COP 6526
Homework 2 Solution

September 21, 2025

Arman Sayan

---

1 Question 1: Parallel k-Means Clustering [50 points]

Dataset: MNIST digits (subset 10,000 images) (or sklearn.datasets.load_digits)

1. [12 points] Implement k-means clustering (sequential code). k=10.

> **Ans:**
> Please check the source code included in the .zip file named as
> **cop_6526_hw2_q1_sequential_arman_sayan.py**
> for the sequential implementation of k-means clustering.

2. [12 points] Parallelize both E-step (assignment) and M-step (update) using Python Multiprocessing. You may split dataset among processes. During the E-step (assignment), each process assigns points to nearest centers. During the M-step (update), each process handles a cluster by computing the mean of its assigned points.

> **Ans:**
> Please check the source code included in the .zip file named as
> **cop_6526_hw2_q1_multiprocessing_arman_sayan.py**
> for the parallel implementation of k-means clustering using Python Multiprocessing.

3. [14 points] Parallelize both E-step (assignment) and M-step (update) using mpi4py.

> **Ans:**
> Please check the source code included in the .zip file named as
> **cop_6526_hw2_q1_mpi_arman_sayan.py**
> for the parallel implementation of k-means clustering using mpi4py.

4. [12 points] Answer the following questions:
   (a) From your experiments, does increasing the number of processes in multiprocessing make the program run faster?
   (b) Does increasing the number of processes in mpi4py make the program run faster?
   (c) Does changing the number of processes have any effect on the accuracy of the K-means clustering results?"

Ans:

2 Question 2 [50 points]

Please parallelize the following nonlinear regression program using multiprocessing and MPI in python. According to the number of CPU cores in your computer (or one computer on stokes), you may use the same number of processes.

Here are the nonlinear regression program and data.

Hints:

(1) To parallelize the training, you can distribute the data to different processes at the beginning. Then under every epoch, every process calculates and summarizes gradients, and then sends them to the main process. Then, in the current epoch, the main process collects gradients from all processes, updates the model, and then propagates the model to every process. Here we do not consider mini-batch. Such kind of distributed/parallel machine learning is "synchronized", which produces the same result as sequential execution.

(2) Please use Numpy for efficiency.

1. [20 points] Multiprocessing in Python

   Ans:

2. [20 points] MPI in Python

   Ans:

3. [10 points] Please compare the entire runtime of the three different implementations (sequential, multiprocessing, MPI) in a report.

   Ans: