Project 1: Stochastic Gradient Descent for Neural Networks using MPI

Given a dataset

$$\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}, \quad x^{(i)} = (x_1^{(i)}, \ldots, x_m^{(i)}) \in \mathbb{R}^m, \quad y^{(i)} \in \mathbb{R},$$

a neural network with one hidden layer approximates the map from x_i to y_i using the following equation:

$$f(x;\theta) = \sum_{j=1}^{n} w_j \sigma\left(\sum_{k=1}^{m} w_{jk} x_k + w_{j,m+1}\right) + w_{n+1},$$

where

$$\begin{aligned}\theta = (&w_1, \ldots, w_n, w_{n+1},\\ &w_{11}, \ldots, w_{1,m+1},\\ &\ldots,\\ &w_{n1}, \ldots, w_{n,m+1})\end{aligned}$$

is the set of all parameters in the model and $\sigma : \mathbb{R} \to \mathbb{R}$ is a nonlinear activation function. The stochastic gradient method aims to solve the following minimization problem:

$$\min_{\theta} R(\theta) = \frac{1}{2N} \sum_{i=1}^{N} \left| f(x^{(i)};\theta) - y^{(i)} \right|^2.$$

To find the optimal value of θ, we start with an initial guess θ_0, and update the solution with

$$\theta_{k+1} = \theta_k - \eta \widetilde{\nabla_\theta R}(\theta_k), \tag{1}$$

where η is the learning rate, and $\widetilde{\nabla_\theta R}$ is the approximation of the gradient

$$\nabla_\theta R(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left[f(x^{(i)};\theta) - y^{(i)} \right] \nabla_\theta f(x^{(i)};\theta).$$

The approximation is done by randomly drawing M distinct integers $\{j_1, \ldots, j_M\}$ from the set $\{1, \cdots, N\}$ and setting $\widetilde{\nabla_\theta R}$ to

$$\widetilde{\nabla_\theta R}(\theta) = \frac{1}{M} \sum_{i=1}^{M} \left[f(x^{(j_i)};\theta) - y^{(j_i)} \right] \nabla_\theta f(x^{(j_i)};\theta).$$

The random set $\{j_1, \ldots, j_M\}$ must be updated for every iteration (1). The iteration terminates when $R(\theta_k)$ no longer decreases.

This projects requires implementation of the above algorithm using MPI. Any programming language can be used. The implementation must satisfy the following requirements:

- The code should work for any number of processes.
- The dataset is stored nearly evenly among processes, and the algorithm should not send the local dataset to other processes, except when reading the data.
- All processes should compute the stochastic gradient $\widetilde{\nabla_\theta R}$ in parallel.
- Once the solution θ is found, the code can compute the RMSE in parallel.

Apply your code to the dataset `nytaxi2022.csv` (the zip file can be found on Canvas). The description of this dataset can be found at `https://www.kaggle.com/datasets/diishasiing/revenue-for-cab-drivers`. The requirements include:

- Split the dataset into a training set (70%) and a test set (30%).
- Build a machine learning model to predict the total fare amount (column `total_amount`) using the following columns as features:
  - `tpep_pickup_datetime`
  - `tpep_dropoff_datetime`
  - `passenger_count`
  - `trip_distance`
  - `RatecodeID`
  - `PULocationID`
  - `DOLocationID`
  - `payment_type`
  - `extra`

  You may need to preprocess of the data by dropping some incomplete rows and normalizing the data.
- Test your code with at least three activation functions σ and at least five batch sizes M. Choose a suitable number of neurons in the hidden layer m for each set of parameters. Report the following results:
  - The parameters you have chosen.
  - Figures showing the training history (the value of $R(\theta_k)$ versus k).
  - The RMSE of the training data and the test data.
  - Training times for different numbers of processes.
  - Any efforts you have made to improve the result.

Please upload the following documents to Canvas no later than 1 Oct 2025:

- Your complete code and a README file explaining how to use your code.
- A report (in PDF format) including a description of your implementation and the results mentioned above.
- Your communication with AI tools (if any).

At least one member of each group should upload the above documents via Canvas. Please indicate the names and the student IDs of all group members in the report.