Practical Data Science: Recommender Systems II

Dr. Yongli Ren

(yongli.ren@rmit.edu.au)

Computer Science & IT

School of Science

Outline

- Part 1: What is Collaborative Filtering?
- Part 2: How to Do Collaborative Filtering?

Practical Data Science – COSC2670

PART 1: WHAT IS COLLABORATIVE FILTERING?

Collaborative Filtering - At Amazon

Roll over image to zoom in

See all 14 customer images

Share your own customer images

Apple iPhone 5 16GB (Black)

by Apple

(151 customer reviews)

List Price: ~~$750.00~~

Price: $696.95

You Save: $53.05 (7%)

Only 12 left in stock.

Ships from and sold by PRIME ELECTRONICS.

125 new from $480.00 65 used from $500.00

Thinking of Selling Your iPho
Now Amazon has three easy wa
for Cash as an Individual Seller,

The principle is like this:

if several members of my community owned and liked the latest Apple gadget, then it is highly likely that I will too.

Customers Who Bought This Item Also Bought

Page 1 of 15

Apple iPhone 5 16GB (White) - Unlocked
(187)
$709.00
Add to Cart
In Stock. Ships from and sold by WorldWide Distributors.

MPERO® 5 Pack of Screen Protectors for New Apple iPhone 5
(909)
$1.49

Apple iPhone 5 16GB (Black) - AT&T
(10)
$719.00

Apple iPhone 5 16GB (Black) - Verizon Wireless
(13)
$719.00

Samsung Galaxy S IV/S4 GT-I9500 Factory Unlocked Phone - International Version ...
(60)
$624.99

Samsung Galaxy S IV/S4 GT-I9500 Factory Unlocked Phone - International Version ...
(104)
$616.00

BLACKBERRY Z10 16GB WHITE FACTORY UNLOCKED GSM
(41)
$467.95

What is Collaborative Filtering?

- Collaborative filtering (CF)
  - is a technique used by recommender systems
    - the most successful recommendation technique to date
  - is the idea of recommending an item depending on other like minded individuals
  - consists of
    - set of users,
    - set of items, and
    - set of opinions about the item: ratings, reviews or purchases.

User-Item Rating Matrix

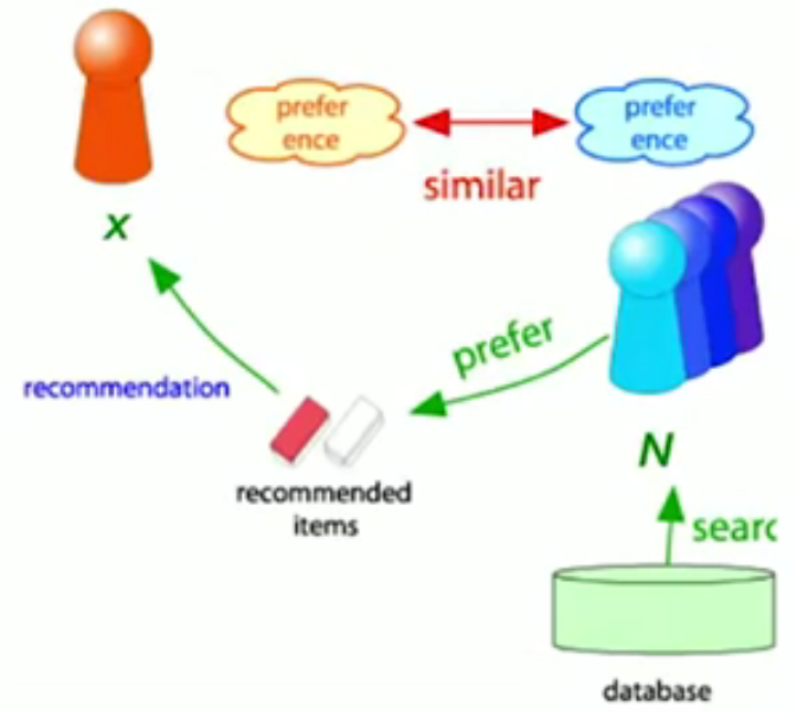
	Troy (Action)	The Godfather (Crime)	Titanic (Romantic)	Forrest Gump (Comedy)
Fahime	5	⊘	5	1
Musi	5	⊘	⊘	1
Hamza	4	4	5	1
Paul	4	⊘	5	5
Adam	1	2	⊘	5

What is Collaborative Filtering?

