Danny North

4/4/16

CS 401R

Dr. Wingate

Recommender Lab

I. The Dataset

The data that I used for my predictions came strictly from the user_ratedmovies_train dataset. This dataset was useful because it provided around 770000+ ratings to test. The data that I found useful for my algorithm was the userID, movieID, and rating columns. While there was other data that could have been considered (such as actors, directors, titles, critic ratings, etc.) from other files, this did not fit with the algorithm that I decided to use.

One of the reasons why I decided to stick with a simple dataset is because of the layout of the prediction.dat file. It only gives us userID and movieID as pertinent data, and the algorithm I chose compares every point with this data. Using this data and ratings keeps the approach and algorithm simple but it seems to work effectively without overhead.

II. The Algorithm

I decided to go with a k-means, k-nearest neighbors styled algorithm. The math for this algorithm is as follows:

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} |x_n - \mu_j|^2,$$

The reason why I went with this approach was because it made sense to me that someone who rated movies (ratingUser) similarly to the test user would also rate the test movie similarly. So the algorithm first finds all users that rated the test movie. Then it compares the rating habits of each ratingUser to the test user using a Euclidian distance algorithm.

After creating an array of all users who watched that movie in comparison to the test user, by comparing the intersection of movies that both users watched, the k-nearest-neighbors are chosen and the most common rating of those k-nearest-neighbors is chosen as the guess.

I considered making the test give the mean of these k-nearest neighbors, and that might have given a slightly better RMSE, but because we were working with increments of 0.5, I decided on the algorithm choosing the most common. The test and training set was partitioned randomly with the test set getting 85000 points and the training set getting the rest. This was the default validation set training that was given to us by Dr. Wingate, and there is no problem with this except for regular validation set problems. Once I got confident with this set of tests, I lowered the test size in order to accommodate a better guess because there was more training data.

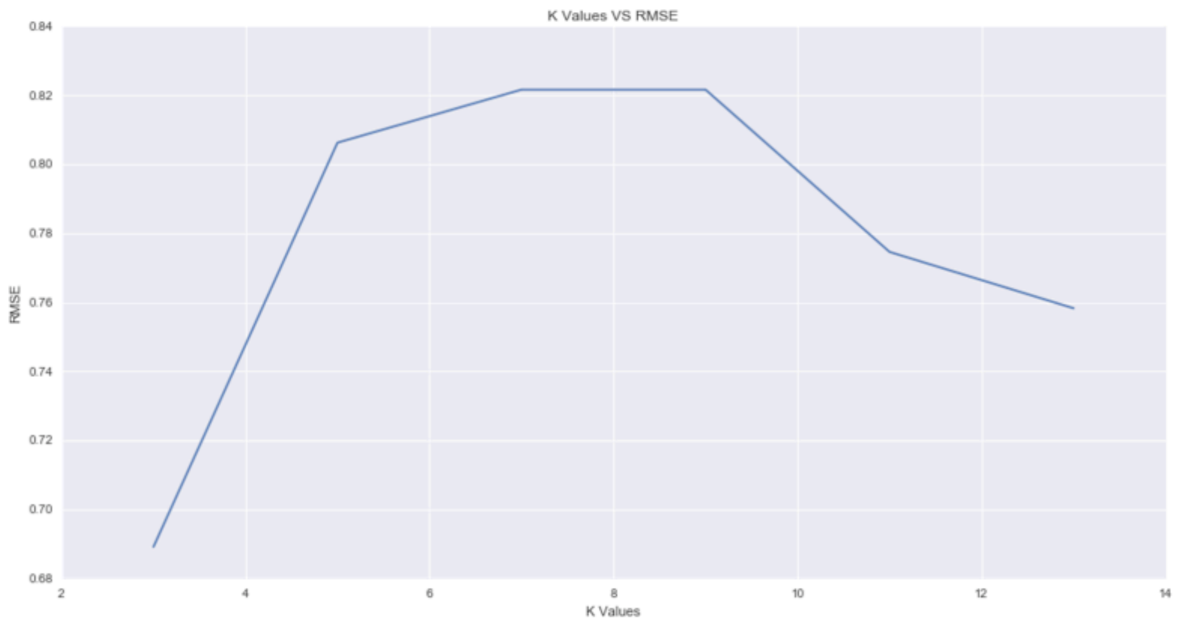
III. Training Analysis

The following table shows some example guesses for the test set, and it looks like it does fairly well, at least getting in the right area of the actual rating.

