Data Science

Assignment 4 Recommender System

Prof. Sangwook Kim  
TAs. Jangwan Koo, Yeonchang Lee

2013011424  
Yedarm Seong  
Division of Computer Science and Engineering

1. Environment

OS: Windows 10  
Language: Python 3.5.2

2. Summary of algorithm

Using Method: Collaborative filtering with User based

Similarity measure: Pearson correlation coefficient

Rating Prediction: Weighted Sum / Normalize value

Weighted Sum =

Normalize value =

Recommend(user u, item i, k) # Rating when recommending I to u, k is number of similar users  
similar users <- GetSimilarUser(u)  
loop s in similar users:  
 weight <- PearsonCorrelation(u, similar user)  
 normalize value += weight  
 weighted sum += weight \* Rating(s, i), - AverageRating(s)  
until No more similar users left or When there are k similar users  
  
if no have similar user, then prediction <- 0  
else, then prediction <- (AverageRating(u) + (weighted sum / normalize value))  
if prediction is greater than 5, then prediction <- 5  
if prediction is less than 1, then prediction <- 1  
return prediction

3. Detailed description of codes (for each function)

Make user rating table

**def user\_ratings\_dict**(data\_set, max\_user\_num, max\_item\_num):  
 user\_ratings = [**None**] \* (max\_user\_num + 1)  
 bool\_matrix = np.zeros((max\_user\_num + 1, max\_item\_num + 1), dtype=bool)  
  
 **for** i **in** range(len(data\_set)):  
 bool\_matrix[data\_set[i][0]][data\_set[i][1]] = **True** d = {data\_set[i][1]: data\_set[i][2]}  
 **if** user\_ratings[data\_set[i][0]] == **None**:  
 user\_ratings[data\_set[i][0]] = d  
 **else**:  
 user\_ratings[data\_set[i][0]].update(d)  
 **return** user\_ratings, bool\_matrix

The user-rating table is created by taking the data set, the maximum user number, and the maximum movie number as arguments.

Make user average rating table

**def** user\_average\_ratings(data\_set, max\_user\_num):  
 user\_average\_ratings = np.zeros(max\_user\_num + 1)  
 user\_rating\_counts = np.zeros(max\_user\_num + 1)  
  
 **for** i **in** range(len(data\_set)):  
 user\_average\_ratings[data\_set[i][0]] += data\_set[i][2]  
 user\_rating\_counts[data\_set[i][0]] += 1  
 user\_average\_ratings /= user\_rating\_counts  
 **return** user\_average\_ratings

It receives the data set and the maximum number of users and creates an average rating table for all users.

Get common item between user1 and user2

**def** common\_items(user1, user2, user\_ratings):  
 commons = []  
 d1 = user\_ratings[user1]  
 d2 = user\_ratings[user2]  
  
 **for** item **in** d1.keys():  
 **if** item **in** d2.keys():  
 commons.append(item)  
 **return** commons

If there is a common item between two users, return it to the list.

User-user Pearson correlation coefficient table.

**def user\_pearson\_matrix**(user\_ratings, max\_user\_num, user\_avg\_ratings):  
 matrix = np.zeros((max\_user\_num + 1, max\_user\_num + 1))  
  
 **for** user1 **in** range(1, max\_user\_num + 1):  
 **for** user2 **in** range(1, user1):  
 similarity = 0  
 p = 0  
 s1 = 0  
 s2 = 0  
  
 user1\_avg\_rating = user\_avg\_ratings[user1]  
 user2\_avg\_rating = user\_avg\_ratings[user2]  
  
 commons = common\_items(user1, user2, user\_ratings)  
  
 **if** 0 != len(commons):  
 **for** item **in** commons:  
 p += (user\_ratings[user1][item] - user1\_avg\_rating) \  
 \* (user\_ratings[user2][item] - user2\_avg\_rating)  
 s1 += np.power(user\_ratings[user1][item] - user1\_avg\_rating,  
 2)  
 s2 += np.power(user\_ratings[user2][item] - user2\_avg\_rating,  
 2)  
  
 similarity = p  
 similarity /= np.sqrt(s1 \* s2)  
  
 **if** np.isnan(similarity) **or** np.isinf(similarity):  
 matrix[user1][user2] = 0  
 matrix[user2][user1] = 0  
 **else**:  
 matrix[user1][user2] = similarity  
 matrix[user2][user1] = similarity  
 **return** matrix

And calculates the Pearson correlation coefficient between the two users. If there is no common item, it is set to 0.

Get similar users (users whose Pearson correlation coefficient is greater than 0)

**def get\_similar\_users**(pearson\_matrix, user):  
 similars = []  
  
 **for** i **in** range(1, len(pearson\_matrix)):  
 **if** pearson\_matrix[i][user] > 0:  
 similars.append(i)  
 similars.sort()  
 **return** similars

Check the Pearson matrix and sort the users whose Pearson correlation coefficient is over 0 into user numbers and return them as a list.

When recommending an item to a user, the user predicts the rating

**def recommend**(user\_ratings, user\_avg\_ratings, pearson\_matrix, bool\_matrix,  
 user, item, k):  
 count = 0  
 prediction = 0  
 normalize\_value = 0  
 K = k  
 weigthed\_scaled\_sum = 0  
  
 similar\_users = get\_similar\_users(pearson\_matrix, user)  
  
 **for** sim\_user **in** similar\_users:  
 **if** bool\_matrix[sim\_user][item] == **True**:  
 count += 1  
  
 weight = pearson\_matrix[user][sim\_user]  
 normalize\_value += weight  
  
 # 가중치와 정규화 상수를 이용한다.  
 weigthed\_scaled\_sum += \  
 weight \* (  
 user\_ratings[sim\_user][item] - user\_avg\_ratings[sim\_user])  
  
 K -= 1  
 **if** K == 0:  
 **break** # 비슷한 유저가 없다면 예상 점수는 0으로 처리한다.  
 **if** 0 == count **or** 0 == normalize\_value:  
 prediction = 0  
 **else**:  
 prediction = user\_avg\_ratings[user]  
 prediction += (weigthed\_scaled\_sum / normalize\_value)  
  
 **if** prediction > 5:  
 prediction = 5  
 **elif** prediction < 1:  
 prediction = 1  
 **return** round(prediction)

The weighted sum and the normalized variable are used to calculate the predicted score. The maximum rating is 5 and the minimum rating is 1, so the above and below are treated as the maximum and minimum values.

4. Instructions for compiling source codes at TA's computer (e.g. screenshot) (*Important!!*)

**Python does not need a compile process**

If you already install pyinstaller for python3, in the directory where the recommender.py file is located, type the following command:

Windows:  
pyinstaller -F recommender.py

Ubuntu:  
pyinstaller -F recommender.py

Then the dist folder is created in the directory where the command is executed, and there is ‘recommender.exe’ file in it.(For linux, the ‘recommender’ file)

Usage: recommender.exe [base file] [test file]

After using pyinstaller, there is an exe file in the dist folder:  
ex) dist/recommender.exe [base file] [test file]

If you move the exe file to the same location as input file, you can use it like this:  
ex) recommender.exe [base file] [test file]

5. Any other specification of implementation and testing

Another execution method:  
Running as a Python file

ex) recommender.py [base file] [test file]