The Implementation of SGD for NN using MPI

Project Overview

This project implements a neural network using MPI to predict New York taxi fares. The system consists of two main components: a Python-based data preprocessing code(process.ipynb) and a C-based MPI neural network trainer(mpi.c).

How to Run The Code

The simplest and quickest way

You can decompress the `try` file to obtain a series of files. These files are a series of results we obtained by setting the random number as `123` through the `process.ipynb` code, and then you can run the `mpi.c` program. `process.ipynb` can be run directly For `mpi.c`:

```
# Compilation Command
mpicc -O3 -lm mpi.c -o nn_mpi
# Execution Command
mpirun -np 8 ./nn_mpi 0 32 1 0.01
# Parameter Syntax
./nn_mpi <activation_type> <batch_size> <hidden_units> <learning_rate>
```

A somewhat troublesome method

You can first run the `process.ipynb` file, then make sure the try file is in the correct location, and change the file path in `mpi.c` Then, use the method in `The simplest and quickest way`

Introduction

Introduction of The Files

We show the introduction of files in the project directory:

File/Folder Name	Description
`process.ipynb`	Python-based data preprocessing notebook
`mpi.c`	C-based MPI neural network trainer
`try/`	Directory containing generated data files
`try/train_part_*.csv`	Training data partitions (eight in total)
`try/test_part_*.csv`	Test data partitions (eight in total)

try/embedding_*.npy	Embedding matrices for high-cardinality categorical features (two in total)
try/label_encoder_*.pkl	Label encoders used for high-cardinality categorical features (two in total)
try/one_hot_categories_*.pkl	Category mappings for one-hot encoding low-cardinality features (three in total)
try/metadata.json	The original data recorded
Report.pdf	Directory containing project report and documentation

Introduction of The Files in MPI.c

Parameter	Default Value	Valid Range	Description
activation_type	0	0-2	Activation function type: 0=Sigmoid, 1=ReLU, 2=Tanh
batch_size	512	Positive integer	Number of samples per training batch
hidden_units	64	Positive integer	Number of neurons in the hidden layer
learning_rate	0.01	Positive float	Step size for gradient descent optimization
EPOCHS	20	Fixed	Maximum number of training epochs
Random seed	123	Fixed	Fixed random seed(you can set it by yourself)

Introduction of Output

Initialization Information

```
Process 0/8 started
Process 1/8 started
Process 2/8 started
...
Process 0: Loaded 3209391 training samples
Process 1: Loaded 3209391 training samples
...
Process 2: Loaded 1375453 test samples
Process 2: Using 3209216 training samples after truncation
...
Running with activation: 0, batch size: 32, hidden units: 64, learning
```

```
rate: 0.0100
```

Description: Shows MPI process initialization and process count Displays training data loading status for each process Prints configuration parameters (activation function, batch size, etc.)

Training Process Output (Per Epoch)

```
Epoch 1, Loss = 0.499917533019
Epoch 2, Loss = 0.205161123839
Epoch 3, Loss = 0.097319935386
...
Epoch 20, Loss = 0.060203482138(for example)
```

Description: Shows training progress with epoch number and loss value Loss values display Includes early stopping notification when triggered

Prediction Results Output (First 5 samples per process)

```
Rank 0: TRAIN - First 5 samples predictions:
  Rank 0: TRAIN Sample 0: y_true=-0.579046, y_pred=-0.550333,
err=0.028713
  Rank 0: TRAIN Sample 1: y_true=2.096578, y_pred=2.449639, err=0.353061
  Rank 0: TRAIN Sample 2: y_true=0.496222, y_pred=0.524391, err=0.028169
  Rank 0: TRAIN Sample 3: y_true=-0.185378, y_pred=-0.117786,
err=0.067592
  Rank 0: TRAIN Sample 4: y_true=-0.606525, y_pred=-0.402305,
err=0.204220
...
```

Description: Displays predictions for first 5 samples of each dataset Shows true value (y_true), predicted value (y_pred), and error (err) Indicates SSE calculation completion for each dataset

Performance Metrics Output

```
Training time: 307.33 seconds
Training RMSE: 0.346685034691
Test RMSE: 0.348314731462
```

Description: Training duration in seconds Root Mean Square Error (RMSE) for training dataset Root Mean Square Error (RMSE) for test dataset

File Output

File: `training_history.csv` CSV file containing epoch numbers and corresponding loss values Useful for plotting training curves and analysis