Homework 11 Solution – Yidan Xu

Parallel statistical computing.

All modern computers, from a basic laptop to a node on a computing cluster, have multiple cores. Parallelizing your code (i.e., taking advantage of multiple cores) can enable computations too large for one core. Write brief answers to the following questions, by editing the tex file available at `https://github.com/ionides/810f21`, and submit the resulting pdf file via Canvas.

1. Trends in statistical computing are driven by trends in hardware. Why is this leading to a growing role for parallel computing?

   `https://en.wikipedia.org/wiki/Parallel_computing`

   Since nowadays' desktop computer manufactures started to produce power efficient processors with multiple cores to deal with the problem of power consumption and overheating the major CPU of the machine, since multi-core processors each core is independent and can access the same memory concurrently.

2. Some key terms for parallel computing are: process, thread, core, node. Briefly define these in your own words.

   `https://en.wikipedia.org/wiki/Parallel_computing`

   Process: running of tasks on processor; Thread: Subtasks in a parallel program; Core: individual processing units within CPU. Node: stand-alone computer in a network.

3. What common statistical computing tasks are embarassingly parallel?

   `https://en.wikipedia.org/wiki/Embarrassingly_parallel`

   Empirical risk (negative log-likelihood for iid sample) estimation using Monte Carlo.

4. A basic tool for embarassingly parallel computing in R is `foreach`. This is now part of the `doParallel` library included in base R. Run the following R codes for generating 10^8 standard normal random variables, on your laptop or some other machine. Explain the relative speeds. The "elapsed" component of the run time is the total time, in seconds, and is the primary outcome of interest. If you like, you can read more about foreach at

   `https://cran.r-project.org/web/packages/foreach/vignettes/foreach.html`

```
library(doParallel)
registerDoParallel()

system.time(
 rnorm(10^8)
```

```
) -> time0

system.time(
  foreach(i=1:10) %dopar% rnorm(10^7)
) -> time1

system.time(
  foreach(i=1:10^2) %dopar% rnorm(10^6)
) -> time2

system.time(
  foreach(i=1:10^3) %dopar% rnorm(10^5)
) -> time3

 system.time(
  foreach(i=1:10^4) %dopar% rnorm(10^4)
) -> time4

rbind(time0,time1,time2,time3,time4)
```

The first one not using parallel is slowest among all. Comparing approach 2-4, the second gives the quickest runtime, which is having 10^2 parallel tasks $rnorm(10^6)$. This is likely due to the trade-off of constrained number of tasks that can be run on one core and the time taken to run one task.

5. What common statistical computing tasks could benefit greatly from using simple parallelization such as `foreach`?

   Simulations – when exploring model properties. Multiple-chains when running MCMC.

6. Once you are using multicore computing on your laptop or desktop, the next step for additional computing resources is greatlakes (`https://arc-ts.umich.edu/greatlakes/`), which we will use next week. Previous experience with cluster computing in this group ranges from novice to expert: briefly describe any previous experience you have had with computing on a cluster.

   I have ran programmes on department's cluster and AWS cloud before. However, I have 0 expierience in setting up environments (e.g. if I want to have specific versions of packages) on clusters and how to manage them. It would be nice if we can talk about this in the class.

7. A popular data science parallel computing approach is Hadoop with MapReduce (`https://en.wikipedia.org/wiki/Apache_Hadoop`). Do you have suggestions on what parallel statistical computing tasks are more appropriate for Hadoop than for `foreach`?

NLP – handling enormous copra of text.