22 - Recommender Systems - 2

Data Science Practicum 2021/22, Lesson 22

Marko Tkalčič

Univerza na Primorskem

Table of Contents

Surprise

- Surprise is a library for recommender systems
  - datasets
  - algorithms
  - evaluation

Surprise

- Surprise is a library for recommender systems
  - datasets
  - algorithms
  - evaluation

```
from surprise import SVD
from surprise import Dataset
from surprise.model_selection import cross_validate

# Load the movielens-100k dataset (download it if needed).
data = Dataset.load_builtin('ml-100k')

# Use the famous SVD algorithm.
algo = SVD()

# Run 5-fold cross-validation and print results.
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9309  0.9363  0.9317  0.9435  0.9363  0.9357  0.0045
MAE (testset)     0.7346  0.7357  0.7351  0.7442  0.7376  0.7374  0.0035
Fit time          4.05    4.14    4.08    4.11    4.09    4.09    0.03
Test time         0.14    0.11    0.13    0.13    0.14    0.13    0.01
```

Datasets

- Built-in
  - Movielens 100k
  - Movielens 1M
  - Jester
- Load from file (three columns: userID, itemID, rating)
- Load from dataframe (three columns: userID, itemID, rating)

Exercise

- Load the internal ml-100k dataset
- Run the NormalPredictor algorithm
- You will be asked to download the dataset
- cross validate with RMSE

Exercise

- Load the internal ml-100k dataset
- Run the NormalPredictor algorithm
- You will be asked to download the dataset
- cross validate with RMSE

```
from surprise import NormalPredictor
from surprise import Dataset
from surprise.model_selection import cross_validate

# Load the movielens-100k dataset (download it if needed).
data = Dataset.load_builtin('ml-100k')

# Use the famous SVD algorithm.
algo = NormalPredictor()

# Run 5-fold cross-validation and print results.
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
Evaluating RMSE of algorithm NormalPredictor on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.5136  1.5168  1.5220  1.5296  1.5159  1.5196  0.0057
Fit time          0.10    0.13    0.14    0.13    0.13    0.13    0.01
Test time         0.42    0.11    0.40    0.11    0.41    0.29    0.15
```

- Normal Predictor: random rating based on the distribution of the training set

Reader

- The Reader object is used to parse external datasets
  - name (string, optional) – If specified, a Reader for one of the built-in datasets is returned and any other parameter is ignored. Accepted values are 'ml-100k', 'ml-1m', and 'jester'. Default is None.
  - line_format (string) – The fields names, in the order at which they are encountered on a line. Please note that line_format is always space-separated (use the sep parameter). Default is 'user item rating'.
  - sep (char) – the separator between fields. Example : ';'.
  - rating_scale (tuple, optional) – The rating scale used for every rating. Default is (1, 5).
  - skip_lines (int, optional) – Number of lines to skip at the beginning of the file. Default is 0.

```
...
r = Reader()
data = Dataset.load_from_df(df, reader=r)
```

Exercise

- Load the comoda dataset
- Adjust the rating_scale parameter of the reader
- Run SVD, cross validate RMSE

Exercise

- Load the comoda dataset
- Adjust the rating_scale parameter of the reader
- Run SVD, cross validate RMSE

```
import pandas as pd
from surprise import SVD
from surprise import Dataset
from surprise import Reader
from surprise.model_selection import cross_validate
df_comoda = pd.read_csv("LDOS-CoMoDa.csv", sep=";")
print(df_comoda.head())
df_comoda_uid = df_comoda[["userID","itemID","rating"]]
print(df_comoda_uid.describe())
reader = Reader(rating_scale=(1,5))
data = Dataset.load_from_df(df_comoda_uid, reader=reader)
algo = SVD()
cross_validate(algo, data, measures=['RMSE'], cv=5, verbose=True)
```

```
   userID  itemID  rating  age  sex  city  country  time  daytype  season  \
0      23      14       5   33    1    20        2     3        2       2
1      21       5       3   28    1    10        3     2        2       2

            userID       itemID       rating
count  2296.000000  2296.000000  2296.000000
min      15.000000     1.000000     1.000000
...
max     268.000000  4381.000000     5.000000

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.0212  1.0316  1.0758  0.9575  0.9848  1.0142  0.0405
Fit time          0.09    0.09    0.09    0.09    0.09    0.09    0.00
Test time         0.00    0.00    0.00    0.00    0.00    0.00    0.00
```

