Homework 11. Due by 11:59pm on Sunday 11/30.

Parallel statistical computing on GPUs using Jax.

This homework introduces Jax, a Python library offering just-in-time compilation and parallelization across CPU and GPU hardware. If you have used Jax before, please share your insights. If not, enjoy taking it for a test drive! You may want to browse `https://docs.jax.dev/en/latest/jax-101.html`. Write brief answers to the following questions, by editing the tex file available at `https://ionides.github.io/810f25/`, and submit the resulting pdf file via Canvas.

Set up Jax on greatlakes. The following code obtains an interactive GPU session on a greatlakes login node:

```
salloc --account=stats_dept1 --partition=gpu --gpus=v100:1 --cpus-per-gpu=1
```

Then, once we have a terminal prompt on the allocated machine, we can carry out a parallel computation similar to Homework 9, running the code in test.py in the class git repository. For good practice, we set up a virtual python environment.

```
module load python
python -m venv .venv-jax
source .venv-jax/bin/activate
pip install --upgrade pip
pip install -U "jax[cuda12]"
python test.py
```

1. Report on any issues encountered running test.py. Hopefully it is straightforward.

   YOUR ANSWER HERE.

2. How does the run time compare to the CPU parallelization we did for homeworks 9 and 10?

   YOUR ANSWER HERE.

3. Inspecting test.py you may find unusual features. Why do we need `block_until_ready` and `jax.random.split`? Is there anything else in the code that you are curious about?

   YOUR ANSWER HERE.

4. What statistical tasks is Jax appropriate for? If you check online, you can find what statitical tasks Jax is currently used for.

   YOUR ANSWER HERE.