- In the narrower sense,
  - collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user
    - by collecting preferences or taste information from many users (collaborating).
  - The underlying assumption is that
    - if person A has the same opinion as person B on an issue,
    - A is more likely to also have B's opinion on a different issue than that of a randomly chosen person.
  - This differs from *MovieAvg* (or *TopPop*)
    - The simpler approach of giving an average score (or popularity) for each item of interest.

What is Collaborative Filtering?

- Consider the active user x;
- find set N of other users whose ratings are "similar" to x's ratings;
- Estimate x's ratings based on ratings of users in N

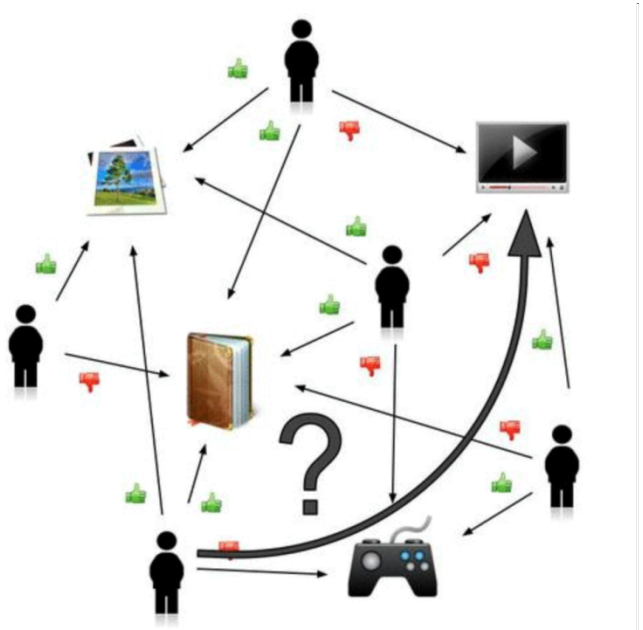


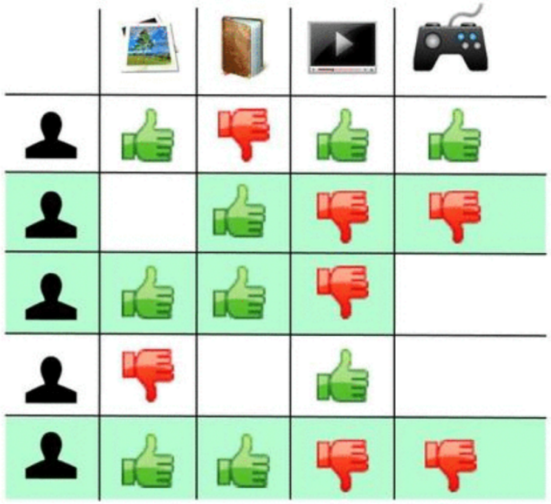
https://www.youtube.com/watch?v=h9gpufJFF-0

Practical Data Science – COSC2670

PART 2:
HOW TO DO COLLABORATIVE FILTERING?

The Process of Collaborative Filtering





This image shows an example of predicting of the user's rating using collaborative filtering.

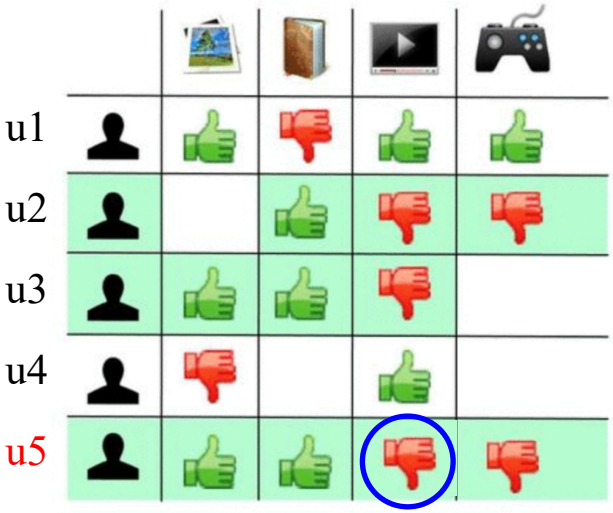
1. People rate different items.
2. The system is making predictions about user's rating for an item, which the user hasn't rated yet.
3. These predictions are built upon the existing ratings of other users, who have similar ratings with the active user.

