Dept. of Computer Science and Engineering,

CS69011: Computing Lab 1

Assignment 7: Pthreads Assignment

Maximum Marks: 40

Consider a simple supervised machine Learning problem of binary classification using logistic regression. You have to train a logistic regression model on a large training dataset. Given an input training datapoint (x, y), where $y \in \{0,1\}$ is the label for a binary classification problem and $x \in R^d$ is a real feature vector in d-dimensional space, the probability of the training datapoint (x, y) using the model parameters $w \in R^d$ and $b \in R$ is given by:

$$P(x, y = 1; w, b) = \sigma(w^T x + b)$$

where, $\sigma(a) = 1/(1 + \exp(-a))$, and $w^T x = \sum_i w_i x_i$.
Hence, given a dataset $D = \{(x_i, y_i), i = 1, \dots, n\}$, the negative loglikelihood loss (also known as the cross-entropy loss) is given by:

$$loss(w, b; D) = \sum_i^n -(y_i \log(p_i) + (1 - y_i) \log(1 - p_i))$$

where, $p_i = \sigma(w^T x_i + b)$. Also, the gradient of the loss function is given by:

$$\nabla_w loss(w, b; D) = \sum_{i=1}^n (p_i - y_i) x_i \ \ and \ \nabla_b loss(w, b) = \sum_{i=1}^n (p_i - y_i)$$

For more information please reading the section on logistic regression in the book by Christopher Bishop.

You have to implement the multi-threaded training of logistic regression model using stochastic gradient descent. The mini-batch stochastic gradient descent algorithm is as follows:

```
1. Split the dataset into minibatches $B_1, \dots, B_m$ of equal size (You can pad the last minibatch with random
   datapoints from other minibatches to match the size).
2. For epochs $t = 1, \dots, T$:
    a. For each minibatch $B_i$:
        i. Update $w \leftarrow w - \eta_t \nabla_w loss(w, b; B_i)$
        ii. Update $b \leftarrow b - \eta_t \nabla_w loss(w, b; B_i)$
```

Where $\eta_t = \eta_0/\sqrt{t}$. Note that the order of updates from minibatches is not important, so long as the staleness (number of other updates after reading the current parameter values and before the current update) is small.

Problem 1: (20 marks)

For large scale training, there are two major bottlenecks:

- Large training data
- Computation of the gradient and updation of parameters.

In the first problem, you are going to address the second situation by implementing following types of threads

- 1 - data reading thread, which takes a data reading request from other gradient update threads and reads a minibatch of data to a buffer in memory.
- A pool of k - gradient update threads which takes a minibatch from the memory and computes the update to parameters.
- 1 - Main thread which coordinates the updates from various gradient update threads.

The key tasks of a gradient update thread (performed in a loop) are:

- Synchronize with the main thread and get the next task.
- Synchronize with data reading thread for getting the next minibatch.
- Calculate the update.
- Synchronize with the main thread and update the parameters.

The key tasks of the data reading thread are:

- Wait for next data reading request.
- Read the data into the buffer.
- Notify the thread that data has been read.

The key tasks of the main thread are:

- Spawn all the child threads.
- Wait for update thread to become free and ask for the next update.
- Wait for update requests from update threads and allow update only if the staleness of all other threads is bounded. Otherwise, the update thread remains blocked until the older update has finished.

Problem 2: (20 marks)

In the second part of the problem, the data reading, and updates are done by the same thread. In this case, we have only the two types of threads:

- A pool of *k* – data reading and gradient update threads which reads a minibatch of data to a buffer in memory and computes the update to parameters. We call these worker threads.
- 1 - Main thread which coordinates the updates from various worker threads.

The key tasks of a worker thread (performed in a loop) are:

- Synchronize with the main thread and get the next task.
- Read the next minibatch from the disk in a synchronized manner from an open filehandle.
- Calculate the update.
- Synchronize with the main thread and update the parameters.

The key tasks of the main thread are:

- Open the file handle for the training data file and set up synchronization mechanisms for accessing it.
- Spawn all the child worker threads.
- Wait for a worker thread to become free and ask for the next update.
- Wait for update requests from worker threads and allow update only if the staleness of all other threads is bounded. Otherwise, the worker thread remains blocked until the older update (assigned to a different worker) has finished.

You have to compare the running time for both the above implemenations for a given input training task, for a given number of epochs (T), while varying the size of thread pool (k). Submit a report with the table along with the original codes.

Input format:

You have to read the training data for a binary classification task from the LibSVM standard datasets: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html. For the assignment report results using the Covtype dataset. But for debugging you can use smaller datasets e.g. Australian.

Output format:
The program should print the training set accuracy (which is the fraction of misclassified training data points in after each epoch. You should also print the final values of parameters w and b.