gpuPANDA and gpuLIONESS: Accelerated inference of gene regulatory networks using the Graphical Processing Unit (GPU)

Author:

Marouen Ben Guebila

Introduction

PANDA [1] infers gene regulatory networks between Transcription Factors (TFs) and their target genes through finding agreement between three experimental networks that are 1) gene-gene co-expression, 2) TF-TF PPI and 3) TF-gene binding as determined by Position Weight Matrices (PWMs).

The computation of the similarity between the data sources relies heavily on large matrix operations, which makes these sections of the PANDA algorithm particularly amenable to Graphical Processing Unit (GPU) accelaration. The current implementation uses MATLAB gpuArrays that invokes CUDA processes underneath the hood.

Requirements

gpuPANDA is compatible with NVIDIA GPUs. Please make sure to install the proper hardware driver from https://www.nvidia.com/Download/index.aspx.

If everything works fine, the following command should display that GPU device was detected and functional

```
gpuDevice
```

Also, make sure to clone netZooM (> 0.4.3) https://github.com/netZoo/netZooM and add the the folder to matlab path and save it

```
addpath(genpath('pathTonetZooM'))
savepath
```

Computing regulatory networks using gpuPANDA

Using gpuPANDA is very starightforward, all you have to do is to set the computing argument to PANDA.

```
computing='gpu'
```

We need to set the regular argument to PANDA, for this tutorial, we will use a small toy network

```
exp_file   = '../tests/test_data/expression.txt';
motif_file = '../tests/test_data/motifTest.txt';
ppi_file   = '../teststest_data/ppi.txt';
panda_out  = '';  % optional, leave empty if file output is not required
save_temp  = '';  % optional, leave empty if temp data files will not be needed afterwa
lib_path   = '';  % path to the folder of PANDA source code
alpha      = 0.1;
save_pairs = 0;%saving in .pairs format
modeProcess= 'intersection';
respWeight = 0.5; % obtain regulatory network through averaging half the availability r
absCoex    = 0; % take the absolute value of the coexpression (1) or the signed coexpre
```

```
similarityMetric= 'Tfunction'; %use modified Tanimoto to compute agreement between data
precision   = 'double'; % could be single (7 decimal precision) or double (15 decimal pr
verbose     = 1; % To display the iterations
```

In addition, a parameter controls the memory usage on the GPU

```
saveGPUmemory=0;
```

Finally, we can proceed to call panda

```
% Call Panda
tic;AgNet = panda_run(lib_path,exp_file, motif_file, ppi_file, panda_out,...
            save_temp, alpha, save_pairs, modeProcess,respWeight, absCoex,...
            similarityMetric, computing, precision, verbose, saveGPUmemory);toc;
```

When computing large networks, some GPU devices are unable to accomodate large matrices. In this case, we can use

```
saveGPUmemory=1;
```

This parameter allows in-memory computation and considers only one half of the gene-gene coexpression matrix which is symmetrical. But this transformation results in slower run times.

```
tic;AgNet = panda_run(lib_path,exp_file, motif_file, ppi_file, panda_out,...
            save_temp, alpha, save_pairs, modeProcess,respWeight, absCoex,...
            similarityMetric, computing, precision, verbose, saveGPUmemory);toc;
```

Another strategy to save memory and computation is to reduce the precision of the netwrok from 15 digit after the decimal with double precision to 7 digits after the decimal with single precision.

```
precision='single';
```

The increase in speed and decrease in memory requirements come at the expense of precision.

Computing single-sample regulatory network with gpuLIONESS

When we would like to compute several PANDA networks in parallel when can use the multi-GPU specifications of modern devices, in a way that each GPU computes a PANDA network in parallel. MATLAB calls Message Passing Interface (MPI) processes in parallel and embeds a CUDA process in each MPI process to compute each network on the GPU.

First, we need to check if the device supports multiple GPUs.

```
gpuDeviceCount
```

This tells us how many devices are presently connected.

Computing batch PANDA networks

To run several gpuPANDA processes in parallel.

```
nPandaNetworks=2; %for example
```

```
parpool(gpuDeviceCount);
parfor i = 1:nPandaNetworks
    %using the previous parameters
    tic;AgNet = panda_run(lib_path,exp_file, motif_file, ppi_file, panda_out,...
            save_temp, alpha, save_pairs, modeProcess,respWeight, absCoex,...
            similarityMetric, computing, precision, verbose, saveGPUmemory);toc;
end
```

Obviously, the resulting networks are exactly the same, but you can modify the input parameters such as the gene expression to obtain different types of networks. This is the case for LIONESS networks.

Computing LIONESS networks

LIONESS [2] estimates single-sample gene regulatory networks through computing the difference between the network of all the samples and the network deprived of the sample of interest. LIONESS is particularly interesting because it allows the generation of a population of gene regulatory networks which can be used to perform differential targeting analysis on the edges of the network across all the population.

The generation of sample specific networks for large-scale genomic studies like TCGA and GTEx requires consequent computation time and budget. gpuLIONESS allows to compute single-sample networks for a fraction of the cost and time.

We will use the parameters of the toy model to demonstrate gpuLioness

```
exp_file   = 'test_data/expression.transposed.mat';
motif_file = 'test_data/motif.normalized.mat';
ppi_file   = 'test_data/ppi.normalized.mat';
panda_file = 'panda2.test.mat';
load('test_data/panda.test.mat');
AgNet      = AgNet';
save('panda2.test.mat','AgNet');
alpha      = 0.1;
START      = 1;  % sample-of-interest starting from this index
END        = 10; % sample-of-interest ending to this index; use -1 to end at the last s
ascii_out  = 0;  % set to 1 if you prefer text output file instead of MAT-file
save_dir   = 'test_data';
lib_path   = '';
verbose    = 0;%supress the output
```

We set the parameter for GPU computing

```
computing  = 'gpu';
```

and we call LIONESS

```
lioness_run(exp_file, motif_file, ppi_file, panda_file, save_dir, START, END, alpha,...
    ascii_out, lib_path, computing, verbose);
```

As specified in the parameters, we computed 10 single-sample networks, that were distributed among the *n* GPU devices, such as each device processes 10/n networks. This way, gpuLIONESS takes advantage of the bilevel parallelism of modern GPU cards, where each network is assigned to a device, and each device computes the network using the parallel capabilites of the GPU.

References

[1] Glass, Kimberly, et al. "Passing messages between biological networks to refine predicted interactions." *PloS one* 8.5 (2013): e64832.

[2] Kuijjer, Marieke Lydia, et al. "Estimating sample-specific regulatory networks." *iScience* 14 (2019): 226-240.