Homework 1 Report

Pedram Pejman

January 20, 2015

1 Abstract Data Types

For this homework assignment, I used multiple list data types. One to hold the unclassified items, and another to hold the classified items. I also created a structure (Java class: Item) to represent the abstraction of a "Data Item." Although we did not necessarily need a separate data type to represent Data Items, the encapsulation of different fields, like x, y-coordinates and distance, allows us to write cleaner code.

2 Data Structures

For this homework assignment, I used the ArrayList Data Structure in Java 7's Collections Framework. Two instances of ArrayLists were used to represent the unclassified and classified data items.

3 Bad Requirements

A possible clarification would be making it more clear when certain things should be done. For example, would it be ok to keep the user waiting while we load up the textfile into memory? At first, I implemented the program in that way (because it would be much faster and more efficient in terms of memory overall) but then I thought it might pose problems if the test cases are too large.

4 Program Design

First thing to understand is what the Item class represents. Of course it holds the fields corresponding to x, y-coordinate and the category of the item. But it also has a distance field which serves the purpose of comparing its distance to another Item. This Item class implements the Comparable Interface so that we can use the Collections Framework helper methods (such as sort()).

The problem is rather simple. All we do in this program is input k, M and the path for the data file. Then we ask for a list of unclassified items and store them in a list called

unclassified. We then read the textfile and create Item instances with the information on each line (the first M lines) and stores them in another list called *classified*. We then go through each item in the *unclassified* list and print out the following:

1. Print out the k nearest neighbors. We do this by iterating through all classified Items, updating each classified Item's distance (to its distance to the current unclassified item).

2. The best category for this element. This is determined by looking at the most frequent category in the nearest k elements.

3. The average distance for each category. Although we do not actually use this information for anything, we still show it as it is part of the requirements.

5 Error Checking

There are multiple things I had to change in my design in order to feel as though I have made a valiant attempt at solving the problem. First was, as mentioned before, processing the textfile after all the inputs have been given. this allows the program to run more smoothly for bigger data sets. Another interesting point was what happens when K is even and we have the same number of occurrences of members of each category. My algorithm handles this by choosing the second category it sees (this is to an extent unpredictable but we can be sure that we will select the category that appears in the nearest k neighbors first).
Another case of error checking that I conducted was taking a k bigger than M, in which case we are looking for more elements than we actually have. Other error checking cases that I did not do, but I should have, could be dealing with bad input, dealing with badly formatted data in the text files and gigantic text files (more than a couple Gigabytes).

6 Test Cases

1. First test case I used was a fairly simple one (very much like the example provided in the homework description). This test case did not aim to do anything specific other than to allow me to debug my code throughout the process of writing it.

2. The second test case I wrote was aimed to allow me to make sure the first requirement worked perfectly. I did this by first drawing out the positions of the items myself and calculating the distance of every classified item to the unclassified items.

3. The third test case I wrote allowed me to make sure the second and third requirements were satisfied by a fairly complicated data set. This was done by having about 20 classified items, testing 5 unclassified items. Again, I did the math myself and then checked it with the output given by my program.

7 Time Spent

I spent about 3 hours writing the code and test cases and about an hour and half writing this report. Allowing for a bit of time spent not being 100% productive (staring into space, trying to understand why I'm not smarter, etc.) the assignment took me about 6 hours.