For instance, in our case the system has made a prediction, that the active user won't like the video.

https://en.wikipedia.org/wiki/Collaborative_filtering

kNN-based Collaborative Filtering

- Input: the User-Item rating matrix.
- Output: the rating of the active user on the target item



	u1	u2	u3	u4
u5	1	2	2	0

The number of items they have common opinions

For the active user

Similarities to other users

Find the k-nearest neighbours

Aggregate neighbours' ratings

Predict rating on target item

https://en.wikipedia.org/wiki/Collaborative_filtering

Similar Users

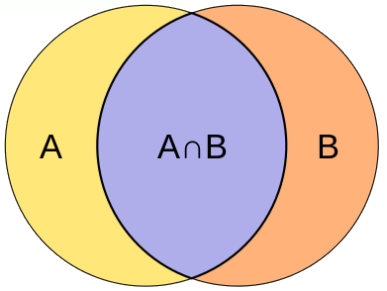
	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- Consider users A and B with rating vectors (row A and B)
- We need a similarity metric $sim\ (A,B)$:
  - Capture intuition that $sim(A,B) > sim(A,C)$

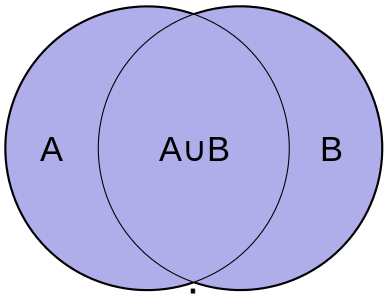
Similar Users
-Jaccard Similarity

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- $sim(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$
- $sim(A,B) = 1/5;\ sim(A,C) = 2/4;$
  - $sim(A,B) < sim(A,C)$
- Problem: ignores rating values



Intersection



union

https://en.wikipedia.org/wiki/Jaccard_index https://www.youtube.com/watch?v=h9gpufJFF-0

Similar Users
-Cosine Similarity

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4	0	0	5	1	0	0
B	5	5	4	0	0	0	0
C	0	0	0	2	4	5	0
D	0	3	0	0	0	0	3

- $sim(A,B) = cos(A, B)$
- If missing values are treated as zero
  - $sim(A,B) = 0.38$, $sim(A,C) = 0.32$;
  - $sim(A,B) > sim(A,C)$, but not by much
- Problem: treats missing ratings as negative

Similar Users
-Centered Cosine Similarity

- Normalize ratings by subtracting row mean

	HP1	HP2	HP3	TW	SW1	SW2	SW3	
A	4			5	1			(4+5+1)/3
B	5	5	4					(5+5+4)/3
C				2	4	5		(2+4+5)/3
D		3					3	(3+3)/2

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	$2/3$			$5/3$	$-7/3$		
B	$1/3$	$1/3$	$-2/3$				
C				$-5/3$	$1/3$	$4/3$	
D		0					0

Similar Users
-Centered Cosine Similarity

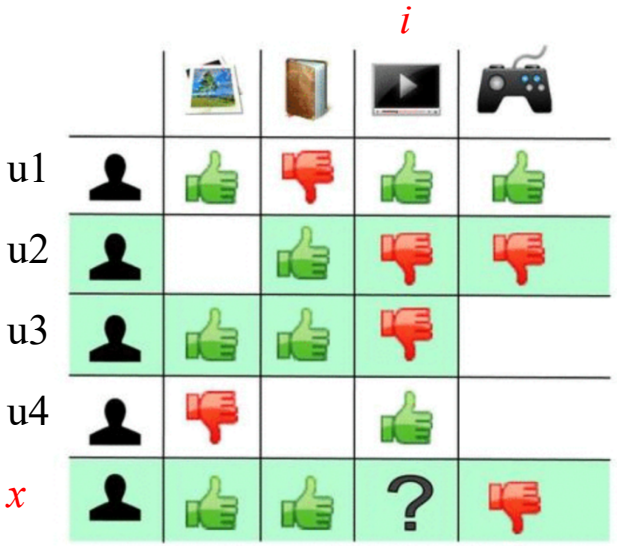
	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	2/3			5/3	−7/3		
B	1/3	1/3	−2/3				
C				−5/3	1/3	4/3	
D		0					0

