Memory-Based CF by computing Cosine Similarity

And SVD Model Based

This implementation is based on codes and ideas from attached references [1].

The algorithm is implemented in Python3.6 using `numpy`, `pandas`, and `sklearn` libraries [1]

First, we loaded the data and then split the data into training set and test set to use it in an offline evaluation later. Then we created the User-Item Matrix. (unique users id are the rows and unique items id are the columns)

```
print('Memory-Based CF by computing cosine similarity')
header = ['user_id', 'item_id', 'rating', 'timestamp']
df = pd.read_csv('ratings.dat', sep='::', names=header,engine='python')
#print (df.item_id)
n_users = df.user_id.unique().shape[0]
#n_items = df.item_id.unique().shape[0]
n_items = max(df.item_id)

print ('Number of users = ' + str(n_users) + ' | Number of movies = ' + str(n_items))
train_data, test_data = cv.train_test_split(df, test_size=0.1)

#Create two user-item matrices, one for training and another for testing
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]
#print (train_data.itertuples())
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]
```

Similarity Comparison

1. Item-Item Collaborative : works on users who like same item
2. User-Item Collaborative : works on all items liked by pair similar users

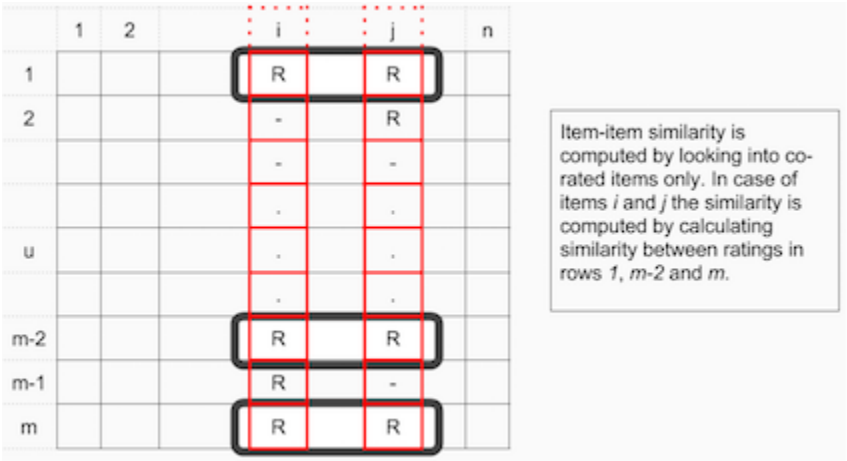

Figure 1: Item-item similarity [1]

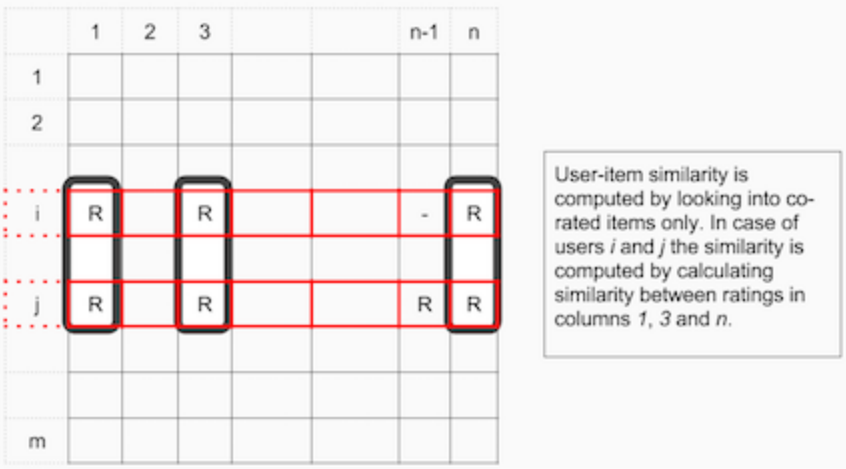

Figure 2: User-item Similarity [1]

Then calculate the cosine similarity for both types

$$sim(u,v)^{COS} = \frac{\vec{r}_u \cdot \vec{r}_v}{\|\vec{r}_u\| \cdot \|\vec{r}_v\|}$$

- In Item-Item Similarity r vector will be the item vector
- In User-item Similarity r vector will be the user vector

```
29
30 user_similarity = pairwise_distances(train_data_matrix, metric='cosine')
31 item_similarity = pairwise_distances(train_data_matrix.T, metric='cosine')
```

