PRACTICAL DATA SCIENCE WITH PYTHON COSC 2670/2738 ASSIGNMENT 3

Karntawan Udomluksopin
S4029434

I certify that this is all my own original work. If I took any parts from elsewhere, then they were non-essential parts of the assignment, and they are clearly attributed in our submission. I will show I agree to this honor code by typing “Yes”: YES

Recommendation system with Adjusted Euclidean distance (AED)

Purposes of this recommendations report

- To validate the performance of new similar measure for k nearest neighbour Collaborative Filtering method based which called adjusted Euclidean distance (AED) by using Euclidean Distance.

Adjusted Euclidean Distance

Enhancement of Recommendation systems

How the Adjusted Euclidean Distance works

Euclidean distance measures the 'straight line' distance between points in a space. With the formula

Why the Adjusted Euclidean Distance works

Why adjust these distances? Because in a multidimensional space, not all dimensions are equally populated with data. By normalizing Euclidean distances, AED takes into account the length of vectors and their dimensional count, ensuring a fair and meaningful similarity measure

formula of AED

$$sim(u,v) = 1 - \frac{dist(\vec{u},\vec{v})}{dist_{\max}} = 1 - \frac{\sqrt{\sum_{s \in S_{uv}} (r_{u,s} - r_{v,s})^2}}{\sqrt{\sum_{i=1}^{m} (V_{\max} - V_{\min})^2}} \quad ,(10)$$

Dataset

MovieLens

In this report uses MovieLens data sets which were collected by the GroupLens Research Project at the University of Minnesota from September 19th, 1997 through April 22nd, 1998.
This data set consists of: 100,000 ratings (1-5) from 943 users on 1682 movies.

Part 1 : Set up and Install

This report imported assignment3_framework.ipynb from Canvas RMIT.

Install and load necesary packages

In [1]:
```
# Please don't change this cell

import pandas as pd
import numpy as np

import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
# Please don't change this cell
df = pd.read_csv('ml-100k/u.data', names=['user_id', 'item_id', 'rating', 'timestamp'],

df.head()
```

Out[2]:

	user_id	item_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

Install necessary packages such as pandas and numpy.

Then load Movies Lens by read_csv ‘ml-100k/u.data’

sperated list by tabs
user id | item id | rating | timestamp.

Part 2 : Training dataset

This report imported assignment3_framework.ipynb from Canvas RMIT.

Split dataset

Random Train and Test Split

```
# please do not change this cell

from sklearn.model_selection import train_test_split

n_users = df.user_id.unique().shape[0]
n_items = df.item_id.unique().shape[0]
print(str(n_users) + ' users')
print(str(n_items) + ' items')

train_df, test_df = train_test_split(df, test_size=0.2, random_state = 10)
train_df, test_df

# Training Dataset
train_ds = np.zeros((n_users, n_items))
item_popularity = np.zeros(n_items)
for row in train_df.itertuples():
    train_ds[row[1]-1, row[2]-1] = row[3]
    item_popularity[row[2]-1] =  item_popularity[row[2]-1] + 1
#train_ds = pd.DataFrame(train_ds)
```

Use train_test_split to split training set into 80% and test set 20%

Training Dataset will be considered as two dimension matrix and compare each user with other users by using itertuples()

train_ds will be divided portion to 80% of total dataset

Part 2 : Testing dataset

This report imported assignment3_framework.ipynb from Canvas RMIT.

```
# Testing Dataset
testsize = 0
test_ds = np.zeros((n_users, n_items))
for row in test_df.itertuples():
    if item_popularity[row[2]-1] > 30:
        test_ds[row[1]-1, row[2]-1] = row[3]
        testsize = testsize + 1
#test_ds = pd.DataFrame(test_ds)

print("Construct the rating matrix based on train_df:")
print(train_ds)

print("Construct the rating matrix based on test_df:")
print(test_ds)

print("Testsize = " + str(testsize))
```

Testing dataset considered as two dimensional and use itertuples for each data rows.

