CS 460G: Homework #2 Report

By Steven Penava

Usage Instructions

1. Run knearest.py with u1-base.base and u1-test.test being in the same directory.
2. The program presents the user with several options, they are as follows:
    a. Make a singular prediction
        i. This allows the user to manually enter a single user and movie for the program to make a prediction which is printed to output.
    b. Calculate mean squared error for the test set with a k-value of 3
        i. This takes about 20 minutes on my 2013 MacBook Pro. It runs through the test set (size 20,000) and compares each rating with the prediction of those same values (user, movie, rating).
    c. Cross validate using 5 k-values with the leave-one-out method
        i. K-values: 3, 4, 7, 8, 10.
        ii. This took about 5.5 hours on my machine.
    d. Print cross validation and test error results
        i. I included the results in the "Error Results" section.

Implementation Details

I decided to implement K-Nearest Neighbor with a Euclidian distance similarity calculation. I started the program by importing the training data and storing the values in a dictionary of dictionaries. It is laid out like this:

```
#   { user 1:
#       { movie number 1 : rating 1 }
#       { movie number 2 : rating 2 }
#       {           ....          }
#     user 2:
#       {           ....          }
#     ...
#   }
```

Additionally, at the beginning of the program I created a dictionary of individual movies to users (getRelUsers function) to automatically have a narrowed subset of users when finding a list of similarities from one user to the others. In my predict function, I get the top K most similar neighbors and then find the weighted average of them with this formula:

$$Total += Similarity * Rating\ (run\ in\ loop\ of\ top\ K)$$

$$Prediction = \frac{Total}{sum\ of\ the\ top\ K\ similarities}$$

I completed cross validation using the leave-one-out method. I checked each time I calculated similarity that it did not calculate similarity with itself (hence leaving one out each run through). I tested the k-values of 3, 4, 7, 8, 10.

Cross Validation Algorithm Description

Import training data file

Loop through an array of k-values (size 5), and for each k-value/iteration…

- Line count = 0
- Mean squared error = 0.0
- Loop through training data file, and for each iteration…
  - Split the line to extract data
  - Append 1 to line count
  - Mean squared error += (actual rating – prediction of data)^2
- Seek back to top of file
- Individual mean squared error = 1/line count * mean squared error

Error Results

Mean squared error of the test set: 1.464

Leave-one-out mean squared error with K =3: 1.305

Leave-one-out mean squared error with K =4: 1.224

Leave-one-out mean squared error with K =7: 1.118

Leave-one-out mean squared error with K =8: 1.104

Leave-one-out mean squared error with K =10: 1.081

Overall average mean squared error with leave-one-out: 1.166

I noticed that, as the k-value went up, the error decreased by a considerable margin. This is due to the fact that there were more neighbors to compare to, increasing the chance of having a higher similarity and making the results less sparse.