Algorithm: BaselineOnly

- BaselineOnly

$$\hat{r}_{ui} = \mu + b_u + b_i$$

Algorithm: BaselineOnly

- BaselineOnly

$$\hat{r}_{ui} = \mu + b_u + b_i$$

- Exercise: apply the BAselineOnly algorithm to Comoda

Algorithm: BaselineOnly

- BaselineOnly

$$\hat{r}_{ui} = \mu + b_u + b_i$$

- Exercise: apply the BAselineOnly algorithm to Comoda

```
df_comoda = pd.read_csv("LDOS-CoMoDa.csv", sep=";")
df_comoda_uid = df_comoda[["userID","itemID","rating"]]
reader = Reader(rating_scale=(1,5))
data = Dataset.load_from_df(df_comoda_uid, reader=reader)

algo = BaselineOnly()

cross_validate(algo, data, measures=['RMSE'], cv=5, verbose=True)
```

```
                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9650  1.0153  1.0275  1.0157  1.0434  1.0134  0.0263
Fit time          0.00    0.00    0.00    0.00    0.00    0.00    0.00
Test time         0.00    0.00    0.00    0.00    0.00    0.00    0.00
```

Algorithm: kNN

- kNNBasic, kNNWithMeans, kNNWithZScore, kNNBaseline
  - k . . . max num of neighbours
  - min_k . . . min num of neighbours
  - sim_options . . . similarity options

Algorithm: kNN

- kNNBasic, kNNWithMeans, kNNWithZScore, kNNBaseline
  - k . . . max num of neighbours
  - min_k . . . min num of neighbours
  - sim_options . . . similarity options

```
sim_options = {'name': 'cosine',
               'user_based': False  # compute  similarities between items
               }
algo = KNNBasic(sim_options=sim_options)
```

- name : cosine, pearson, MSD, pearson_baseline
- user_based: True/False
- min_support: min num of common items/users

Exercise

- Compare item-based vs. user-based similarity RMSE on Comoda

Exercise

- Compare item-based vs. user-based similarity RMSE on Comoda

```
import pandas as pd
from surprise import KNNBasic
from surprise import Dataset
from surprise import Reader
from surprise.model_selection import cross_validate

df_comoda = pd.read_csv("LDOS-CoMoDa.csv", sep=";")
df_comoda_uid = df_comoda[["userID","itemID","rating"]]
reader = Reader(rating_scale=(1,5))
data = Dataset.load_from_df(df_comoda_uid, reader=reader)

# item-based
sim_options = { 'user_based': False  # compute  similarities between items
              }
algo = KNNBasic(sim_options=sim_options)
cross_validate(algo, data, measures=['RMSE'], cv=5, verbose=True)

# user-based
sim_options = { 'user_based': True  # compute  similarities between items
              }
algo = KNNBasic(sim_options=sim_options)
cross_validate(algo, data, measures=['RMSE'], cv=5, verbose=True)
```

```
                 Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)   1.0375  1.0116  1.0623  1.0605  1.0725  1.0489  0.0219

                 Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)   1.1487  1.0922  1.1457  1.1734  1.0945  1.1309  0.0321
```

Exercise

- Use the comoda dataset
- Repeat:
  - remove the top N items (in terms of number of ratings)
  - N from 1 to 80
- Plot how RMSE changes
  - user-based similarity
  - item-based similarity

Exercise

- Use the comoda dataset
- Repeat:
  - remove the top N items (in terms of number of ratings)
  - N from 1 to 80
- Plot how RMSE changes
  - user-based similarity
  - item-based similarity

```
def remove_items(df,n):
    for i in range(n):
        m = df.iloc[:,1].mode()
        df = df[df.iloc[:,1] != m[0]]
    return df
```

Exercise

```
df_comoda = pd.read_csv("LDOS-CoMoDa.csv", sep=";")
df_comoda_uid = df_comoda[["userID","itemID","rating"]]
reader = Reader(rating_scale=(1,5))

ns = range(80)
rmse_i = []
rmse_u = []
for n in ns:
    #print("N = ",n)
    df_comoda_short = remove_items(df_comoda_uid,n)
    data = Dataset.load_from_df(df_comoda_short, reader=reader)

    # item-based
    sim_options = { 'user_based': False  }
    algo = KNNBasic(sim_options=sim_options)
    cv_i = cross_validate(algo, data, measures=['RMSE'], cv=5)
    #print(" CV_i: ",np.mean(cv_i["test_rmse"]))
    rmse_i.append(np.mean(cv_i["test_rmse"]))

    # user-based
    sim_options = { 'user_based': True    }
    algo = KNNBasic(sim_options=sim_options)
    cv_u = cross_validate(algo, data, measures=['RMSE'], cv=5)
    rmse_u.append(np.mean(cv_u["test_rmse"]))

import matplotlib.pyplot as plt
plt.plot(ns,rmse_u, marker="o", label="user-based RMSE")
plt.plot(ns,rmse_i, marker="x", label = "item-based RMSE")
plt.xlabel("Number of users removed")
plt.ylabel("RMSE")
plt.legend()
plt.grid();
plt.show()
```

