Rudra_Ronit_CS422_HW4_Practicum

November 30, 2016

0.0.1 Importing Modules

```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.metrics.pairwise import cosine_distances,cosine_similarity
```

0.0.2 Read in required data (Referenced from Notebook5)

```
In [2]: rating_cols = ['user_id',
                       'movie_id',
                       'rating',
                       'timestamp']
        ratings = pd.read_csv('ml-100k/u.data',sep='\t',names=rating_cols)

        user_cols = ['user_id',
                     'age',
                     'gender',
                     'occupation',
                     'zip_code']
        users = pd.read_csv('ml-100k/u.user',sep='|',names=user_cols)

        item_cols = ['movie id',
                     'movie title',
                     'release date',
                     'video release date',
                     'IMDb URL',
                     'Unknown',
                     'Action',
                     'Adventure',
                     'Animation',
                     'Childrens',
                     'Comedy',
                     'Crime',
                     'Documentary',
                     'Drama',
                     'Fantasy',
                     'FilmNoir',
                     'Horror',
```

```
                  'Musical',
                  'Mystery',
                  'Romance',
                  'SciFi',
                  'Thriller',
                  'War',
                  'Western']
items = pd.read_csv('ml-100k/u.item',
                    sep='|',
                    names=item_cols,
                    encoding='latin-1')
```

In [3]:
```
users.head()
```

Out[3]:
```
   user_id  age gender  occupation zip_code
0        1   24      M  technician    85711
1        2   53      F       other    94043
2        3   23      M      writer    32067
3        4   24      M  technician    43537
4        5   33      F       other    15213
```

In [4]:
```
ratings.head()
```

Out[4]:
```
   user_id  movie_id  rating  timestamp
0      196       242       3  881250949
1      186       302       3  891717742
2       22       377       1  878887116
3      244        51       2  880606923
4      166       346       1  886397596
```

In [5]:
```
items.head()
```

Out[5]:
```
   movie id        movie title release date  video release date  \
0         1   Toy Story (1995)  01-Jan-1995                 NaN
1         2   GoldenEye (1995)  01-Jan-1995                 NaN
2         3  Four Rooms (1995)  01-Jan-1995                 NaN
3         4  Get Shorty (1995)  01-Jan-1995                 NaN
4         5     Copycat (1995)  01-Jan-1995                 NaN

                                            IMDb URL  Unknown  Action  \
0  http://us.imdb.com/M/title-exact?Toy%20Story%2...        0       0
1  http://us.imdb.com/M/title-exact?GoldenEye%20(...        0       1
2  http://us.imdb.com/M/title-exact?Four%20Rooms%...        0       0
3  http://us.imdb.com/M/title-exact?Get%20Shorty%...        0       1
4  http://us.imdb.com/M/title-exact?Copycat%20(1995)        0       0

   Adventure  Animation  Childrens   ...     Fantasy  FilmNoir  Horror  \
0          0          1          1   ...           0         0       0
1          1          0          0   ...           0         0       0
```

```
2           0            0            0  ...            0           0        0
3           0            0            0  ...            0           0        0
4           0            0            0  ...            0           0        0

   Musical  Mystery  Romance  SciFi  Thriller  War  Western
0        0        0        0      0         0    0        0
1        0        0        0      0         1    0        0
2        0        0        0      0         1    0        0
3        0        0        0      0         0    0        0
4        0        0        0      0         1    0        0

[5 rows x 24 columns]
```

In [6]:
```python
utility = ratings.pivot(index='user_id',
                        columns='movie_id',
                        values='rating')
```

In [7]:
```python
# user id and movie id is index+1 and column+1
utility.head(10)
```

Out[7]:
```
movie_id  1     2     3     4     5     6     7     8     9     10    ...
user_id                                                               ...
1          5.0   3.0   4.0   3.0   3.0   5.0   4.0   1.0   5.0   3.0  ...
2          4.0   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   2.0  ...
3          NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN  ...
4          NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN  ...
5          4.0   3.0   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN  ...
6          4.0   NaN   NaN   NaN   NaN   NaN   2.0   4.0   4.0   NaN  ...
7          NaN   NaN   NaN   5.0   NaN   NaN   5.0   5.0   5.0   4.0  ...
8          NaN   NaN   NaN   NaN   NaN   NaN   3.0   NaN   NaN   NaN  ...
9          NaN   NaN   NaN   NaN   NaN   5.0   4.0   NaN   NaN   NaN  ...
10         4.0   NaN   NaN   4.0   NaN   NaN   4.0   NaN   4.0   NaN  ...

movie_id  1673  1674  1675  1676  1677  1678  1679  1680  1681  1682
user_id
1          NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
2          NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
3          NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
4          NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
5          NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
6          NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
7          NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
8          NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
9          NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
10         NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN

[10 rows x 1682 columns]
```

