Arrays and files in Visual Basic I

Topics

- Arrays and parallel arrays
- Reading from CSV files into parallel arrays
- More practice with formatting strings

TL;DR

Implement a program to predict the rating that the user will assign to a movie based on their ratings for other movies and the ratings of other viewers for the same movies.

Background

The objective for the lab is to make an accurate prediction of the rating that the user will assign to a particular movie based on their ratings for other movies and the ratings of other viewers for the same movies. The basic idea is to take the user's ratings for a set of movies and find other viewers with similar ratings. These similar viewers can then be used to inform our prediction.

This lab exercise uses a dataset that is maintained by the GroupLens research group at the University of Minnesota. The dataset is part of the MovieLens project. In their own words, MovieLens is a platform for "*Non-commercial, personalized movie recommendations*"; think Netflix recommendations, but not for profit. The dataset that we will be exploring consists of movie ratings for five popular movies from 22419 unique viewers. In the file `ML-ratings.csv` each row constitutes the ratings of a single viewer for each of the five movies listed in the file `ML-titles.txt`. The first three rows in the file look like:

```
4.0,4.5,4.0,4.0,3.0
3.0,4.0,5.0,4.0,5.0
5.0,5.0,5.0,5.0,5.0
```

This should be understood to mean that viewer 1 rated movie 1, 4.5 stars out of 5 possible; movie 2, 4.0 stars; movie 3, 4.0 stars and so on. Likewise for viewer 2 and 3.

Now, suppose that we added another row to the three rows above, for *the user*, who rated movies 1–4: 3.5, 4.5, 4.0, and 3.5, respectively. Then the task at hand is to make a prediction for that user's rating for movie 5 (`???` in the example below).

```
4.0,4.5,4.0,4.0,3.0
3.0,4.0,5.0,4.0,5.0
5.0,5.0,5.0,5.0,5.0
3.5,4.5,4.0,3.5,???
```

A simple but unreliable way to do that would be to generate a random number between 0.0 and 5.0. Hopefully you see the potential limitations of this approach. If not, discuss it with someone around you.

Since we have at our disposal a dataset that contains the ratings of 22419 other users (*viewers*), isn't it likely that some of them will have movie preferences similar to the user's? If so, couldn't we somehow use their ratings for movie 5 to create a better than random prediction for the user's rating of movie 5? The answer is *hopefully*; the accuracy of a prediction system like Netflix, and the one that your are going to build, depends on it.

In order to identify viewers similar to the user, we must have some quantitative way to determine similarity, i.e., a similarity measure. A common way to do this is to calculate the absolute difference for each of the movie ratings between the user and a viewer and add these together to compute a total difference measurement. This difference measure is known as the Manhattan Distance (a.k.a. Taxicab distance). It is important to note that in this case, *distance* is the opposite of *similarity*

(what we are really after). So, to find the most similar viewers, we should find the viewers that have the least distance to the user.

Since we will ultimately be trying to predict the rating for movie 5, we will not include that rating in our distance calculation. Thus, if we take the user with ratings 3.5, 4.5, 4.0, and 3.5, then the distance between the user and viewer 1 is calculated as:

```
  |3.5-4.0| + |4.5-4.5| + |4.0-4.0| + |3.5-4.0|
=  0.5      +  0.0      +  0.0      +  0.5
=  1.0
```

So we would say that the Manhattan distance between the user and viewer 1 is 1.0. Likewise between the user and viewer 1:

```
  |3.5-3.0| + |4.5-4.0| + |4.0-5.0| + |3.5-4.0|
=  0.5      +  0.5      +  1.0      +  0.5
=  2.5
```

Of the two viewers, the user is least different (read: most similar) to viewer 1, since the distance calculation was 1.0 for viewer 1, compared with 2.5 for viewer 2.

If we repeated this for all of the viewers in the dataset, we could determine which viewer has movie preferences most similar to the user, at which point we could use that viewer's rating of movie 5 to predict the user's rating of movie 5. However, as you will see, using a single viewer to inform predictions can also be unreliable. So in a final attempt at a more accurate prediction, most systems will aggregate the ratings of many similar users to inform the prediction. How many and how to combine their ratings will be discussed more in later parts of this lab.

Instructions

1. Launch the RAPTOR software and use it to open the program called `RatingPredictions_Part1.rap` inside of your lab folder.

   As written, the program creates two parallel arrays, called `viewerRatings1` and `viewerRatings2`, that hold the ratings for movie 1 and 2, respectively, for the three viewers shown in Background. The user is asked to input their ratings for movies 1 and 2, which are stored into variables `userRating1` and `userRating2`. At this point the program is meant to compute and display the distance between the user and each of the viewers. Currently, the program does not compute the distances; you are to complete the program to do this.