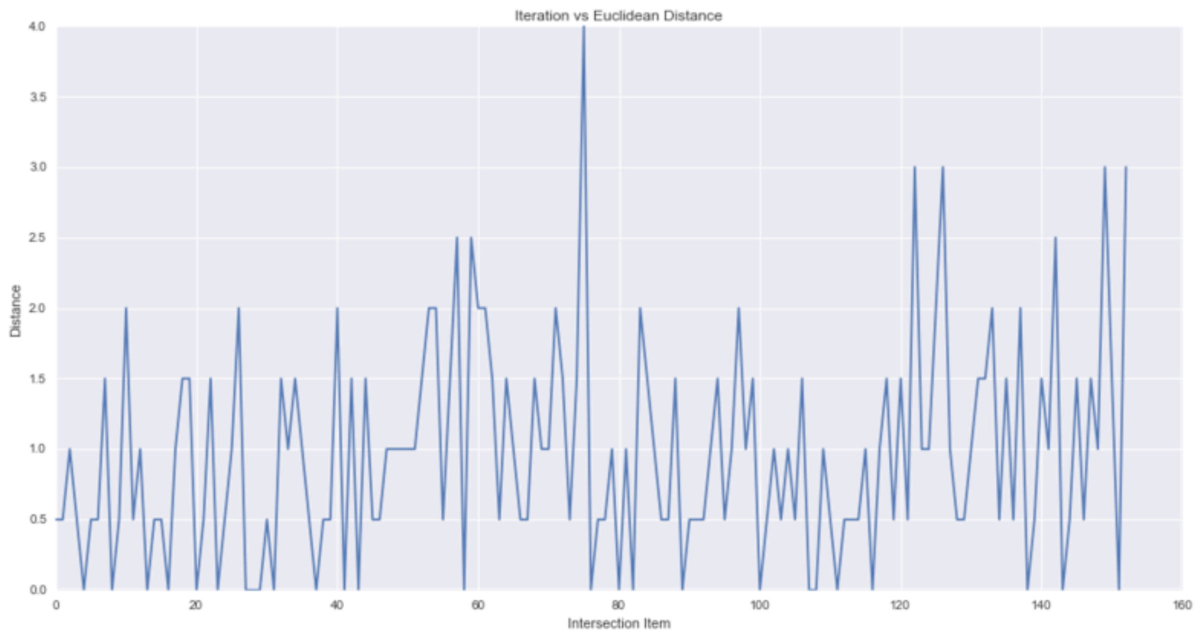
testID	Prediction	Actual
0	4.5	4
1	3.5	3.5
2	3.5	3.5
3	3.5	3.5
4	4	4
5	4	5
6	4	3.5
7	3.5	4
8	3.5	3

The RSME for my test set of 100 data points for testing and the rest for training came out to be 0.85. This was with a k-value of 5. I don't believe I overfit, if anything the algorithm was most likely an underfit of the data. This is because the algorithm used a very simple model and minimum data, so it wasn't taking advantage of all the possible features it could.

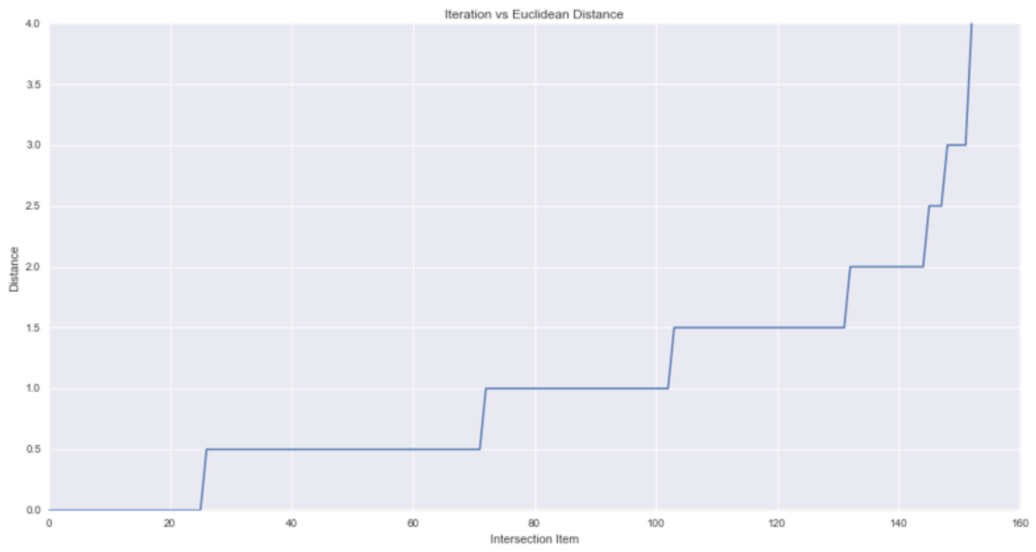
One test I did to analyze the training process was to get different RMSEs on a small dataset for different k-values. This was done so that I could understand which k-value better fit my dataset. The plot for this test is shown below:



I also thought it might be interesting to graph the individual Euclidian distances for a single iteration to see if it gives any good visual information. Below you can see the results of the graph which is sporadic. This is fine, because the compared users are in a random order:



It is interesting to note, however, what this same graph looks like when it is sorted. You can see the nearest neighbors are those below and it makes a step-like function that shows the distance to its neighbors that are the furthest away:


Iteration vs Euclidean Distance
Distance
Intersection Item