- $sim(A,B) = cos(A, B) = 0.09$
- $sim(A, C) = -0.56$
- $sim(A,B) > sim(A,C)$
- Captures intuition better
  - Missing ratings treated as "average"
  - Handles "tough raters" and "easy raters"
- Also known as "Pearson Correlation"

https://www.youtube.com/watch?v=h9gpufJFF-0

Rating Predictions

- Let r_x be the vector of user x's ratings
- let N be the set of k users most similar to x who have also rated item I
  - $N = \{u2, u3\}$
- Prediction for users x and item i
  - option 1: average rating in the neighbourhood;
  - (👎 + 👎) / 2 = 👎
  - option 2: weighted average rating in the neighbourhood
  - (2 * 👎 + 2 * 👎) / (2+2) = 👎

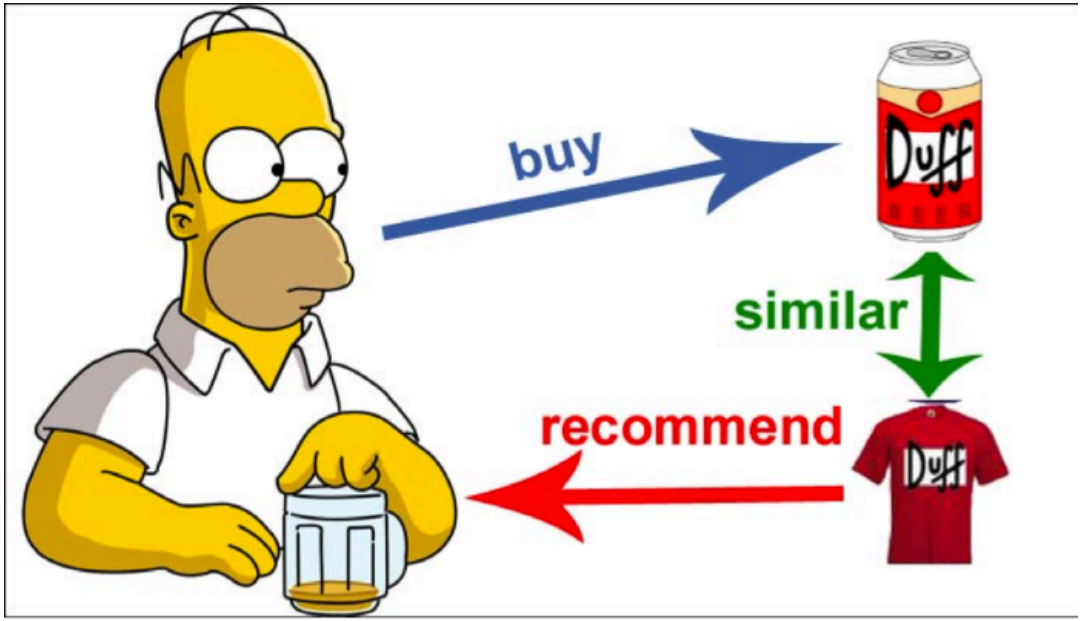


	u1	u2	u3	u4
x	1	2	2	0

https://www.youtube.com/watch?v=h9gpufJFF-0

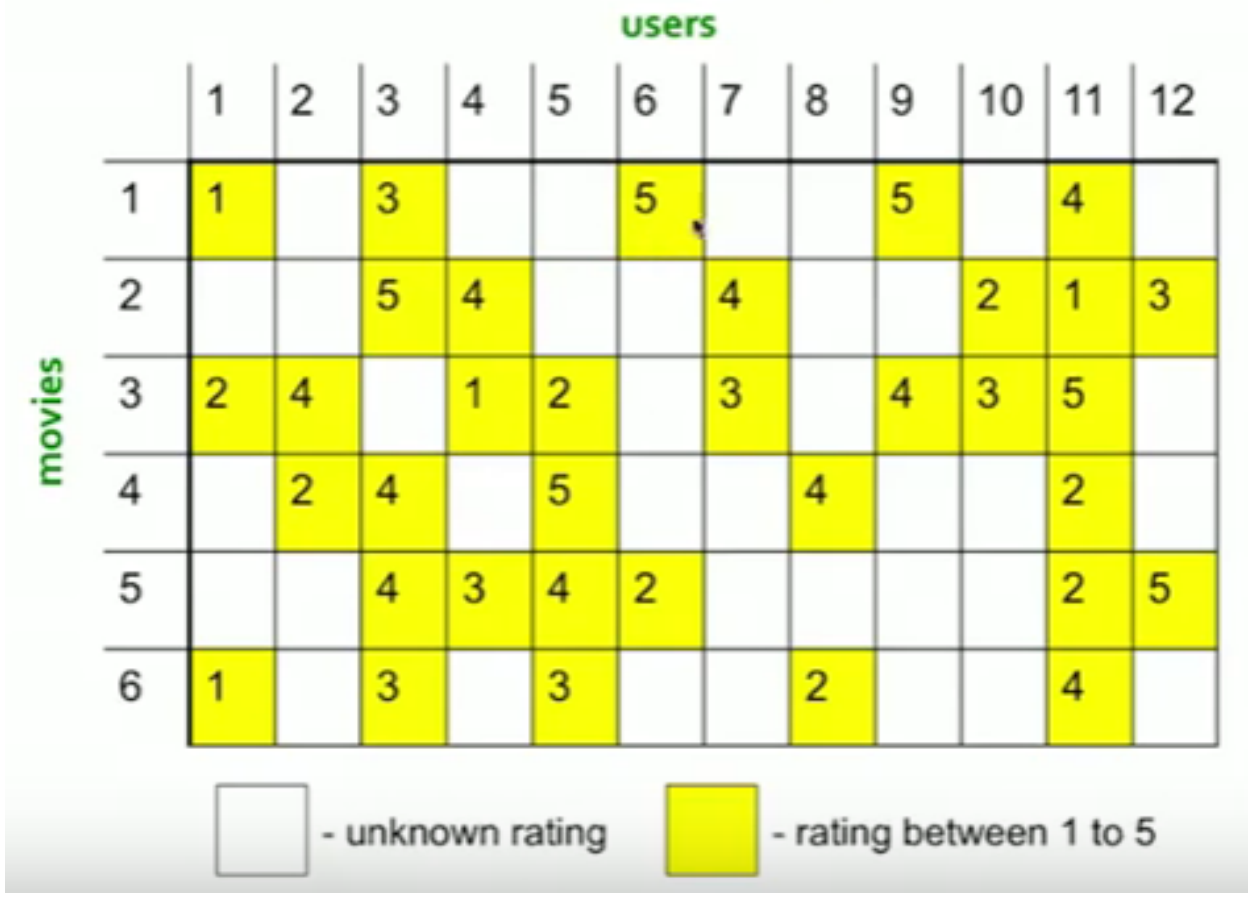
Item-item Collaborative Filtering

- So far: user-user collaborative filtering
- Another view: item-item
  - for item i, find other similar items
  - estimate rating for item i based on ratings for similar items;
  - can use same similarity metrics and prediction functions as in user-user model