   Before attempting to finish the program, fill out the *expected output* column in Test Table 1 in `TestTables.docx`. Then, after you get your program working, fill out the *actual output* column, making sure that the two columns are consistent.

Checkpoint 1 (30/100)

- ☐ program produces correct output for all test cases in Test Table 1

2. Launch the VS Express 2013 software and open the project called `RatingPredictions_Part2` inside of your lab folder.

   Complete the action for `btnComputeDistances` so that it is equivalent to your RAPTOR program. You MUST use a `for`-loop to do this. Test the project by checking that it is producing correct results for all of the test cases in Test Table 1.

   Then, update the button action so that it stores the ratings for all five of the movies in parallel arrays called: `movie1`, `movie2`, `movie3`, `movie4`, and `movie5`. Use the values from the example above. Remember to update your distance calculation to include ratings for movies 1–4 only.

   Before attempting to finish the program, fill out the *expected output* column in Test Table 2 in `TestTables.docx`. Then, after you get your program working, fill out the *actual output* column, making sure that the two columns are consistent.

Checkpoint 2 (65/100)

- ☐ project is based on a complete and correct `RatingPredictions_Part1.rap` program
- ☐ project produces correct output for all test cases in Test Table 2

3. Copy and paste the folder `RatingPredictions_Part2`. Rename the copy `RatingPredictions_Part3`. Launch the VS Express 2013 software and open the project `RatingPredictions_Part3`.

   Complete the action for `btnLoadData` so that instead of the rating values being assigned in `btnComputeDistances`, they are instead read from the comma-separated value file, `ML-ratings-small.csv`. This file is provided for you in the project folder `RatingPredictions_Part3`, in a sub-folder called `Bin/Debug`. *Note: this will require changes in both buttons.*

   Test the project by checking that it is still producing correct results for all of the test cases in Test Table 2.

Checkpoint 3 (70/100)

- ☐ project populates ratings with values from the file `ML-ratings-small.csv`
- ☐ project produces correct output for all test cases in Test Table 2

4. Copy and paste the folder `RatingPredictions_Part3`. Rename the copy `RatingPredictions_Part4`. Launch the VS Express 2013 software and open the project `RatingPredictions_Part4`.

   Update the action for `btnComputeDistances` so that instead of the output statement from the previous parts, it outputs a table with the following format (the values in the following example are assuming the user input the values specified in Background):

```
Viewer ID   Distance
--------------------
        1        3.5
        2        3.0
        3        7.0
      ...        ...
--------------------
```

   You can declare a format string for such a table as follows:

```
Dim fmtStr As String = "{0,9}   {1,8:0.0}" & vbNewLine
```

   Which could then be used to format two variables like so:

```
String.Format(fmtStr, i + 1, distances(i))
```

   Thus, in `fmtStr`, we say that there are two *format items*, one for each of the two columns in the table. Each format item is demarked by `{...}`. Within the curly braces, the first number indicates the *value index*, i.e., which value in the line `String.Format(...)` is to formatted. The first value after the format string, `fmtStr` in this case, is considered value 0. So, `{0,7}` is the format item for the value `i`, `{1,7:0.0}` for value `distances(i)`, and so on if there were more format items. After the value index is the *alignment*, i.e., how wide the formatted column should be, as well as whether it should be left or right justified. So, `{0,7}` indicates that value `i` should be formatted as seven characters wide and right justified. You can see that in this format string, all format items are seven characters wide and right justified. Lastly, following the alignment is the *format specifier*, which indicates out a value should look when formatted. There are many possibilities for a format specifier, so we will not go into the details. Suffice it to say that the `0.0` format specifier in `{1,7:0.0}` means that value `distances(i)` should be formatted as a decimal value with a single digit following the decimal. If you would like more information on any of these topics, see this documentation from Microsoft.

   Lastly, update the action for `btnLoadData` so that it reads the rating values from the file `ML-ratings-medium.csv` instead of `ML-ratings-small.csv`. This is a set of 100 viewer ratings for the five movies described in `ML-titles.txt`.

Checkpoint 4 (75/100)

- ☐ project populates ratings with values from the file `ML-ratings-medium.csv`
- ☐ project produces correctly formatted output according to the example above

Submission Instructions

During our next lab meeting, you will be asked to expand your solutions to Part 4. Prior to the meeting, you are expected to

- finish ALL parts of this lab,
- study (again) any material you struggled with in this lab, and
- study new material needed for the next lab

You will submit your work for this lab and the next one together by midnight Thursday.