.
In my program, I set up two function sets to calculate the result for Xi^T * y, Xi^T * X-i and Xi^T * Xi. Specifically, I construct functions constructPool and computePool to apply parallel program to calculate the results for Xi^T * y and Xi^T * Xi. Then I store the results in a structure called resultPool. For Xi^T * X-i, since the calculation for this item involves in the multiplication of vector and matrix, I applied another set of functions which are constructXTX1 and computeXTX1 to calculate the results. The last component of the project where the parallel methods can be applied is the part to calculate the loss value for each iteration. In my program, I set up a set of functions which are calculateLoss and computeLoss to split the total task to several subtasks and feed each thread a portion of the the matrix(several columns) to compute and combine the results after that.
The following are the results for testing the algorithm with two methods pthreads and openmp with data_large.csv and data_large.label. For each test, the outer iteration is set to 10.

Results for lc_pthreads

Num_threads	Total Run Time(seconds)	Speed_up	Final Loss Value
1	1940.2736		30199.849609
2	1214.0602	x1.60	30201.953125
4	705.9958	x2.75	30202.626953
8	351.5906	x5.52	30202.841797
16	168.6422	X11.51	30202.861328

Results for openMp

Num_threads	Total Run Time(seconds)	Speed_up	Final Loss Value
1	1621.2635		30199.677734
2	1214.1039	X1.34	30202.041016
4	690.4352	X2.35	30202.560547
8	345.1350	X4.70	30202.853516
16	190.1690	x8.53	30203.023438