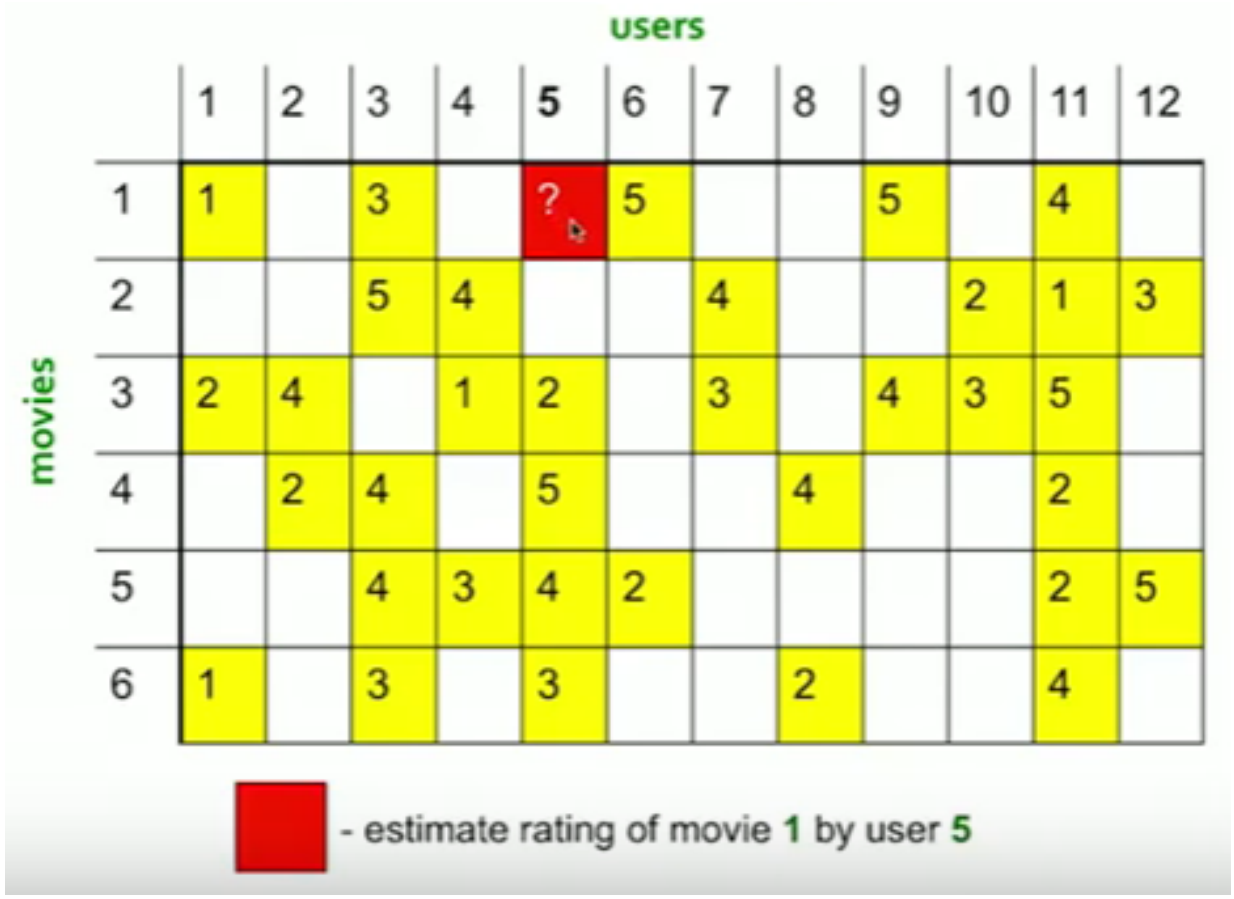
https://gouache-design.ru/collaborative-filter.php https://www.youtube.com/watch?v=h9gpufJFF-0

