[Assignment report #4]

Recommendation Algorithm with Movielens Data

2012023871
경영학과
강민철

1) Summary of Algorithm

To make recommendation on movielens data, I used collaborative filtering algorithm. As the name implies, collaborative filtering(CF) makes use of the wisdom of the crowd to recommend items. The basic assumption of CF is that the preferences of people remains stable and consistent over time. When I start building the recommendation algorithm, I made both user-based and item-based collaborative filtering. However, the performance of user-based CF outperformed that of item-based approach even though the computing time of item-based CF was worse than user-based approach. Therefore, I only introduce user-based approach in this report. User-based approach (a.k.a user-based nearest-neighbor collaborative filtering) finds set of other users whose ratings are similar to the target person's ratings. Based on that set, it estimates user's unknown ratings.

2) Detailed Description of Codes

[Library Import]

```
from math import sqrt
import sys
```

- 'sys' enables a .py file to take arguments on the bash level.
- 'math' are used to calculate square root and power for Euclidean distance.

[Main function]

```
if __name__ == "__main__":

    input = sys.argv[1]
    compare = sys.argv[2]

    train = Readtofavors(input)
    test = FileReader(compare)



    # User-based approach
    u_Res = make_user_prediction(train, test)
    file_name = input + "_prediction.txt"
    out_file = open(file_name, 'w')
    Result = write_file(u_Res, out_file)
    out_file.close()
```

First two lines are for setting the sys.argv for taking arguments from user. Then it reads train file with Readtofavors function and test file with FileReader function. The reason why we use different function for train data is that we have to make preference information as a certain dictionary for the ease of computing similarity. Next step is to run user-based CF algorithm on train and test data. The result is stored in u_Res variable and we write the output file with u_Res with a predefined output name.

[Readtofavors & FileReader function]

```python
def Readtofavors(file_path):

    favors = {}
    for line in open(file_path):
        (user, movie, rating, ts) = line.split('\t')
        favors.setdefault(user, {})
        favors[user][movie] = float(rating)
    return favors


def FileReader(file_path):
    df = []
    for row in open(file_path):
        (user, movie, rating, ts) = row.split('\t')
        df.append((user, movie, rating, ts))
    return df
```

The first function reads the u#.base file and make a dictionary which contains the user preference data. The other function opens the test data file by iterating each row of the data. Watch that the delimeter is '₩t'.

[make user prediction function]

```python
# Read each row in u#.test data and
# Make user-based recommendation to each user-item combination
def make_user_prediction(train, test):

    user_Total = []
    check = set()

    for (user, movie, rating, ts) in test:
        # The test file is sorted by user
        # Therefore we only have to make user_recommendation set
        # when each user appears for the first time
        if user not in check:
            Res = User_GetRec(train, user)

        # Make a check set to avoid repeated recommendation for same user
        check.add(user)

        # Compare the test movie in recommendation set
        get_rate = [rating for (rating, p_movie) in Res if p_movie == movie]
        if len(get_rate) == 0: get_rate = [5]
        row = [user, movie, get_rate[0]]
        user_Total.append(row)


    return user_Total
```

First it starts with reading each rows in u#.test data. Then it makes user-based recommendation to each user-item combinations. As the test file is sorted by user, we only have to make the set for user-based recommendation set whenever each user appears for the first time. Therefore, I made a set named 'check' to check whether we already made a recommendation set or not whenever a new row of test data comes. Then it compares the movie of each row of test data and find the matching movie in the user-based recommendation set. Finally, it returns the user_Total instance which contains the required data for the output file.