Then we predict the items ratings

```
33 def predict(ratings, similarity, type='user'):
34     if type == 'user':
35         mean_user_rating = ratings.mean(axis=1)
36         #You use np.newaxis so that mean_user_rating has same format as ratings
37         ratings_diff = (ratings - mean_user_rating[:, np.newaxis])
38         pred = mean_user_rating[:, np.newaxis] + similarity.dot(ratings_diff) / np.array([np.abs(similarity).sum(axis=1)]).T
39     elif type == 'item':
40         pred = ratings.dot(similarity) / np.array([np.abs(similarity).sum(axis=1)])
41     return pred
42
43 item_prediction = predict(train_data_matrix, item_similarity, type='item')
44 user_prediction = predict(train_data_matrix, user_similarity, type='user')
45
```

Calculation of root mean square error between the Predicted matrix and the test matrix

(Taking in consideration the comparisons are with the adjacent users and items

The last two lines to calculate the sparsity of the data set, which is the cause of high RMSE in the data.

```
46 def rmse(prediction, ground_truth):
47     prediction = prediction[ground_truth.nonzero()].flatten()
48     ground_truth = ground_truth[ground_truth.nonzero()].flatten()
49     return sqrt(mean_squared_error(prediction, ground_truth))
50 print ('User-based CF RMSE: ' + str(rmse(user_prediction, test_data_matrix)))
51 print ('Item-based CF RMSE: ' + str(rmse(item_prediction, test_data_matrix)))
52 #print(item_prediction)
53
54 sparsity=round(1.0-len(df)/float(n_users*df.item_id.unique().shape[0]),3)
55 print ('The sparsity level of MovieLens100K is ' +  str(sparsity*100) + '%')
```

Formula for user-based CF [1]

Prediction = Mean of user ratings + (dot product of User Similarity and (ratings – its mean)) /sum of similarities vectors magnitudes.

$$\hat{x} = \bar{x} + \frac{\sum_u sim^{cosine}(u,v)(x-\bar{x})}{\sum_u |sim^{cosine}(u,v)|}$$

The subtraction of the mean is because some users are always giving high or low ratings only.

Formula for user-based CF [1]

$$\hat{x} = \frac{\sum_u sim^{cosine}(u,v)(x)}{\sum_u |sim^{cosine}(u,v)|}$$

Results

```
Memory-Based CF by computing cosine similarity
Number of users = 6040 | Number of movies = 3952
User-based CF RMSE: 3.167669057797835
Item-based CF RMSE: 3.500890143940474

The sparsity level of MovieLens1M is 95.5%
```

The cold-start problem and sparsity problems are the main causes in error for memory based CF problems. The cold start problem can be solved Content based recommenders which should collect data on the users information and movie's genres. Model-based CF such as SVD and ALS can solve sparsity. To overcome different recommender system challenges hybrid systems should be used.

Singular value decomposition [1, 2]

SVD is Matrix Factorization method becomes famous after Netflix Prize [2] because the winning entry was depends on SVD and SVD++ model based Collaborative filters.

Prediction Equation using SVD factorization [1]

$$\hat{X} = U.S.V^T$$

U is orthogonal matrix represents the feature vectors corresponding to the users in the hidden feature space.

S is diagonal matrix which elements are the singular values of X.

V is orthogonal matrix represents the feature vectors corresponding to the items in the hidden feature space.

It is iterative method with k iteration and its implementation with scipy.sparse

```
59 from scipy.sparse.linalg import svds
60
61 #get SVD components from train matrix. Choose k.
62 u, s, vt = svds(train_data_matrix, k = 20)
63 s_diag_matrix=np.diag(s)
64 X_pred = np.dot(np.dot(u, s_diag_matrix), vt)
65 print ('SVD model User-based CF RMSE: ' + str(rmse(X_pred, test_data_matrix)))
```

```
Model-Based CF using Single Value Distance
SVD model User-based CF RMSE: 2.596665106063839
```

References

[1] Johannsdottir, A. (n.d.). *tutorials*. Retrieved from cambridgespark.com: https://cambridgespark.com/content/tutorials/implementing-your-own-recommender-systems-in-Python/index.html

[2] Gower, S. (2014). *Netflix Prize and SVD.* Math. http://buzzard.ups.edu. Retrieved from http://buzzard.ups.edu/courses/2014spring/420projects/math420-UPS-spring-2014-gower-netflix-SVD.pdf