20 - Recommender Systems - 1

Data Science Practicum 2021/22, Lesson 20

Marko Tkalčič

Univerza na Primorskem

Table of Contents

Recommender Systems

- Goal: recommend best items (Netflix, Amazon . . .)
- What are best items?
  - Highest rated (highly questionable!!!)
- New goal:
  - Predict unseen (unknown) items ratings
  - Recommend highest-rated (obvious)

Recommender Systems

- Goal: recommend best items (Netflix, Amazon . . .)
- What are best items?
  - Highest rated (highly questionable!!!)
- New goal:
  - Predict unseen (unknown) items ratings
  - Recommend highest-rated (obvious)

- User-item Matrix

	Matrix	Godfather	Titanic	Avatar	Fight Club	Fantozzi
John	5		3			1
Paul		4	3		2	
Mary	4	1		3		
Anna	2				2	2

Default Data Structure

user_id	item_id	rating
1	1	5
1	3	3
2	2	3
3	4	1

- Often we have additional columns
  - user characteristics (age, gender, personality . . .)
  - item characteristics (genre, year, author . . .)
  - context/interaction (weather, social, location . . .)

Exercise

- Load the CoMoDa dataset
- Inspect the dataset
- How would you proceed with predicting missing ratings?

Exercise

- Load the CoMoDa dataset
- Inspect the dataset
- How would you proceed with predicting missing ratings?

```
import pandas as pd
df_comoda = pd.read_csv("LDOS-CoMoDa.csv", sep=";")
df_comoda.head()
```

	userID	itemID	rating	age	sex	city	country	time	daytype	season	...	movieCountry	movieLanguage	movieYear	genre1	genre2	genre3	ac
0	23	14	5	33	1	20	2	3	2	2	...	36	9	2007	7	-1	-1	1
1	21	5	3	28	1	10	3	2	2	2	...	37	9	1998	7	6	10	1
2	21	6	4	28	1	10	3	4	2	2	...	37	9	2008	7	10	18	1
3	22	13	4	28	1	20	2	3	2	3	...	37	9	2010	1	14	19	1
4	21	7	3	28	1	10	3	4	2	2	...	37	9	2003	3	10	18	

5 rows × 30 columns

Exercise

- Predict the ratings using some regressor
- 80/20 split
- MSE

Exercise

- Predict the ratings using some regressor
- 80/20 split
- MSE

```
## Load dataset
import pandas as pd
df_comoda = pd.read_csv("LDOS-CoMoDa.csv", sep=";")
df_X = df_comoda
df_y = df_comoda["rating"]
del df_X["rating"]
X = df_X.values
y = df_y.values
# test/train splitting
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = \
    train_test_split(X, y, test_size=0.20, random_state=1)
#use RandomForestRegressor to predict ratings
from sklearn.ensemble import RandomForestRegressor
clf = RandomForestRegressor().fit(X_train, y_train)
y_pred = clf.predict(X_test)
mean_squared_error(y_test, y_pred)
```

```
0.7793109321231162
```

Is this a good result?

Exercise

- Make the predicted ratings equal to the average rating

Exercise

- Make the predicted ratings equal to the average rating

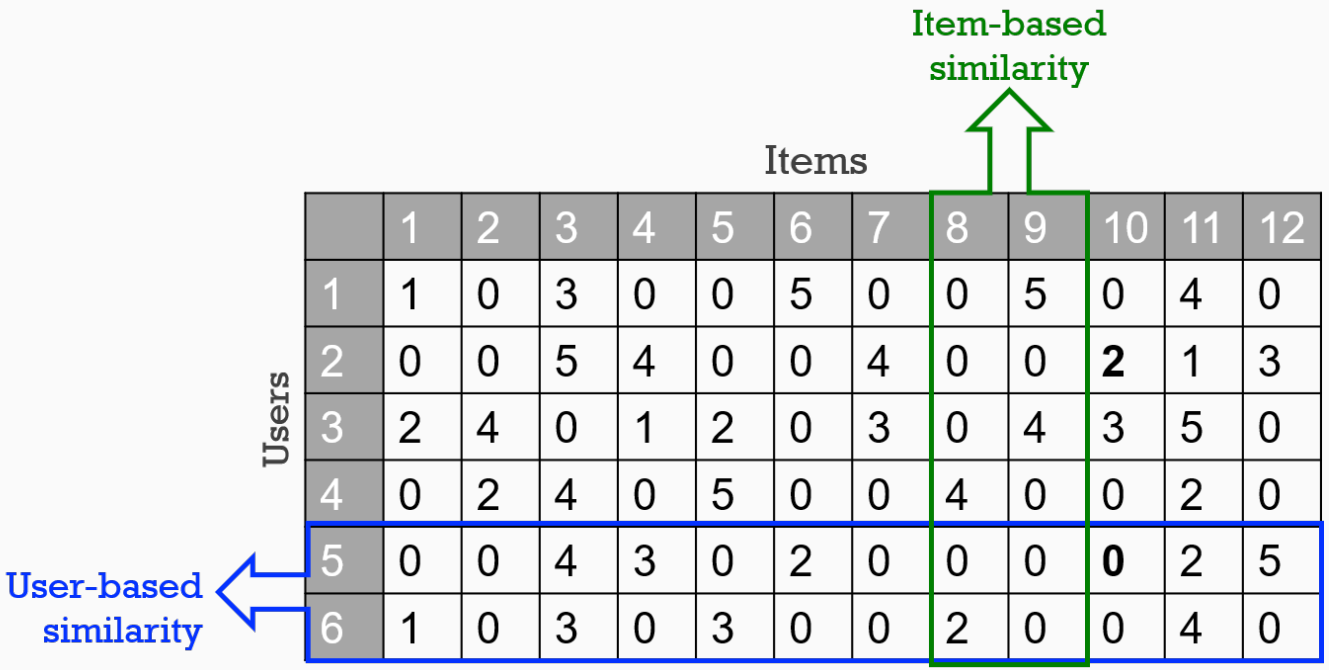
```
# predicted rating as average of all ratings
import numpy as np

a = np.mean(y_train)
y_pred = np.full(y_test.shape,a)
mean_squared_error(y_test, y_pred)
```

```
1.1553692755769438
```

