EL2310 C Project Assignment

1 Objective

After reading through the expected solution, figure out the objective of the assignment.

2 Expected solution

2.1 Functional modules

1. Investigate the argument parsing in C. Write a main function that can parse integer inputs from command line. You should be able to parse an integer value when you run the program from commandline and store the value in a variable:
   e.g. if *./cproject 2*, then m=2

2. Write a function that can read a formatted (space separated values) text file line by line and fills a type double matrix of any shape. An example function prototype is:
   void fileReader(char fileName, double** matrix)*

3. Write a function that can generate random numbers within an interval [a,b] where $a < b$. An example of function prototype is:
   int randomNumberGenerator(int a, int b).

4. Write a function that can calculate the Euclidean distance between two vectors of size 1xN of type double. An example function prototype is:
   double calculateEuclideanDistance(double vector1, double* vector2, int N)*

2.2 Main Program

Using the functions that you have created previously, write a function that implements the following steps:

- Gets command line argument m as integer
- Opens and reads the contents of file *data.txt* (provided along with this assignment) to a multidimensional array of size r (rows) by c (cols) . You can determine the r and c of the array from the dimension of *data.txt*. Each row of the *data.txt* corresponds to one datapoint. Each column in a row correspond to a feature of that datapoint.

- Using the random number generator, pick m **different** datapoints from the data array. Let's call this set $P_m = \{p_2, p_2, ..., p_m\}$
- All the r datapoints together (which includes the m datapoints) is called as set $D_r = \{d_1, d_2, ..., d_r\}$.

2.2.1 Iteration

1. Calculate the Euclidean distance between each datapoint in D_r and each element of P_m. For each datapoint, save the index of p that is closest. For example, for datapoint d_2 if p_1 is closest then the index of datapoint d_2 should be 1.
2. Store the current value of p's to another array such as $p_{previous}$. For each p, calculate the new value by using the average of closest datapoints. For example, if you have found out that $\{d_2, d_5, d_{17}\}$ are closest to p_1 (i.e. if those datapoints have been indexed as 1), then p_1's new value is the average of this set (averaged along each feature).
3. Check if the new calculated value of p's are the same as $p_{previous}$. If they are same, then stop the iteration and return the calculated values of each p. Otherwise continue with Step 1.

Try the whole program with different values of m and find optimal m that takes the least iterations.

3 Submission instructions

List of files that your "zipped" file should contain:

- cproject.h: A header file that contains all defines and function declarations
- randomgen.c: A secondary source file that contains the definition of the "random number generator" function.
- program.c: A primary source file that contains the main program and the function definitions
- Makefile: A make file to compile and execute the program
- README.txt: A text file to include a brief report on how the program is implemented. Don't forget to include comments in your program at right places.
- DESCRIPTION.txt Describe what your program does, and why it is useful in the scientific community. Are something missing in the objectives? if so, then what are the ways it can be extended?

Important note: The data.txt provided is an example file. We will use another data file with your program while evaluating your solution.