CSCE 735 Fall 2021

Major Project

Due: 11:59pm Wednesday, December 15, 2021

(No late submission allowed)

Gaussian Process Regression can be used to predict the values of a function at a point from observations at other points in the domain. As an example, consider an m x m grid of points on a two-dimensional unit square. Node coordinates are given by

$$(x_i, y_j) = (ih, jh), i = 1, \dots, m, j = 1, .., m, \quad (1)$$

where h = 1/(m+1) is the mesh width. Observed data value at the point $r(x_i, y_j)$ is given by the function

$$f(x_i, y_j) = 1 - \left((x_i - 0.5)^2 + (y_j - 0.5)^2\right) + d_{ij}, \quad (2)$$

where d_{ij} is a random value between [–0.05,0.05]. For the remaining discussion, it helps to think of the m^2 grid points as a list of $n = m^2$ points obtained by concatenating $(x_i, y_j), i = 1, \dots, m, j = 1, .., m,$ in that order.

The predicted value of the function at a prediction point $r^*(x, y)$ is given by

$$f^* = {k_*}^T (tI + K)^{-1} f, \quad (3)$$

where f is an $n \times 1$ vector. K is a matrix representing correlation between the data points and is defined as follows: for two points $r(x, y)$ and $s(u, v)$,

$$K(r, s) = \frac{1}{\sqrt{2\pi}} e^{-\left(\frac{(x-u)^2}{2l_1^2} + \frac{(y-v)^2}{2l_2^2}\right)}, \quad (4)$$

where *e* is the base of natural logarithm. Note that K is an $n \times n$ matrix where elements in row *r,* from left to right, correspond to values obtained using (4) for points s for $(x_i, y_j), i = 1, \dots, m, j = 1, .., m,$ in that order. The one-dimensional vector k_* for the prediction point *r*(x,y)* is computed as given below:

$$k_* = \frac{1}{\sqrt{2\pi}} e^{-\left(\frac{(x-u)^2}{2l_1^2} + \frac{(y-v)^2}{2l_2^2}\right)}, \quad (5)$$

for all grid points $s(u, v)$ where the ordering of elements of k_* is similar to f.

Note once again that for an $m \times m$ grid, $n = m^2$. Thus, the vectors f, f^*, k_* in (3) are $n \times 1$ and the matrix K is $n \times n$. Further, t is a noise parameter that is set to 0.01.

We also introduce two hyper-parameters l_1 and l_2 that are be chosen to maximize the likelihood of the prediction being accurate for the given data. In this project, we will use $l_1 = l_2 = 1$.

1. (70 points) In this project, you have to develop GPU code to compute f(x,y) for a given point (x,y). The code should use a **single** processor of the device but should be parallelized to exploit all the cores within the processor. The code should initialize the grid points and observed data values using equations (1) and (2) on the host and move these values to the GPU device. Next, the matrix A = (tI+K) should be computed on the device. This should be followed by LU factorization of A on the device. These factors should be used to compute the solution of the system Az=f using the L and U factors obtained in the previous step. Finally, the predicted value

should be computed as $f(x, y) = k^T z$. You must develop your own code to compute the LU factors and to solve the triangular systems. LU factorization can be replaced by Cholesky factorization, which is a more efficient algorithm for symmetric positive definite matrices.

2. (20 points) Describe your strategy to parallelize the algorithm for a **single** multiprocessor of the GPU. Discuss any design choices you made to improve the parallel performance of the code.
3. (10 points) Compute the flop rate you achieve in the factorization routine and in the solver routine. Compare this value with the peak flop rate achievable on the processor, and estimate the speedup obtained over one core and the corresponding efficiency/utilization of the cores on the device. You may choose appropriate values for the grid size to study the features of your implementation.

Submission: You need to upload the following to Canvas:

1. Submit the code you developed.
2. Submit a single PDF or MSWord document that includes the following.
    - Responses to Problem 1, 2, and 3. Response to 1 should consist of a brief description of how to compile and execute the code on the parallel computer

Helpful Information:

1. Information on compiling and running CUDA programs on a GPU for Grace is available in the HPRC user guide (See https://hprc.tamu.edu/wiki/Grace).
2. To develop code interactively, log on to one of the GPU-equipped login nodes on Grace; these nodes are named graceX.hprc..tamu.edu, where X is to be replaced by the numbers 1, 2, or 3 (See https://hprc.tamu.edu/wiki/Grace:Intro#Login_Nodes). If you use portal.hprc.tamu.edu to access Grace, you are randomly assigned a node where X can be 1-5. You will then need to use ssh to log into a GPU-equipped login node from the initial shell.
3. Check out https://hprc.tamu.edu/wiki/Terra:Compile:All#CUDA_Programming on how to compile and execute your code. Current recommendations are listed below.
    - Load the modules for compiler and CUDA prior to compiling your program. Use:

      ```
      module load intel CUDA/8.0.44
      ```

    - Compile C programs using `nvcc`. For example, to compile `code.cu` to create the executable `code.exe`, use

      ```
      nvcc -ccbin=icc -o code.exe code.cu
      ```

4. The run time of a code should be measured when it is executed in dedicated mode. Sample batch files for Grace are available at https://hprc.tamu.edu/wiki/Grace:Batch#Job_File_Examples. To execute the code on a GPU-equipped node, you must use the submission options for GPUs.