CSCI 4372/5372/6397: Data Clustering

Phase 1: Reading a Data Set

Submission Deadline: **Wednesday, January 21 (23:59 CT)**

Hint #1: There is **no** clustering involved in this program. This is a warm-up phase.

Objective: Write a **command-line** program to read a data set.

Hint #2: A command-line program does **not** have a graphical user interface. However, you can write (or execute) a command-line program using a graphical IDE (examples include but are **not** limited to Visual Studio, NetBeans, and Eclipse.) In other words, you do **not** need a command-line interpreter/shell to write (or execute) a command-line program.

Input: Your program should be **non-interactive** (that is, the program should **not** interact with the user by asking them explicit questions) and take the following **command-line** arguments: <F> <K> <I> <T> <R>, where

- *F*: name of the data file
- *K*: number of clusters (**positive** integer greater than one)
- *I*: maximum number of iterations (**positive** integer)
- *T*: convergence threshold (**non-negative** real)
- *R*: number of runs (**positive** integer)

Warning #1: Do **not** hard-code any of these parameters (*e.g.*, using a statement such as "int R = 100;") into your program (you should also **not** use meaningless variable names such as *R*). The values of these parameters should be given by the user at run-time through the command prompt. See below for an example.

The first line of *F* contains the number of points (*N*) and the dimensionality of each point (*D*). Each of the subsequent lines contains one data point in blank separated format. In your program, you should represent the attributes of each point using a **double-precision floating-point** data type (*e.g.*, "double" data type in C/C++/Java). In other words, you should **not** use an integral data type (byte, short, char, int, long, etc.) to represent an attribute. You should also **not** use a string data type (char * in C/C++, std::string in C++, or String in Java) to store the data points in memory. However, you may use non-integral data types (*e.g.*, string's) for temporary storage (*e.g.*, to store a line you read from the input file).

Once (or as) you read the input file, you should select your *K* initial cluster centers. These centers should be selected **uniformly at random** from the data points.

Warning #2: Do **not** confuse *uniform random selection* and *uniform random generation*. The former involves selecting centers from the existing/real data points uniformly at random, that is, each data point has a (roughly) equal chance to be selected. The latter involves generating

artificial centers within the data space uniformly at random. You are supposed to select randomly, **not** generate randomly. In other words, the selected centers **must** be real data points.

Output: Display the initial cluster centers selected from the data set. Note that the output of your program may be different in each execution due to randomization.

Sample Input/Output:

% F = iris_bezdek.txt (name of data file)
% K = 3 (number of clusters)
% I = 100 (maximum number of iterations in a run)
% T = 0.000001 (convergence threshold)
% R = 100 (number of runs)
% test is the name of the executable file
% ">" indicates command-prompt, which is **not** part of the output

```
> test iris_bezdek.txt 3 100 0.000001 100

5.1 3.4 1.5 0.2
7.2 3.2 6 1.8
4.6 3.1 1.5 0.2
```

Note that the output above consists of **only** three lines of information because $K = 3$ and that each line of the output contains four values because iris_bezdek is a 4-dimensional data set.

Testing: Test your program on the 10 data sets given on Blackboard.

Hint #3: Develop your program incrementally. Write some code and then test it. If it works, add some more code. Do **not** try to write the entire program at once and expect it to work flawlessly.

Hint #4: You might want to dump/write the data set you read to a file or the terminal to ensure your file input routine works correctly. Remember that there are different ways to do that, depending on the programming language.

Programming Language: C, C++, or Java. You may **only** use the built-in facilities of these languages. In other words, you may **not** use third-party libraries.

Submission: Submit your source code (C, C++, or Java source files) and output files (**for each** input file, there **must** be an output file in .TXT format) through Blackboard. Do **not** submit your entire project folder or the input data files I provided (I already have them 😊). In addition, do **not** submit your files individually; pack them into a single archive (*e.g.*, zip) and submit the archive file.

Grading: *Functional correctness* and *adherence to good programming practices* are respectively worth **90%** and **10%** of your grade. Given a valid input, a functionally correct program is one that produces the correct output. There are a lot of generic or language-specific

guides on good programming practices on the WWW. You **must** identify an appropriate one, include its URL at the top of your program source code(s), and use it throughout this project. In general, you should pay more attention to structural issues (*e.g.*, modularity) than cosmetic ones (*e.g.*, naming and formatting). Remember that if your program is incorrect, whether or not you adhered to good programming practices is immaterial.

Contacting the Instructor for Help: Please consider the following issues before seeking help with your program:

- Please do **not** ask me about compile-time errors (this is a senior/graduate-level class).
- Please do **not** send me your entire program, asking me to find out what is wrong with it. You should at least have an idea about which part does not work.
- Please do **not** ask me to pre-grade your program before the submission deadline and then tell you how to improve your grade.

Hint #5: Use the Blackboard discussion forum to get clarifications. Remember **not** to post code fragments (or URLs pointing to code fragments) on the forum.

Hint #6: Before working on this phase, please read my email with the subject line "Welcome to 4372/5372/6397" again (it's also available in the Announcements section of Blackboard).