Recommender Systems Taxonomy

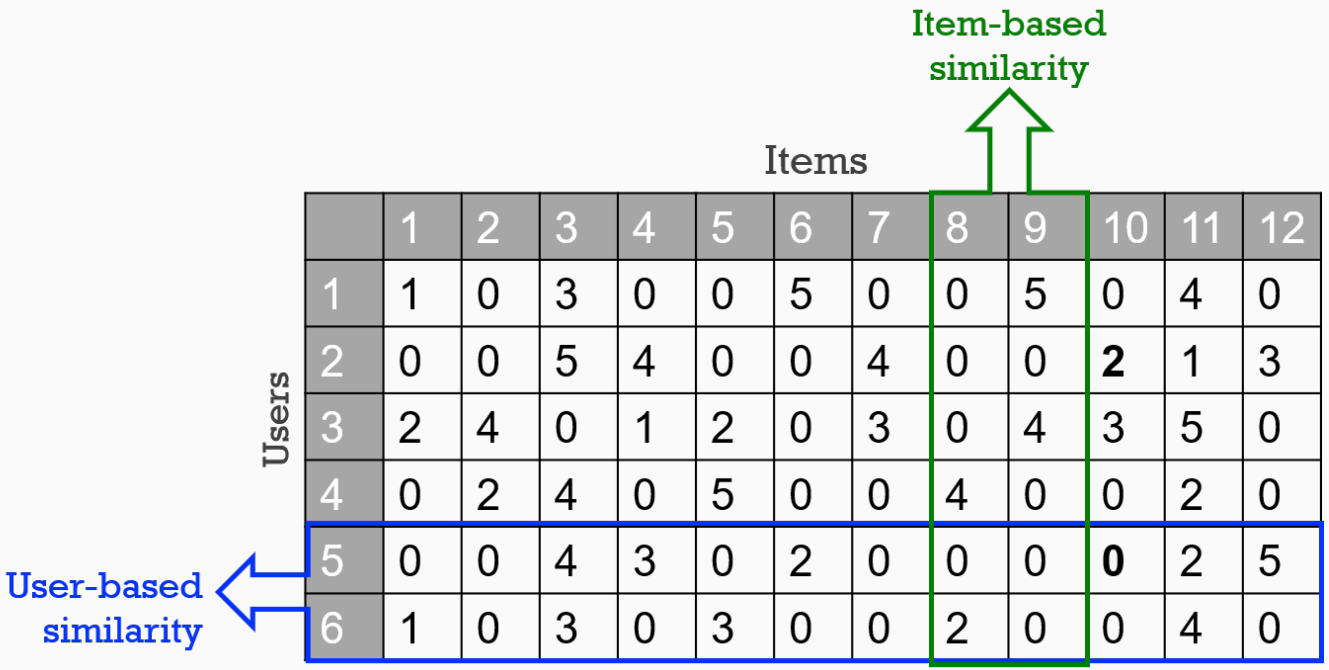
- Content-based
  - User profile
  - Item profile
- Collaborative Filtering
  - Memory-based (neighborhood-based)
  - Model-based (e.g. Matrix Factorization)
- Hybrid

k-Nearest Neighbors



	1	2	3	4	5	6	7	8	9	10	11	12
1	1	0	3	0	0	5	0	0	5	0	4	0
2	0	0	5	4	0	0	4	0	0	**2**	1	3
3	2	4	0	1	2	0	3	0	4	3	5	0
4	0	2	4	0	5	0	0	4	0	0	2	0
5	0	0	4	3	0	2	0	0	0	**0**	2	5
6	1	0	3	0	3	0	0	2	0	0	4	0

