Model-Based CF by using singular value decomposition (SVD)

Data: MovieLens dataset It contains 100k movie ratings from 943 users and a selection of 1682 movies.

You can download the dataset here (http://files.grouplens.org/datasets/movielens/ml-100k.zip).

u.data file: contains the full dataset. description of the dataset here (http://files.grouplens.org/datasets/movielens/ml-100k-README.txt).

u.data -- The full u data set, 100000 ratings by 943 users on 1682 items. Each user has rated at least 20 movies. Users and items are numbered consecutively from 1. The data is randomly ordered.

```
 Note: This is a **tab** separated list of
**user id | item id | rating | timestamp**.
```

Import libs

In [1]:
```
import numpy as np
import pandas as pd
```

Load Data

In [2]:
```
column_names = ['user_id', 'item_id', 'rating', 'timestamp']
data_org = pd.read_csv('data/u.data', sep='\t', names=column_names)
```

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

Out[3]:

	user_id	item_id	rating	timestamp
0	0	50	5	881250949
1	0	172	5	881250949
2	0	133	1	881250949
3	196	242	3	881250949
4	186	302	3	891717742

We have item_id, which is not the movie name.

Use the Movie_ID_Titles csv file to grab the movie names and merge it with this dataframe:

In [4]:
```
movie_titles = pd.read_csv("data/Movie_Id_Titles")
movie_titles.head()
```

Out[4]:

	item_id	title
0	1	Toy Story (1995)
1	2	GoldenEye (1995)
2	3	Four Rooms (1995)
3	4	Get Shorty (1995)
4	5	Copycat (1995)

Both data_org anf movie_titles have 'item_id' in common, merge by that.

In [5]:
```
data_org.shape
```

Out[5]: (100003, 4)

In [6]:
```
movie_titles.shape
```

Out[6]: (1682, 2)

In [7]:
```
data= pd.merge(data_org,movie_titles,on='item_id')
data.head()
```

Out[7]:

	user_id	item_id	rating	timestamp	title
0	0	50	5	881250949	Star Wars (1977)
1	290	50	5	880473582	Star Wars (1977)
2	79	50	4	891271545	Star Wars (1977)
3	2	50	5	888552084	Star Wars (1977)
4	8	50	5	879362124	Star Wars (1977)

In [8]:
```
data.shape
```

Out[8]: (100003, 5)

Tarin/Test Split

In [9]:
```
from sklearn.model_selection import train_test_split
train_data, test_data = train_test_split(data, test_size=0.25)
```

In [10]:
```
print("df dimension={}".format(data.shape))
print("train_data dimension={}".format(train_data.shape))
print("test_data dimension={}".format(test_data.shape))
```

```
df dimension=(100003, 5)
train_data dimension=(75002, 5)
test_data dimension=(25001, 5)
```

In [11]:
```
train_data.head(3)
```

Out[11]:

	user_id	item_id	rating	timestamp	title
40785	178	271	4	882823395	Starship Troopers (1997)
71582	761	148	5	876189829	Ghost and the Darkness, The (1996)
95511	934	972	3	891225716	Passion Fish (1992)

Create two user_id - item_id matrices, one for training and another for testing

Each element is the rating

In [12]:
```
n_users = data.user_id.nunique()
n_items = data.item_id.nunique()

print('Num. of Users: '+ str(n_users))
print('Num of Movies: '+str(n_items))
```

```
Num. of Users: 944
Num of Movies: 1682
```

In [13]:
```
train_data_matrix = np.zeros((n_users, n_items))
for line in train_data.itertuples():
    train_data_matrix[line[1]-1, line[2]-1] = line[3]

test_data_matrix = np.zeros((n_users, n_items))
for line in test_data.itertuples():
    test_data_matrix[line[1]-1, line[2]-1] = line[3]
```

In [14]:
```
print(train_data_matrix.shape)
train_data_matrix
```

```
(944, 1682)
```

Out[14]:
```
array([[5., 0., 4., ..., 0., 0., 0.],
       [4., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 5., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [15]:
```
# check sarsity of matrix
print(len(data_org))
print(n_users*n_items)
print( len(data_org) / (n_users*n_items))
sparsity=round(1.0-len(data_org)/float(n_users*n_items),3)
print('The sparsity level of MovieLens100K is ' +  str(sparsity*100) + '%')
```

```
100003
1587808
0.06298179628771237
The sparsity level of MovieLens100K is 93.7%
```

Build the model

In [16]:
```
import scipy.sparse as sp
from scipy.sparse.linalg import svds

#get SVD components from train matrix. Choose k.
u, s, vt = svds(train_data_matrix, k = 20)
```

In [17]:
```
train_data_matrix.shape
```

Out[17]: (944, 1682)

In [18]:
```
u.shape
```

Out[18]: (944, 20)

In [19]:
```
s.shape
```

Out[19]: (20,)

In [20]:
```
vt.shape
```

Out[20]: (20, 1682)

Prediction

In [21]:
```
s_diag_matrix=np.diag(s)
X_pred = np.dot(np.dot(u, s_diag_matrix), vt)
```

In [22]:
```
X_pred.shape
```

Out[22]: (944, 1682)

In [23]:
```
train_data_matrix.shape
```

Out[23]: (944, 1682)

In [24]:
```
test_data_matrix.shape
```

Out[24]:
```
(944, 1682)
```

Evaluation

Only consider predicted ratings that are in the test dataset (the non zero values in test data set)

In [25]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt
def rmse(prediction, ground_truth):
    prediction = prediction[ground_truth.nonzero()].flatten()
    ground_truth = ground_truth[ground_truth.nonzero()].flatten()
    return sqrt(mean_squared_error(prediction, ground_truth))
```

In [26]:
```
print('User-based CF MSE: ' + str(rmse(X_pred, test_data_matrix)))
```

```
User-based CF MSE: 2.711322923469662
```