Data Science – Report #4

I. Environment

A. OS: Mac

B. Python 3.8.2

II. Example of Compiling

A. `$ python recommender.py u1.base u1.test`

B. Time to program runs takes for about 15~20 minutes.

III. Summary of Algorithm

A. I used "collaborative filtering" to implement a recommender system. I implement rating matrix and similarity matrix computed by Pearson Correlation Coefficient by using pandas module. I get top 30 neighbors if a user has neighbors and estimate the rating of the given item.

B. Aggregation method that I used for rating

$$r_{c,s} = \bar{r}_c + k \sum_{c' \in \hat{C}} sim(c, c') \times (r_{c',s} - \bar{r}_{c'})$$

C. Reference: Recommender System PPT lecture

IV. Detailed Description of Code

A. getData

```
def getData():
    matrix = np.loadtxt(sys.argv[1], dtype='int', usecols=(0,1,2)) # Load data
    matrix = pd.DataFrame({'user':matrix[:,0],
                           'item':matrix[:,1],
                           'rating':matrix[:,2]})
    # Make it as a rating matrix form
    rating_matrix = matrix.pivot_table('rating', index='user',columns='item', fill_value=0)
    return rating_matrix
```

i. This function is to load the given database and changes to rating matrix form by using pandas module.

B. get_simlilarity

```
# Get simliarty by using PCC
def get_similarity(rating_matrix):
    sim_matrix = (rating_matrix.T).corr(method='pearson')
    return sim_matrix
```

i. This function is to calculate the similarity matrix by using pearson correlation coefficient.

C. Estimate

```
# To estimate the rate
def estimate(user, item, rating_matrix, sim_matrix, avg, neighbor):
    # If the item is not rated by all users, return the minimum rate
    if item not in list(rating_matrix.columns):
        return 1.0

    norm = cnt = new_rate = res = 0
    # To add the rate of each user
    for idx, sim in neighbor:
        idx = int(idx)
        if rating_matrix.loc[idx][item] == 0:
            continue
        if cnt == 30:
            break
        cnt += 1
        norm += sim
        rate = rating_matrix.loc[idx][item]
        new_rate += sim*(rate-avg[idx])

    if norm == 0:
        res = avg[user]
    else:
        res = avg[user] + (new_rate/norm)
    if res > 5:
        res = 5.0
    elif res < 1:
        res = 1.0

    return res
```

i. This function is to get the rating of the given item. I calculate aggregation of ratings by top 30 neighbors

D. predict

```
# To predict the rate and write the result
def predict(rating_matrix, sim_matrix):
    output = sys.argv[1] + '_prediction.txt' #Output file
    avg, neighbors = get_avg_neighbor(rating_matrix) #Get neighbor list and average list

    with open(sys.argv[2], 'r') as tf, open(output, 'w') as f:
        while True:
            line = tf.readline().strip()
            if not line or len(line) == 0:
                break
            line = line.split('\t')
            prediction = estimate(int(line[0]), int(line[1]), rating_matrix, sim_matrix, avg, neighbors[int(line[0])])
            result = line[0] + '\t' + line[1] + '\t' + str(prediction) + '\n'
            f.write(result)
```

i. This function is to save the result predicted by estimate function.

E. get_avg_neighbor

```
# Get average rate and neighbors of each user
def get_avg_neighbor(rating_matrix):
    average = {}
    neighbors = {}
    for uid in list(rating_matrix.index):
        # Get average rate of each user
        rating = list(rating_matrix.loc[uid])
        cnt = len(list(filter(lambda x: x!=0, rating)))
        if cnt != 0:
            average[uid] = sum(rating)/cnt
        else:
            average[uid] = 0
        # Get neighbor
        neighbor = sim_matrix[uid].sort_values(ascending=False)
        neighbor = neighbor.reset_index().values.tolist()
        neighbors[uid] = neighbor

    return average, neighbors
```

i. This function is to get average ratings of each user and neighbors of each user in the descending order.

F. Main function

```
if __name__ == '__main__':
    rating_matrix = getData() #processing data into rating matrix
    sim_matrix = get_similarity(rating_matrix) #get similarity matrix
    predict(rating_matrix, sim_matrix) #To predict the rate of given users
```

V. Test Result

```
C:₩Users₩hsy16₩Desktop₩test>PA4.exe u1
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 0
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 0.9578037

C:₩Users₩hsy16₩Desktop₩test>PA4.exe u2
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 0
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 0.9467625

C:₩Users₩hsy16₩Desktop₩test>PA4.exe u3
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 0
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 0.9413204
```

```
C:₩Users₩hsy16₩Desktop₩test>PA4.exe u4
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 0
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 0.937904

C:₩Users₩hsy16₩Desktop₩test>PA4.exe u5
the number of ratings that didn't be predicted: 0
the number of ratings that were unproperly predicted [ex. >=10, <0, NaN, or format errors]: 0
If the counted number is large, please check your codes again.

The bigger value means that the ratings are predicted more incorrectly
RMSE: 0.939675
```