test_ds will be divided portion to 20% of total dataset

Part 3 : Implement AED Similarity Measure

```
import numpy as np

GAMMA = 30
EPSILON = 1e-9
V_min = 1  # Rating scale minimum
V_max = 5  # Rating scale maximum

np_user_aed_sim = np.zeros((n_users, n_users))
```

Part 3 : Implement AED Similarity Measure

```
for i, user_i_vec in enumerate(train_ds.values):
    for j, user_j_vec in enumerate(train_ds.values):

        # Identify indices of co-rated items
        mask_i = user_i_vec > 0
        mask_j = user_j_vec > 0

        # corrated item index, skip if there are no corrated ratings
        corrated_index = np.intersect1d(np.where(mask_i), np.where(mask_j))
        if len(corrated_index) == 0:
            continue

        # Compute Euclidean distance between the rating vectors i and j
        dist = np.sqrt(np.sum((user_i_vec[corrated_index] - user_j_vec[corrated_index])

        # Compute the maximum possible Euclidean distance in the rating scale (0-5)
        dist_max = np.sqrt((V_max - V_min)**2)

        # Calculate similarity
        sim = 1 - dist / (dist_max + EPSILON)
```

$$sim(u,v) = 1 - \frac{dist(\vec{u},\vec{v})}{dist_{\max}} = 1 - \frac{\sqrt{\sum_{s \in S_{uv}} (r_{u,s} - r_{v,s})^2}}{\sqrt{\sum_{1}^{m} (V_{\max} - V_{\min})^2}} \quad ,(10)$$

Part 3 : Implement AED Similarity Measure

```
        # Significance weighting
        weighted_sim = (min(len(corrated_index), GAMMA) / GAMMA) * sim

        np_user_aed_sim[i][j] = weighted_sim

np_user_aed_sim
```

```
array([[ 1.        ,  0.05125314,  0.01396204, ...,  0.06460459,
        -0.56205712, -1.92617498],
       [ 0.05125314,  1.        , -0.08971143, ...,  0.00423389,
        -0.0354102 ,  0.12928932],
       [ 0.01396204, -0.08971143,  1.        , ...,  0.02939887,
        -0.06977556,  0.03333333],
       ...,
       [ 0.06460459,  0.00423389,  0.02939887, ...,  0.56666667,
         0.04309644,  0.04881554],
       [-0.56205712, -0.0354102 , -0.06977556, ...,  0.04309644,
         1.        , -0.02628607],
       [-1.92617498,  0.12928932,  0.03333333, ...,  0.04881554,
        -0.02628607,  1.        ]])
```

Part 3 : Implement AED Similarity Measure

```python
import numpy as np

K = 100  # Number of nearest neighbors
EPSILON = 1e-9

np_predictions = np.zeros((n_users, n_items))

# Iterate over all user-item pairs in the test dataset
for (i, j), rating in np.ndenumerate(test_ds.values):
    if rating > 0:  # Only predict for rated items
        # Find top-K most similar users to the current user, exclude the current user
        sim_user_ids = np.argsort(-np_user_aed_sim[i])[:K]

        # Coefficients of similarity for top K users
        sim_val = np_user_aed_sim[i][sim_user_ids]

        # The ratings of these K users
        sim_users = train_ds.values[sim_user_ids]

        # Users who rated item j
        mask_rated_j = sim_users[:, j] > 0
        if np.any(mask_rated_j):
            sim_val = sim_val[mask_rated_j]
            sim_users = sim_users[mask_rated_j]

            # Compute the prediction
            numer = np.dot(sim_val, sim_users[:, j])
            denom = np.sum(sim_val)

            np_predictions[i][j] = numer / (denom + EPSILON)
             # Clip to convert to the rating scale 0-5
            np_predictions[i][j] = np.clip(np_predictions[i][j], V_min, V_max)
```