Item-item Collaborative Filtering (|N| = 2)



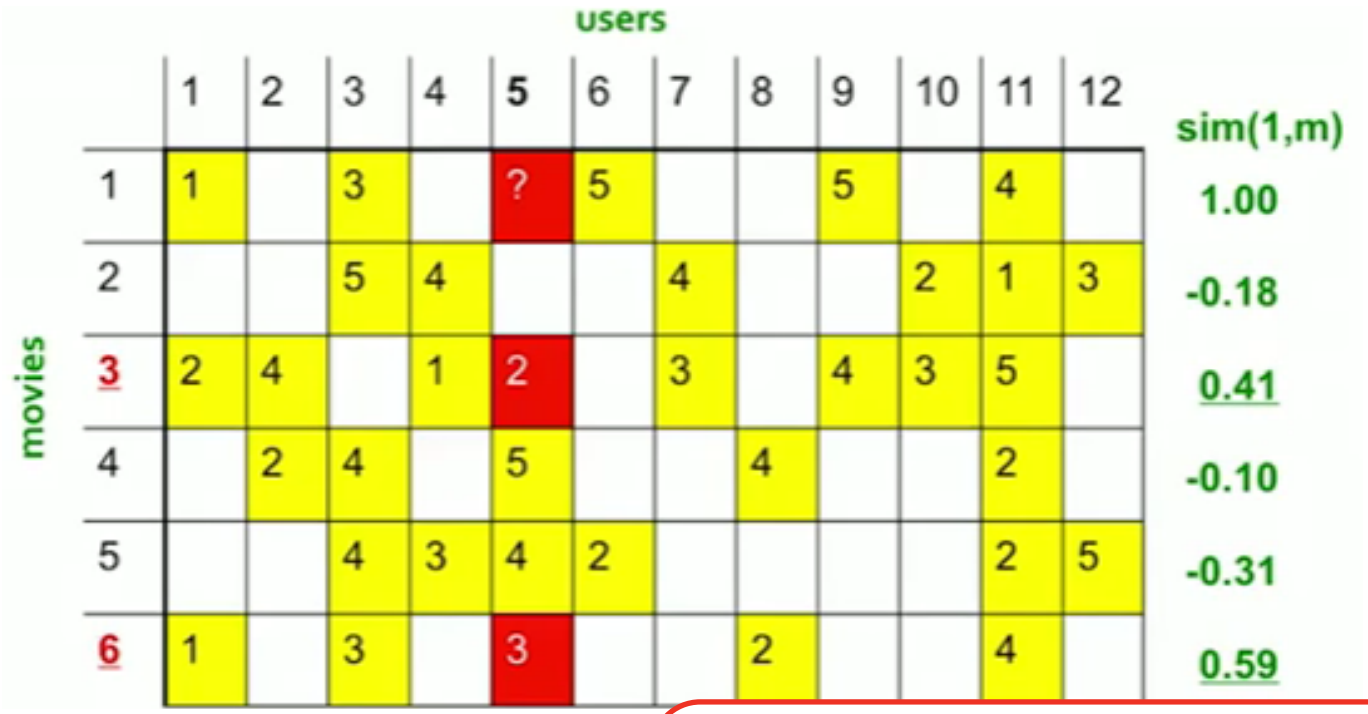
https://www.youtube.com/watch?v=h9gpufJFF-0

