matrix_factorization_svd

November 6, 2017

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

In [2]:
```
training_files = ['ml-100k/u1.base', 'ml-100k/u2.base', 'ml-100k/u3.base', 'ml-100k/u4
validation_file = 'ml-100k/u5.base'
out_file = 'mf-k.csv'
k=5
```

In [3]:
```
frames = []
for training_file in training_files:
    frames.append(pd.read_csv(training_file, sep='\t',  header=0,
                          names=['userId', 'movieId', 'rating','timestamp'], engine='pyth
ratings = pd.concat(frames)
ratings.head()
```

Out[3]:

	userId	movieId	rating	timestamp
0	1	2	3	876893171
1	1	3	4	878542960
2	1	4	3	876893119
3	1	5	3	889751712
4	1	7	4	875071561

In [4]:
```
# R_df = ratings.pivot(index = 'userId', columns ='movieId', values = 'rating').fillna
R_df = pd.pivot_table(ratings,values='rating',index='userId',columns='movieId').fillna
R_df.head()
```

Out[4]:

movieId	1	2	3	4	5	6	7	8	9	10	...	\
userId											...	
1	0.0	3.0	4.0	3.0	3.0	5.0	4.0	1.0	5.0	3.0	...	
2	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	...	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
5	4.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	

movieId	1673	1674	1675	1676	1677	1678	1679	1680	1681	1682
userId										
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
3         0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
4         0.0   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   0.0   0.0   0.0   0.0   0.0

[5 rows x 1682 columns]
```

In [5]:
```
# normalized matrix
R = R_df.as_matrix()
user_ratings_mean = np.mean(R, axis = 1)
R_demeaned = R - user_ratings_mean.reshape(-1, 1)
```

In [6]:
```
# SVD decomposition
from scipy.sparse.linalg import svds
U, sigma, Vt = svds(R_demeaned, k = 50)
sigma = np.diag(sigma)
```

In [7]:
```
# Prediction
all_user_predicted_ratings = np.dot(np.dot(U, sigma), Vt) + user_ratings_mean.reshape(
preds_df = pd.DataFrame(all_user_predicted_ratings, columns = R_df.columns)
```

In [8]:
```
# top k recommendations for users
top_k_list = {}
for user in preds_df.index.unique():
    rated_movie = ratings[ratings['userId'] == (user)]['movieId']
    pred_rating = preds_df[[x for x in preds_df.columns.values if x not in rated_movie
    top_k_list[user] =pred_rating.index.values.tolist()[:k]
```

In [9]:
```
top_k_df = pd.DataFrame.from_dict(top_k_list, orient='index')
top_k_df.head()
```

Out[9]:

	0	1	2	3	4
0	100	176	89	12	135
1	286	302	313	275	285
2	268	340	333	328	327
3	50	748	301	269	289
4	168	181	228	222	208

In [10]:
```
top_k_df.to_csv(out_file)
```

In [11]:
```
# Validation, not sure how to do yet
validation_rating = pd.read_csv(validation_file, sep='\t',  header=0,
                        names=['userId', 'movieId', 'rating','timestamp'], engine='pyth
validation_rating.head()
```

Out[11]:

	userId	movieId	rating	timestamp
0	1	2	3	876893171
1	1	4	3	876893119
2	1	5	3	889751712
3	1	6	5	887431973
4	1	7	4	875071561

In [12]:
```
def predict(row):
    return preds_df.loc[row['userId'] - 1, row['movieId']]
validation_rating['pred rating'] = validation_rating.apply (lambda row:  predict(row)
validation_rating.head()
```

Out[12]:

	userId	movieId	rating	timestamp	pred rating
0	1	2	3	876893171	2.911756
1	1	4	3	876893119	3.180375
2	1	5	3	889751712	1.615407
3	1	6	5	887431973	1.591248
4	1	7	4	875071561	3.495347

In [13]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(validation_rating['rating'], validation_rating['pred rating'])
```

Out[13]: 3.4196413821591585