Part 3 : Implement AED Similarity Measure

```
#==================MAE on Testing set====================#
labels = test_ds.values

# absolute error on all ratings
absolute_error = np.abs(np_predictions - labels)

# weight
weight = np.clip(labels, 0, 1)

# absoulte error on rated ratings
abs_error = absolute_error * weight

# MAE
MAE = np.sum(abs_error) / np.sum(weight)

print("MAE on Tesing set (User-based): " + str(MAE));
```

```
MAE on Tesing set (User-based): 1.2649306048684652
```

Part 3 : Implement AED Similarity Measure

```
#==================RMSE on Testing set==================
labels = test_ds.values

# squared error on all ratings
squared_error = np.square(np_predictions - labels)
weight = np.clip(labels, 0, 1)

# squared error on rated ratings
squared_error = squared_error * weight
# squared_error = (np_predictions - labels) ** 2 * weight


# RMSE
RMSE = np.sqrt(np.sum(squared_error) / np.sum(weight))

print("RMSE on Tesing set (User-based): " + str(RMSE));
```

```
RMSE on Tesing set (User-based): 1.7545832547201852
```

Part 4 : Implement user based(1)

Compute Pearson Correlation Coefficient for Each Pair of Users in Training Dataset

In [4]:
```
train_ds = pd.DataFrame(train_ds)
test_ds = pd.DataFrame(test_ds)
print(train_ds)
print(test_ds)
```

```
GAMMA = 30
EPSILON = 1e-9

np_user_pearson_corr = np.zeros((n_users, n_users))
```

Using DataFrame to identification data frame

GAMMA

EPSILON is smallest number to prevent calculation process to not divided by zero

np_user_pearson_corr uses numpy to evaluataion and calculation

Part 3 : Implement user based(2)

Compute Pearson Correlation Coefficient for Each Pair of Users in Training Dataset

```
GAMMA = 30
EPSILON = 1e-9

np_user_pearson_corr = np.zeros((n_users, n_users))

for i, user_i_vec in enumerate(train_ds.values):
    for j, user_j_vec in enumerate(train_ds.values):

        # ratings corated by the current pair od users
        mask_i = user_i_vec > 0
        mask_j = user_j_vec > 0

        # corrated item index, skip if there are no corrated ratings
        corrated_index = np.intersect1d(np.where(mask_i), np.where(mask_j))
        if len(corrated_index) == 0:
            continue

        # average value of user_i_vec and user_j_vec
        mean_user_i = np.sum(user_i_vec) / (np.sum(np.clip(user_i_vec, 0, 1)) + EPSILON
        mean_user_j = np.sum(user_j_vec) / (np.sum(np.clip(user_j_vec, 0, 1)) + EPSILON

        # compute pearson corr
        user_i_sub_mean = user_i_vec[corrated_index] - mean_user_i
        user_j_sub_mean = user_j_vec[corrated_index] - mean_user_j

        r_ui_sub_r_i_sq = np.square(user_i_sub_mean)
        r_uj_sub_r_j_sq = np.square(user_j_sub_mean)

        r_ui_sum_sqrt = np.sqrt(np.sum(r_ui_sub_r_i_sq))
        r_uj_sum_sqrt = np.sqrt(np.sum(r_uj_sub_r_j_sq))

        sim = np.sum(user_i_sub_mean * user_j_sub_mean) / (r_ui_sum_sqrt * r_uj_sum_sqr
```

Part 3 : Implement user based(2)

Compute Pearson Correlation Coefficient for Each Pair of Users in Training Dataset

```
        # significance weighting
        weighted_sim = (min(len(corrated_index), GAMMA) / GAMMA) * sim

        np_user_pearson_corr[i][j] = weighted_sim

np_user_pearson_corr
```

```
array([[ 1.        ,  0.15598017,  0.01121976, ...,  0.08421613,
        -0.02888167, -0.05569836],
       [ 0.15598017,  1.        ,  0.04418078, ..., -0.01518162,
         0.02540037,  0.1535033 ],
       [ 0.01121976,  0.04418078,  1.        , ...,  0.0566994 ,
        -0.07264523,  0.03333333],
       ...,
       [ 0.08421613, -0.01518162,  0.0566994 , ...,  0.56666667,
        -0.06666667,  0.00212055],
       [-0.02888167,  0.02540037, -0.07264523, ..., -0.06666667,
         1.        ,  0.17427709],
       [-0.05569836,  0.1535033 ,  0.03333333, ...,  0.00212055,
         0.17427709,  1.        ]])
```

adasadadasds

Part 3 : Implement user based(2)