Item-item Collaborative Filtering (|N| = 2)



https://www.youtube.com/watch?v=h9gpufJFF-0

Item-item Collaborative Filtering (|N| = 2)



Neighbour selection:
Identify movies similar to movie 1, rated by user 5

Here, we use Centered Cosine as similarity:

1) Subtract mean rating from each movie:
   meanRating1 = (1+3+5+5+4)/5 = 3.6
   row 1 = [-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4)
2) Compute cosine similarities between rows

Predict by taking weighted average:
(0.41 * 2 + 0.59 * 3)/(0.41 + 0.59) = 2.6

https://www.youtube.com/watch?v=h9gpufJFF-0

User-User vs Item-Item

- In theory,
  - user-user and item-item are dual approaches.
- in practice,
  - item-item outperforms user-user in many user cases.
- items are "simpler" than users
  - items belong to a small set of "genres",
  - users have varied tastes;
  - item similarity is more meaningful than user similarity

User-based Collaborative Filtering with Python

```
import numpy as np
import pandas as pd
```

Load packages

```
cd ml-100k/
```

Move into the data directory

```
/Users/yongli/A-Teaching/PDS/code/recsys/ml-100k
```

```
ls
```

List the files in the ml-100k folder

```
README          u.info          u1.test         u4.base         ua.test
allbut.pl*      u.item          u2.base         u4.test         ub.base
mku.sh*         u.occupation    u2.test         u5.base         ub.test
u.data          u.user          u3.base         u5.test
u.genre         u1.base         u3.test         ua.base
```

```
!head u.data
```

Print the first 10 lines in u.data:
user id | item id | rating | timestamp.

```
196     242     3       881250949
186     302     3       891717742
22      377     1       878887116
244     51      2       880606923
166     346     1       886397596
298     474     4       884182806
115     265     2       881171488
253     465     5       891628467
305     451     3       886324817
6       86      3       883603013
```

User-based Collaborative Filtering with Python

```
names = ['user_id', 'item_id', 'rating', 'timestamp']
df = pd.read_csv('u.data', sep='\t', names=names)
df.head()
```

	user_id	item_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

Load the rating data

User-based Collaborative Filtering with Python

Load the item data

```
i_cols = ['movie id', 'movie title' ,'release date','video release date', 'IMDb URL', 'unknown', 'Action', 'Adventure',
 'Animation', 'Children\'s', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy',
 'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western']
items = pd.read_csv('u.item', sep='|', names=i_cols, encoding='latin-1')

items.head()
```

	movie id	movie title	release date	video release date	IMDb URL	unknown	Action	Adventure	Animation	Children's	...	Fantasy	Film-Noir	Horror	Mus
0	1	Toy Story (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%2...	0	0	0	1	1	...	0	0	0	0
1	2	GoldenEye (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?GoldenEye%20(...	0	1	1	0	0	...	0	0	0	0
2	3	Four Rooms (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Four%20Rooms%...	0	0	0	0	0	...	0	0	0	0
3	4	Get Shorty (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Get%20Shorty%...	0	1	0	0	0	...	0	0	0	0
4	5	Copycat (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Copycat%20(1995)	0	0	0	0	0	...	0	0	0	0

5 rows × 24 columns

User-based Collaborative Filtering with Python

```
n_users = df.user_id.unique().shape[0]
n_items = df.item_id.unique().shape[0]
print(str(n_users) + 'users')
print(str(n_items) + 'items')
```

```
943 users
1682 items
```

Check the size of the data

```
ratings = np.zeros((n_users, n_items))
for row in df.itertuples():
    ratings[row[1]-1, row[2]-1] = row[3]

ratings_df = pd.DataFrame(ratings)
```

Construct the user-item rating matrix: ratings

Then, use 'for' loop statement to convert data into a matrix

Finally, convert the 'ratings' to a dataframe.