[UserGetRec function]

```
# Gets recommendations for a person by using a weighted average
# of every other user's ratings
def User_GetRec(favors, person, similarity=sim_pearson):
    totals = {}
    simSums = {}
    for not_me in favors:
        # don't compare me to myself
        if not_me == person: continue
        sim = similarity(favors, person, not_me)

        # ignore scores of zero or lower
        if sim <= 0: continue
        for item in favors[not_me]:

            # only need to evaluate movies I haven't seen yet
            if item not in favors[person] or favors[person][item] == 0:
                totals.setdefault(item, 0)
                totals[item] += favors[not_me][item] * sim

                simSums.setdefault(item, 0)
                simSums[item] += sim

    # Make weighted average based on the correlation information
    SimSet = [(total / simSums[item], item) for item, total in totals.items()]

    # Return the sorted list
    SimSet.sort()
    SimSet.reverse()
    return SimSet
```

This function gets recommendations for each person by using a weighted average of every other user's ratings. As we don't have to compare me to myself, it skips when it finds the target person. Then it computes the similarity or Pearson

coefficient. Plus, we don't have to make a recommendation on the movie that the target person has already watched, so we skip that kind of cases. Then we calculate the weighted average based on the information that we calculated the correlation with every other people available from the train data. For the ease of searching, we sort the similarity set and return it.

[sim_pearson function]

```
# Returns the Pearson correlation coefficient for a pair
def sim_pearson(favors, p1, p2):
    # Get the list of mutually rated items
    share = {}
    for item in favors[p1]:
        if item in favors[p2]: share[item] = 1

    # When there is no ratings in common
    if len(share) == 0:
        return 0

    n = len(share)

    # Sums of all the preferences
    sum1 = sum([favors[p1][it] for it in share])
    sum2 = sum([favors[p2][it] for it in share])

    # Sums of the squares
    sum1Sq = sum([pow(favors[p1][it], 2) for it in share])
    sum2Sq = sum([pow(favors[p2][it], 2) for it in share])

    # Sum of the products
    pSum = sum([favors[p1][it] * favors[p2][it] for it in share])

    # Calculate r (Pearson score)
    num = pSum - (sum1 * sum2 / n)
    den = sqrt((sum1Sq - pow(sum1, 2) / n) * (sum2Sq - pow(sum2, 2) / n))
    if den == 0:
        return 0

    pr = num / den

    return pr
```

This function calculates the Pearson correlation coefficient for each given pair of people. First, it gets the list of mutually rated items. Then it calculates the Pearson correlation coefficient. When there is no common movies for the two people, it returns 0 as coefficient.

[write_file function]

```python
def write_file(res, f):
    # Loop through each element
    # and write a row in a given format
    attr_len = len(res)
    row_len = len(res[0])
    for i in xrange(attr_len):
        row = ""
        for j in xrange(row_len):
            row += "%s\t" % (res[i][j])
        row = row[:-1]
        row += "\n"

        f.write(row)

    return f
```

This function loops through each element of u_Res file which is already includes necessary information of output file and write each row of output file in a given format.

3) Instructions for compiling the source codes

[Step 1]:

copy the repository and change directory to /project_recommendation

```
?git clone http://hconnect.hanyang.ac.kr/2017_ITE4005_10065/2017_ITE4005_2012023871.git
```

- Copy my gitlab repository by using *git clone* or download the *zip file* from the website. [http://hconnect.hanyang.ac.kr/2017_ITE4005_10065/2017_ITE4005_2012023871]

```
cd ~/DS_git/2017_ITE4005_2012023871/Programming_Assignment_4/project_recommendation/
```

- go the the project_recommendation directory by 'cd' command.

[Step 2] : Run the code with python console(2.7 ONLY)

```
python recommender.py u1.base u1.test
python recommender.py u2.base u2.test
python recommender.py u3.base u3.test
python recommender.py u4.base u4.test
python recommender.py u5.base u5.test
```

run $python recommender.py u1.based u1.test on terminal or command line

[Caution] python should be 2.7 # python3 will cause error

Then you will see the output files have been made in the same directory.

- recommender.py
- u1.base
- u1.base_prediction.txt
- u1.test
- u2.base
- u2.base_prediction.txt
- u2.test
- u3.base
- u3.base_prediction.txt
- u3.test
- u4.base
- u4.base_prediction.txt
- u4.test
- u5.base
- u5.base_prediction.txt
- u5.test