Exercise

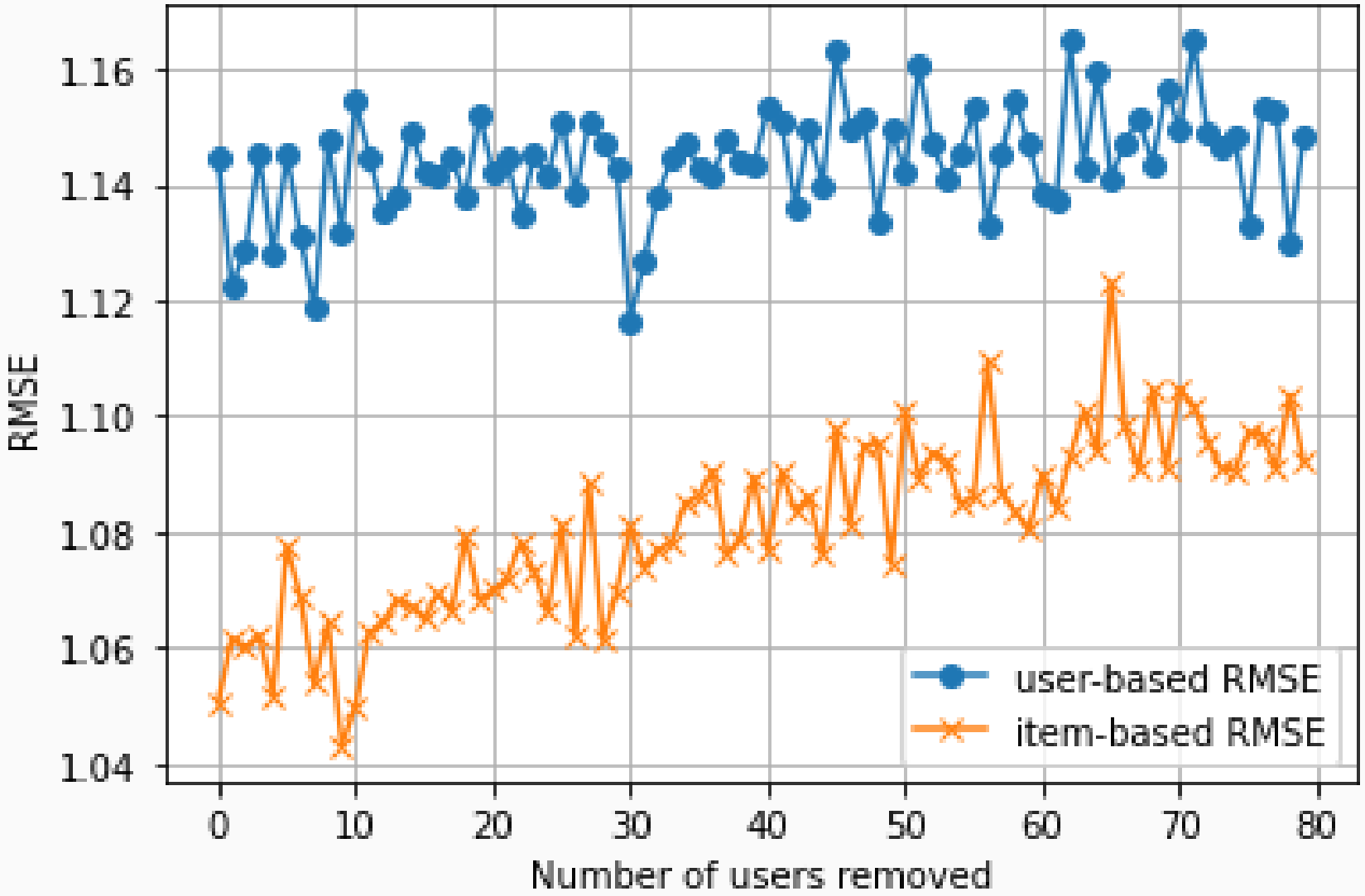
RMSE
1.16
1.14
1.12
1.10
1.08
1.06
1.04
0
10
20
30
40
50
60
70
80
Number of users removed
user-based RMSE
item-based RMSE

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.

-