Final Project :

Recommendation System

108062608 黃柏翰

Project 目的：

採取 item based Collaborative Filtering, Similarity 的部分為 Pearson coefficient 來估計, 推估完目前評分過的所有電影後，推薦給各個 user 一部電影

此次作業的程式碼分成 3 個部分:

Preprocess -> Pearson coefficient -> Recommendation

Preprocess

找出有多少不同的評分電影，找出 list for 電影(因為編號有到 19 萬，相異卻只有 9000 個)

```
# For building the E_list for building E[x]
def Preprocess():
    global num_movie, num_user, comb, movie_list, check_movie_id
    data = pd.read_csv(data_path, sep="\t", header=None, names = ["Person", "Movie", "Rating"])
    num_user = data['Person'].max()

    check_movie_id = sorted(list(set(data['Movie'].tolist())))
    num_movie = len(check_movie_id)

    for value in range(num_movie):
        movie_list.update({check_movie_id[value] :value})
```

Pearson coefficient

Rate 做成(Movie, (E[x], sqrt(E[x**2]-E[x]**2), (user1_rate, ...)), 表示這部電影對於所有使用者的評分狀況(含 0)，再做卡式基並求出 Pearson coefficient 後，reduce by key 就能得到各個 movie 對所有電影的相似度

這裡因為 data 的大小關係，所以只用了一半的 user，因為結果用了 collectAsMap 將 Pearson coefficient 做成 dictionary，只取相似度大於 0 者，畢竟在後續計算只考量大於 0

程式碼

```
def Get_Pearson_coef():
    # read Data
    global User_list, Pearson_list, Pearson_dict
    data = sc.textFile(data_path)

    # Make it into (user, (movie rating.......))
    User = data.map(shuffle).reduceByKey(lambda a, b: a + b).mapValues(update_User_list).cache()


    # Make them into (Movie, Rating) key-value pair in each data,
    # Use E_list to build whole list for it
    # Rate for (Movie, (E[x], sqrt(E[x**2]-E[x]**2), (user1_rate, .....))
    Rate = data.map(shuf).reduceByKey(lambda a, b: a + b).mapValues(update_Elist).mapValues(make_Exp)
    #print(Rate.first())
    # make it to ((movie, movie), ((E[x], sqrt(E[x**2]-E[x]**2), (user1_rate, .....),(E[x], sqrt(E[x**2]-E[x]**2), (user1_rate, .....)))
    Rate_comb = Rate.cartesian(Rate).map(lambda x: (tuple((x[0][0], x[1][0])), x[0][1], x[1][1]))


    # Find Pearson list (movie, Pearson coef movie to all)
    Pearson_list = Rate_comb.map(Pearson).reduceByKey(lambda a, b : a + b).mapValues(sortkey).cache()
    Pearson_list.unpersist()


    Pearson_dict = Pearson_list.collectAsMap()
    del Pearson_list, Rate, Rate_comb

    return User
```

Recommendation

只針對沒有評分的再去處理，擔心會影響整體評分，所以不考慮後來更新分數後的分數，只考量原來有評分的，計算方法是用相似度和評分 dot product 再除上相似度的和

程式碼

```
################################
#        Rate for others       #
################################
def Recommendation(data):

    user_id = data[0]
    # recording the rating in 9700 movies of a user
    m_list = data[1]
    rate = []
    id_list = []

    # total movie rating -> about 9700 movies
    for movie in range(len(m_list)):
        Point = m_list[movie]

        # if this movie is not rating
        if Point == 0:
            # find the right list of movie rating
            sub_p = Pearson_dict[check_movie_id[movie]]
            r = sum([m * p for m, p in zip(m_list, sub_p)])
            sum_sub_p = sum(sub_p)

            if sum_sub_p != 0:
                r /= sum_sub_p

            rate.append(r)
            id_list.append(check_movie_id[movie])

    rec_movie_id = id_list[rate.index(max(rate))]

    return (user_id, rec_movie_id)


# Find the best recommendation for user
def Rate(User):
    # Make it to ((User, Rating), P_list), calculate Pearson coef real-time
    Rec = User.map(Recommendation)
    User.unpersist()
    del User
    Result = Rec.collect()

    return sorted(Result, key = lambda x : x[0])
```

最後寫入 output.txt

後記

這次因為 data 太大而刪減 data 使得程式運行順利，也因此加入了幾個小技巧，像是刪除用不到的資源，以及更改原本 memory setting

結果如圖(只取部分截圖)

```
Recommend User 1 : Movie 2342
Recommend User 2 : Movie 140956
Recommend User 3 : Movie 59103
Recommend User 4 : Movie 2342
Recommend User 5 : Movie 280
Recommend User 6 : Movie 280
Recommend User 7 : Movie 55854
Recommend User 8 : Movie 27416
Recommend User 9 : Movie 8605
Recommend User 10 : Movie 7316
Recommend User 11 : Movie 547
Recommend User 12 : Movie 3496
Recommend User 13 : Movie 5109
Recommend User 14 : Movie 280
Recommend User 15 : Movie 8605
Recommend User 16 : Movie 142115
Recommend User 17 : Movie 142115
Recommend User 18 : Movie 140956
Recommend User 19 : Movie 2103
Recommend User 20 : Movie 7303
Recommend User 21 : Movie 102007
Recommend User 22 : Movie 7312
Recommend User 23 : Movie 142115
Recommend User 24 : Movie 142115
Recommend User 25 : Movie 140956
Recommend User 26 : Movie 142115
Recommend User 27 : Movie 7303
Recommend User 28 : Movie 5319
Recommend User 29 : Movie 160848
```