1 Question 1

Load the Movielens 100k dataset (ml-100k.zip) into Python using Pandas dataframes. Build a user profile on unscaled data for both users 200 and 15 , and calculate the cosine similarity and distance between the user's preferences and the item/movie 95. Which user would a recommender system suggest this movie to?

1.0.1 Extract ratings for users 200 and 15

```
In [8]: user_200 = ratings.iloc[np.where(ratings["user_id"]==200)]
        user_15 = ratings.iloc[np.where(ratings["user_id"]==15)]
        #extract list of movie ids rated for each user
        user_200_movies = np.array(user_200["movie_id"])
        user_15_movies = np.array(user_15["movie_id"])

In [9]: user_200.head()

Out[9]:      user_id  movie_id  rating  timestamp
        12       200       222       5  876042340
        189      200       673       5  884128554
        243      200       318       5  884128458
        326      200       304       5  876041644
        367      200        96       5  884129409
```

1.0.2 Extract genre features

Only the columns containing genre are exctracted.

```
In [10]: features = items.iloc[:,5:]

In [11]: # Movie id is index + 1
         features.head()

Out[11]:    Unknown  Action  Adventure  Animation  Childrens  Comedy  Crime  \
         0        0       0          0          1          1       1      0
         1        0       1          1          0          0       0      0
         2        0       0          0          0          0       0      0
         3        0       1          0          0          0       1      0
         4        0       0          0          0          0       0      1

            Documentary  Drama  Fantasy  FilmNoir  Horror  Musical  Mystery  Romanc
         0            0      0        0         0       0        0        0
         1            0      0        0         0       0        0        0
         2            0      0        0         0       0        0        0
         3            0      1        0         0       0        0        0
         4            0      1        0         0       0        0        0

            SciFi  Thriller  War  Western
         0      0         0    0        0
```

```
1       0           1     0         0
2       0           1     0         0
3       0           0     0         0
4       0           1     0         0
```

1.0.3 Average out utility matrix

Take row averages of the utility matrix to center it for each user.

```
In [12]: user_means = utility.mean(axis=1)
         utility_centered = utility - user_means
         utility_centered = utility_centered.where((pd.notnull(utility_centered))
                                                            ,0)
```

```
In [13]: utility_centered.head()
```

```
Out[13]:            1         2         3         4         5         6         7
         user_id
         1        1.389706 -0.709677  1.203704 -1.333333  0.125714  1.364929  0.034
         2        0.389706  0.000000  0.000000  0.000000  0.000000  0.000000  0.000
         3        0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000
         4        0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000
         5        0.389706 -0.709677  0.000000  0.000000  0.000000  0.000000  0.000

                   8         9         10     ...   1673  1674  1675  1676  1677
         user_id                              ...
         1       -2.79661  0.727273 -1.206522  ...    0.0   0.0   0.0   0.0   0.0
         2        0.00000  0.000000 -2.206522  ...    0.0   0.0   0.0   0.0   0.0
         3        0.00000  0.000000  0.000000  ...    0.0   0.0   0.0   0.0   0.0
         4        0.00000  0.000000  0.000000  ...    0.0   0.0   0.0   0.0   0.0
         5        0.00000  0.000000  0.000000  ...    0.0   0.0   0.0   0.0   0.0

                  1678  1679  1680  1681  1682
         user_id
         1         0.0   0.0   0.0   0.0   0.0
         2         0.0   0.0   0.0   0.0   0.0
         3         0.0   0.0   0.0   0.0   0.0
         4         0.0   0.0   0.0   0.0   0.0
         5         0.0   0.0   0.0   0.0   0.0

         [5 rows x 1682 columns]
```

1.0.4 Generate user profiles for users 200 and 15

Multiply the centered utility matrix row of both users to the feature dataframe containing feature vector for each movie. Since the movies not rated by the user will have rating of zero, the profile generated would not be counted towards the overall user profile. This makes it safe to multiply the ratings with the entire feature dataframe. Note that the row indices are user id - 1.

```
In [14]: user_200_profile=((features.values*utility_centered.iloc[199,
                           :].values.reshape(-1,1)).sum(axis=0)).reshape(1,-1)
         user_15_profile=((features.values*utility_centered.iloc[14,
                           :].values.reshape(-1,1)).sum(axis=0)).reshape(1,-1)
```