Compute Pearson Correlation Coefficient for Each Pair of Users in Training Dataset

```
np_predictions = np.zeros((n_users, n_items))

K = 100
EPSILON = 1e-9

for (i, j), rating in np.ndenumerate(test_ds.values):
    if rating > 0:
        # find top-k most similar users as the current user, remove itself
        sim_user_ids = np.argsort(np_user_pearson_corr[i])[-(K + 1):-1]

        # the coefficient values of similar users
        sim_val = np_user_pearson_corr[i][sim_user_ids]

        # the average value of the current user's ratings
        sim_users = train_ds.values[sim_user_ids]
        user_mean = np.sum(train_ds.values[i]) / (np.sum(np.clip(train_ds.values[i], 0, 1)) + EPSILON)
        sim_user_mean = np.sum(sim_users, axis=1) / (np.sum(np.clip(sim_users, 0, 1), axis=1) + EPSILON)

        # select the users who rated item j
        mask_rated_j = sim_users[:, j] > 0

        # sim(u, v) * (r_vj - mean_v)
        sim_r_sum_mean = sim_val[mask_rated_j] * (sim_users[mask_rated_j, j] - sim_user_mean[mask_rated_j])

        np_predictions[i][j] = user_mean + np.sum(sim_r_sum_mean) / (np.sum(sim_val[mask_rated_j]) + EPSILON)
        np_predictions[i][j] = np.clip(np_predictions[i][j], 0, 5)
```

Part 3 : Implement user based(2)

Compute Pearson Correlation Coefficient for Each Pair of Users in Training Dataset

```
#==================MAE on Testing set==================#
labels = test_ds.values

# absolute error on all ratings
absolute_error = np.abs(np_predictions - labels)

# weight
weight = np.clip(labels, 0, 1)

# absoulte error on rated ratings
abs_error = absolute_error * weight

# MAE
MAE = np.sum(abs_error) / np.sum(weight)

print("MAE on Tesing set (User-based): " + str(MAE));
```

```
MAE on Tesing set (User-based): 0.7242796715862666
```

Part 3 : Implement user based(2)

Compute Pearson Correlation Coefficient for Each Pair of Users in Training Dataset

```
#==================RMSE on Testing set====================
labels = test_ds.values

# squared error on all ratings
squared_error = np.square(np_predictions - labels)
weight = np.clip(labels, 0, 1)

# squared error on rated ratings
squared_error = squared_error * weight

# RMSE
RMSE = np.sqrt(np.sum(squared_error) / np.sum(weight))

print("RMSE on Tesing set (User-based): " + str(RMSE));
```

```
RMSE on Tesing set (User-based): 0.933517530304885
```

Comparison and Conclusion

Adjusted Euclidean Distance

Evaluation

MAE (Mean Absolute Error): 1.2649306048684652
RMSE (Root Mean Square Error): 1.7545832547201852

Adjust to real world situation

The AED method in this report need to be improvement in the future to ensure all of formula applied correctly.

User-user KNN based Collaborative Filtering with Centred Cosine similarity

Evaluation

MAE (Mean Absolute Error): 0.7242796715862666
RMSE (Root Mean Square Error): 0.933517530304885

Adjust to real world situation

The User-based method, in this case, appears to perform better than the AED method in terms of both MAE and RMSE.

References

Dataset
F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages.
DOI=http://dx.doi.org/10.1145/2827872

Canvas RMIT - Practical Data Science with Python (2410)
COSC2670 PGRD Semester 1 2024 (2410)
Slides week 9-10