Computing at Scale in Machine Learning: Distributed computing and algorithmic approaches (14038)

B-TU Cottbus-Senftenberg · WiSe 2023
Prof. Dr. Alexander Schliep, Nathalie Gocht, M.Sc. and Aleksandra Khatova, M.Sc.

Problem set 1 from October 25, 2023 · Due on November 3, 2023

There is a detailed tutorial about using the command line at `https://ryanstutorials.net/linuxtutorial/` . In particular if you have little experience with Unix or Linux systems, I recommend reading sections 1. The Command Line, 2. Basic Navigation, 3. More About Files, 5. File Manipulation, 9. Filters, 10. Grep and Regular Expressions, and 11. Piping and Redirection of the latter.

Problem 1 (Linux File Manipulation, 6 Points)**.** The goal of this problem is to familiarize yourself with working with the command line and specifically with some command line tools which help you with downloading and manipulating data.

a) Find out which Unix command allows you to download files and download a data file from `https://schlieplab.org/Static/Teaching/DIT852/private-healthcare-percent-gdp.csv` using the command line. The file contains health-care spending as a percentage of GDP for year from 1960-2016 for a number of countries.

b) Create a new file containing the data from `private-healthcare-percent-gdp.csv` without the header lines and excluding "ABW". The first line of the new starting with the country code "AFG".

c) Compute how many different countries are in the new file.

d) Create another new file only containing only the Country Code and the health care spend- ing from 2004 until 2014 and also omits the header lines in the file.

e) List the top ten spenders on health-care in 2004 in decreasing order on the terminal.

f) Create yet another file only containing the data from the nordic countries for all years.

Problem 2 (Monte Carlo, 7 points)**.** Investigate the Monte Carlo simulation for computing π.

a) Measure the proportion of serial computation in `pi_montecarlo.py` . Indicate which parts of the program you consider to be executed serially, and which parts are executed in parallel. Measure the time of each of those sections using just one core for sequential execution, and compute the amount of serial computation that does not benefit from parallelization.

b) The maximal theoretical speedup when parallelizing using k cores is k when neglecting serial parts of the program and otherwise given by Amdahl's law for a measured proportion of serial computation. A graph displaying both the measured and the theoretical speedup for $k = 1, 2, 4, \ldots$ cores is a good way to display the efficiency of the implementation. Produce such a plot for our Monte Carlo simulation. Write a Python script for creating such plots for further submissions.

c) Make runs of the code repeatable by specifying an explicit random number generator seed using a command line option `--seed value` . Look up how the random number generators are seeded in the code and extend it to use an explicit seed.

d) How do you need to set the explicit seeds for the multiprocessing solution?

Note: Please hand in:

- A PDF containing:
  - commands and output for problem 1)
  - your explanation how you measured the serial parts of the code for 2.a) a graph for problems 2.b)
  - your suggestion for random seeding 2.c)
  - your answer to 2.d)
- A Python script for creating the graph for problem 2.2 named problem-2-b.py
- A Python script named problem-2-c.py

Python code must be handed in as '.py'. Code submission in PDF will be graded with 0 points. Your code files can be accompanied with PDF. Please use the naming as suggested above.

You are allowed to use the example solutions provided where they might be helpful for answering question on this problem set.

Please feel free to discuss the problems and approaches to solving them with other students. However *you* should understand results of discussions well enough to write up the solution by yourself and possibly explain the solution. When you do collaborate, please make other's contributions clear. Present your solutions concisely.