User-based Collaborative Filtering with Python

Load the cosine_similarity package to calculate the pairwise cosine similarity between users.

```
from sklearn.metrics.pairwise import cosine_similarity
userSimilarity = cosine_similarity(ratings, ratings)

data_ubs_fast = pd.DataFrame(userSimilarity)
data_ubs_fast.head()
```

Finally, convert the numpy array 'userSimilarity' to a dataframe 'data_ubs_fast'

	0	1	2	3	4	5	6	7	8	9	...
0	1.000000	0.166931	0.047460	0.064358	0.378475	0.430239	0.440367	0.319072	0.078138	0.376544	...
1	0.166931	1.000000	0.110591	0.178121	0.072979	0.245843	0.107328	0.103344	0.161048	0.159862	...
2	0.047460	0.110591	1.000000	0.344151	0.021245	0.072415	0.066137	0.083060	0.061040	0.065151	...
3	0.064358	0.178121	0.344151	1.000000	0.031804	0.068044	0.091230	0.188060	0.101284	0.060859	...
4	0.378475	0.072979	0.021245	0.031804	1.000000	0.237286	0.373600	0.248930	0.056847	0.201427	...

User-based Collaborative Filtering with Python

Create a dataframe 'data_pre' to store the prediction of user ratings on un-rated items.

Nested 'for' loop statement to make prediction for all users

```
data_pre = pd.DataFrame(index=range(0,n_users),columns=items['movie title'])

epsilon=1e-9

for i in len(data_pre.index):
    for j in range(0,len(data_pre.columns)):
        if ratings_df.iloc[i][j] > 0:
            data_pre.iloc[i][j] = 0
        else:
            mask_rated_users = ratings_df[j] > 0
            top_neighbours = data_ubs_fast.loc[mask_rated_users,i].sort_values(ascending=False)[0:10].index
            top_neighbours_sim = data_ubs_fast.loc[mask_rated_users,i].sort_values(ascending=False)[0:10].values
            neighbours_ratings = ratings_df.iloc[top_neighbours,j]

            data_pre.iloc[i][j] = sum(neighbours_ratings*top_neighbours_sim)/(sum(top_neighbours_sim) + epsilon)
```

Small value

If the active user i rated item j already, we exclude item j from prediction/recommendations by setting its value to zero

Select users who rated j with a mask

Select k nearest neighbour's ratings on j.

Final prediction on j for user i by using weighted sum formula.

User-based Collaborative Filtering with Python

Create a dataframe to store the final top-6 recommendations

Make recommendations to each user by using 'for' loop statement to sort the predictions for each user

```
data_recommend = pd.DataFrame(index=data_pre.index, columns=['1','2','3','4','5','6'])

for i in range(0,len(data_pre.index)):
    data_recommend.ix[i,0:] = data_pre.ix[i,:].sort_values(ascending=False)[0:6].index.transpose()

data_recommend.head(3)
```

	1	2	3	4	5	6
0	Prefontaine (1997)	Saint of Fort Washington, The (1993)	Golden Earrings (1947)	Anna (1996)	Some Mother's Son (1996)	Santa with Muscles (1996)
1	Letter From Death Row, A (1998)	Prefontaine (1997)	Star Kid (1997)	Little City (1998)	The Deadly Cure (1996)	Some Mother's Son (1996)
2	Braveheart (1995)	Star Kid (1997)	Hugo Pool (1997)	Letter From Death Row, A (1998)	Prefontaine (1997)	Little City (1998)

References and Further Reading

- Xiaoyuan Su and Taghi M. Khoshgoftaar, “A Survey of Collaborative Filtering Techniques,” Advances in Artificial Intelligence, vol. 2009, Article ID 421425, 19 pages, 2009. doi:10.1155/2009/421425
- Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Trans. on Knowl. and Data Eng. 17, 6 (June 2005), 734-749. DOI: https://doi.org/10.1109/TKDE.2005.99
- Yongli Ren, Gang Li, Wanlei Zhou: A survey of recommendation techniques based on offline data processing. Concurrency and Computation: Practice and Experience 27(15): 3915-3942 (2015)

Data
Science

Thanks!