RandomMatrixMultiplication Documentation

I have written a program whose main function is to do matrix multiplication by taking a number for the dimensions of a square matrix, a number of iterations, and values for B and L-values used in a squeeze function. The program constructs a square matrix of the given size and an activation vector that can be multiplied with the matrix. The matrix is filled with random integer between 0(included) and 10(excluded), and the activation vector is filled with random decimal values between 0(included) and 1(excluded). The activation vector and the matrix are multiplied, and the values of the resulting vector are bounded by the following squeeze function, which keep the values bounded between 0 and 1:

$$f(x) = 1 / (1 + \exp(-L * (x - B)))$$

Where x is a value in the vector resulting from the multiplication operation, and L and B are parameters supplied by the user. L = 4 and B = 0.5 would an example of typical values inputted by the user. After the squeeze function is applied to every value of the resulting vector, the new vector is stored into a matrix storing the results. This process is repeated for the number of iterations entered by the user, until you have a matrix with the resulting vectors from each iteration.

The program also has functions to plot the vectors in the resulting matrix, print the vectors, write to a file, and read from a file. To use this program, in the python interpreter, type:

```
>>> import RandomMM
```

From there instructions will be printed out detailing how to use the program. Here are the instructions:

Functions:

multiplication(int dim, int iterations, float L, float B): matrix of dimensions dim x dim will be multiplied by activation vector, producing another vector. This will be multiplied by the matrix again, and this process repeats for the number of iterations entered. L and B are values to be entered in the following squeeze function:
f(x) = 1 / (1 + exp(-L * (x - B))). This function will be applied to every element of the resulting vector to keep values bounded between 0 and 1. The result of this function
will be a 1-dimensional array containing the results from each iteration.

plotResults(): scatter plot of the points in each vector created in each iteration. Points from different vectors will have different colors. There are only 8 colors available,
so if there are more than 8 iterations, there will have to be repeating colors.

plotResults(array, length, numVectors): plot from given array of activation vectors, given the length of each activation vector and number of activation vectors in the array.

writeResultsToFile(fileName): writes array produced by the multiplication function to the given file.

readFromFile(fileName): reads the array stored in the given files.

printResults(): prints sequence of activation vectors. The vectors are printed out vertically.

printResults(array, length, numVectors): prints out supplied array of activation vectors, given length of each activation vector and the number of vectors in the array.

Note: functions that do not take an array as a parameter rely on the array of activation vectors calculated in the last use 'multiplication' function

----

Here are all the files contributing to this program:

❏ RandomMatrixMultiplication.cu: a cuda/c++ file that contains the code that does the heavy lifting of the program, which performs the matrix multiplication and squeeze operations and stores the results on the heap.
❏ RandomMatrixMultiplication.h: the file where the functions contained in RandomMatrixMultiplication.cu are defined. RandomMatrixMultiplication.cu contains the actually code, RandomMatrixMultiplication.h is simply where the functions are originally defined. This file is necessary because cython cannot interface with .cu files, only c++ files.
❏ libRandomMatrixMultiplication.so:RandomMatrixMultiplication.cu is compiled into this shared library. This step is also necessary so that cython can interface with the Cuda code.
❏ RandomMatrixMultiplication.pyx: this cython code that wraps the matricies generated by the cuda code into NumPy arrays that can be used by python. Cython is a programming language that allows for both Python and C code. Happily, this code does not copy the

underlying data allocated in the heap, just converts it to a NumPy array. This is where all the functions available to the user from the python interpreter are defined.

- ❑ SetupRandomMM.py: cython files require a python file to compile them. This file is to compile RandomMatrixMultiplication.pyx
- ❑ SetupRandomMM.py is compiled into a shared library called RandomMM.cpython-36m-x86_64-linux-gnu.so. When the user imports RandomMM in the python interpreter, this is the file that they are importing.

Compilation Instructions

1. Put the following files in the same directory: RandomMatrixMultiplication.cu, RandomMatrixMultiplication.h, SetupRandomMM.py, RandomMatrixMultiplication.pyx

2.If nvcc is not in your PATH variable, type the following:

```
    PATH="/usr/local/cuda-9.1/bin:$PATH"
```

3. Compile RandomMatrixMultiplication.cu into a shared library by typing the following:

```
     nvcc --compiler-options '-fPIC' -shared -o libRandomMatrixMultiplication.so
RandomMatrixMultiplication.cu
-I/home/psimen/anaconda3/lib/python3.6/site-packages/numpy/core/include/numpy
-I/home/psimen/anaconda3/include/python3.6m -lcublas -lcurand
```

You should now have file in your directory called libRandomMatrixMultiplication.so. Note the cublas and curand flags at the end.

4. Now you must modify the SetupRandomMM.py file. Open it and you should see the following:

```
Extension('RandomMM',
            sources=['RandomMM.pyx'],
            library_dirs=['/usr/local/cuda-9.1/lib64',
'/home/thomasnemeh/CudaPrograms/RandomMatrixMultiplication'],
            libraries=['cudart', RandomMatrixMultiplication],
            language='c++',
            runtime_library_dirs=['/usr/local/cuda-9.1/lib64',
'/home/thomasnemeh/CudaPrograms/RandomMatrixMultiplication'],
            include_dirs =
['/home/psimen/anaconda3/lib/python3.6/site-packages/Cython/Includes',
'/usr/local/cuda-9.1/include', '/home/psimen/anaconda3/lib/python3.6/site-packages']
```

There are places where "/home/thomasnemeh/CudaPrograms/RandomMatrixMultiplication" is listed. One is in library_dirs and the other is in runtime_library_dirs. Delete "/home/thomasnemeh/CudaPrograms/MatrixMultiplication" and replace it with the name of the directory you put the files in.

4. Now compile SetupRandomMM.py by typing the following:

```
/home/psimen/anaconda3/bin/python3.6 SetupRandomMM.py build_ext --inplace
```