CSEP 546: Data Mining (Spring 2017)
Assignment 2: Collaborative Filtering and Bayesian Networks

-Nithin Mahesh

Problem 1: Collaborative Filtering on Netflix Ratings

Code:

The code for the assignment is submitted as a zip file. It is implemented in Java through IntelliJ IDEA and can be compiled and run through the same. It is also available in the following github repository: https://github.com/nithinmahesh/DataMiningAssignments/tree/master/Assignment2

Here are the key functions in the code:

1. **Load Training Data**
   - This section has all functions related to loading the training data.
   - *loadTrainingData(String csvFile)*
     - This takes a file as input and loads all data from the input file and uses addDataPoint to add to the in-mem data set.
   - *addDataPoint(Integer userId, Integer movieId, Double rating)*
     - Adds a data point to the various in-mem objects.
2. **Post Processing on Training Data**
   - This section has all functions concerned with post processing of training data.
   - *doPostProcessing()*
     - This is the entry point for post processing of training data. This co-ordinates the various post processing functions like calculating mean and weights.
   - *calcDistance(Integer user1, Integer user2, Double u1mean, Double u2mean)*
     - This calculates the weight between a given user pair
   - *addWeight(int user1, int user2, double weight)*
     - This is a helper function which adds the weight between a user pair to the in-mem objects.
   - *loadWeights(String csvFile)*
     - The program allows reading pre-computed weights from a file and this function loads the weights from a file.

3. **Evaluate Test Data**
    - *evaluateOnTestData(String filename, Double threshold)*
        - This is the entry point for evaluating the test data to make predictions.
    - *addTestDataPoint(int userId, int movieId, double actualRating, Double threshold)*
        - This function calls predictVote to predict the vote for a single test data and then manages all the error computations for the same.
    - *predictVote(Integer userId, Integer movieId, Double threshold)*
        - This is the actual function which predicts the vote for a given test data.
    - *getDistance(Integer user1, Integer user2)*
        - This is a helper function to get the weight between a given user pair.
4. **Result Helper Function**
    - *printStats(Double threshold)*
        - This helper function is used to print the Root Mean Squared Error and Mean Absolute Error.

Accuracy:

The best accuracy that has been obtained by considering all users is the following:

Mean Absolute Error: 0.69492

Root Mean Squared Error: 0.88446

These were obtained by using all users who had watched the same movie as was currently being predicted.

Initially, the accuracies that I had obtained were RMSE > 2 and this was because when computing the k as a reciprocal of sum of absolute weights, I had missed to do the absolute value of each of weights thereby resulting in a higher k value than expected.

One of the other things that I tried was to include only certain weights in prediction whose absolute value is above a given threshold. This way, users who aren't even close enough to the user being predicted for are avoided from affecting the prediction of this user. This technique gave varied results by varying the threshold. Following are the results below (rounded off to 4 digits after decimal):

Threshold	**RMSE**	**MAE**
0	0.8845	0.6949
0.01	0.8844	0.6949
0.05	0.8839	0.6944
0.1	0.8825	0.6930
0.2	0.8783	0.6889
0.25	**0.8761**	**0.6868**
0.5	0.8882	0.6926

As we can see, the best RMSE score is for threshold 0.25 which gives an RMSE score of 0.8761. Going higher to a threshold of 0.5, the errors start to increase again.

Best RMSE Score: 0.8761

Best MAE Score: 0.6868

Shortcomings of Collaborative Filtering:

1. The runtime of the algorithm is linearly dependent on the number of users and the number of movies. This does not scale well for very large datasets which is the main practical use of recommender systems.
2. The algorithm cannot immediately recommend movies for a new user until he watches and rates some movies. Same way when a new movie is added it cannot recommend that to anyone until some users have rated it.
3. It does not allow grouping of movies to be able to have associated relation between objects.