1.0.5 Extract vector for item 95

```
In [15]: item_95 = features.iloc[95,:].reshape(1,-1)

In [16]: print("For User 200:")
         print("Cosine Distance is %s and Cosine Similarity is %s"
               % (cosine_distances(user_200_profile,item_95)[0,0],cosine_similarity
```

```
For User 200:
Cosine Distance is 0.293108142037 and Cosine Similarity is 0.706891857963
```

```
In [17]: print("For User 15:")
         print("Cosine Distance is %s and Cosine Similarity is %s"
               % (cosine_distances(user_15_profile,item_95)[0,0],cosine_similarity
```

```
For User 15:
Cosine Distance is 1.62448866308 and Cosine Similarity is -0.624488663084
```

The system would recommend movie/item 95 to user 200 as the similarity score is higher/distance score is lower.

2 Question 2

2.0.1 Utility matrix for data has already been generated

```
In [18]: utility_centered.head()
```

```
Out[18]:                 1         2         3         4         5         6         7
         user_id
         1        1.389706 -0.709677  1.203704 -1.333333  0.125714  1.364929  0.034
         2        0.389706  0.000000  0.000000  0.000000  0.000000  0.000000  0.000
         3        0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000
         4        0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000
         5        0.389706 -0.709677  0.000000  0.000000  0.000000  0.000000  0.000

                        8         9        10   ...   1673  1674  1675  1676  1677
         user_id                                ...
         1        -2.79661  0.727273 -1.206522  ...    0.0   0.0   0.0   0.0   0.0
         2         0.00000  0.000000 -2.206522  ...    0.0   0.0   0.0   0.0   0.0
         3         0.00000  0.000000  0.000000  ...    0.0   0.0   0.0   0.0   0.0
         4         0.00000  0.000000  0.000000  ...    0.0   0.0   0.0   0.0   0.0
         5         0.00000  0.000000  0.000000  ...    0.0   0.0   0.0   0.0   0.0
```

```
           1678  1679  1680  1681  1682
user_id
1           0.0   0.0   0.0   0.0   0.0
2           0.0   0.0   0.0   0.0   0.0
3           0.0   0.0   0.0   0.0   0.0
4           0.0   0.0   0.0   0.0   0.0
5           0.0   0.0   0.0   0.0   0.0

[5 rows x 1682 columns]
```

2.0.2 Find similar users for user 1

```
In [19]: similar_users=cosine_similarity(utility_centered.iloc[0,
                     :].values.reshape(1,-1),utility_centered.values).reshape(-1,)

In [20]: users['similarity'] = pd.Series(similar_users, index=users.index)

In [21]: users = users.sort_values("similarity",ascending=0)
```

Top 10 users similar to User 1 are:

```
In [22]: users.iloc[1:11,:]

Out[22]:      user_id  age gender  occupation zip_code  similarity
         737      738   35      M  technician    95403    0.291487
         591      592   18      M     student    97520    0.278402
         275      276   21      M     student    95064    0.268151
         266      267   23      M    engineer    83716    0.264761
         642      643   39      M   scientist    55122    0.264003
         756      757   26      M     student    55104    0.262368
         456      457   33      F    salesman    30011    0.262337
         605      606   28      M  programmer    63044    0.260847
         915      916   27      M    engineer    N2L5N    0.255624
         43        44   26      M  technician    46260    0.252954

In [23]: top_10=np.array(users.iloc[1:11,:]["user_id"].index)
```

Ratings of these similar users for item 508 are:

```
In [24]: utility.iloc[top_10,507]

Out[24]: user_id
         738     NaN
         592     5.0
         276     5.0
         267     NaN
         643     4.0
         757     NaN
```

```
457     NaN
606     4.0
916     NaN
44      NaN
Name: 508, dtype: float64
```

Note that item id = column index + 1

```
In [25]: print("The Expected Rating of User 1 for Item 508:")
         print("based on average rating of it's similar users is %s."
               % (utility.iloc[top_10,507].mean()))
```

```
The Expected Rating of User 1 for Item 508:
based on average rating of it's similar users is 4.5.
```