k-Nearest Neighbors



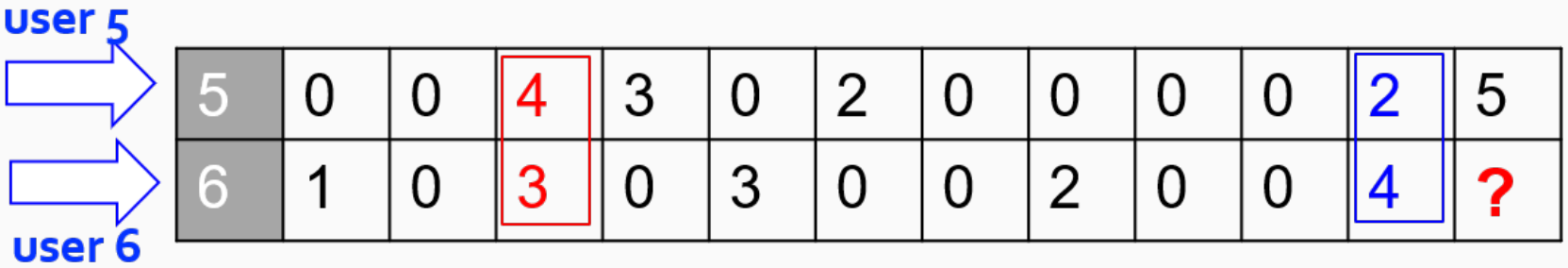
	1	2	3	4	5	6	7	8	9	10	11	12
1	1	0	3	0	0	5	0	0	5	0	4	0
2	0	0	5	4	0	0	4	0	0	**2**	1	3
3	2	4	0	1	2	0	3	0	4	3	5	0
4	0	2	4	0	5	0	0	4	0	0	2	0
5	0	0	4	3	0	2	0	0	0	**0**	2	5
6	1	0	3	0	3	0	0	2	0	0	4	0

- Distance metrics
- Similarity metrics
  - Cosine similarity
  - Pearson correlation

```
from scipy import spatial
v1 = [0,0,1]
v2 = [3,2,1]
s = spatial.distance.cosine(v1,v2)
```

Cosine similarity

- Compute only on items that both users have rated



k-Nearest Neighbors

- Rating Prediction for user u and item j

$$\hat{r}_{uj} = r_u + \frac{1}{\sum |w_{uv}|} \sum w_{uv}(r_{vj} - r_v)$$

- predicted rating $\hat{r}_{uj}$
- average user rating r_u
- user similarity w_{uv}
- rating r_{vj}
- average user rating r_v

Exercise

- Write a function that calculates the cosine similarity between two vectors of users

Exercise

- Write a function that calculates the cosine similarity between two vectors of users

```
# cosine similarity
# only on items both have rated
from scipy import spatial

def cos_sim(u1,u2):
    v1 = []
    v2 = []
    for i in range(len(u1)):
        if (u1[i] != 0) and (u2[i] != 0):
            v1.append(u1[i])
            v2.append(u2[i])
    if (len(v1) == 0):
        return 0
    else:
        return 1- spatial.distance.cosine(v1,v2)

u1 = [0,0,0,1,0,0,5,0,0,4,0,0,3,0,0,2]
u2 = [0,2,0,3,1,0,0,2,0,5,0,0,0,5,0,0]

#u1 = [0,1]
#u2 = [1,0]

cos_sim(u1,u2)
```

Exercise

- From the comoda dataset create a user-item matrix

Exercise

- From the comoda dataset create a user-item matrix

```
# create user-item matrix
import pandas as pd
import numpy as np

df_comoda = pd.read_csv("LDOS-CoMoDa.csv", sep=";")
users = np.sort(df_comoda["userID"].unique())
items = np.sort(df_comoda["itemID"].unique())
df_users_items_ratings = pd.DataFrame(index = users, columns = items )
df_users_items_ratings = df_users_items_ratings.fillna(0)

## test/train splitting
cutoff = round(0.8 * len(df_comoda))
df_comoda_train = df_comoda[:cutoff]
df_comoda_test = df_comoda[cutoff:]

for index, row in df_comoda_train.iterrows():
    df_users_items_ratings.loc[row["userID"],row["itemID"]] = row["rating"]

np_users_items_ratings = df_users_items_ratings.values
```

Exercise

- Calculate a matrix with user-user similarities

Exercise

- Calculate a matrix with user-user similarities

```
# create user-user similarities
df_users_users =  pd.DataFrame(index = users, columns = users )
df_users_users = df_users_users.fillna(0)
for u1 in users:
    for u2 in users:
        u1_vector = np_users_items_ratings[:,u1]
        u2_vector = np_users_items_ratings[:,u2]
        d = cos_sim(u1_vector,u2_vector)
        df_users_users.loc[u1,u2] = d

df_users_users.describe()
```

Table of Contents

References

Part of the material has been taken from the following sources. The usage of the referenced copyrighted work is in line with fair use since it is for nonprofit educational purposes.

-