Effective Missing Data Prediction for Collaborative Filtering

Practical Data Science COSC2670

HAMED ALKAFF

AUG 2021

WHAT IS COLLABORATIVE FILTERING

Collaborative Filtering is a technique used in recommender systems to predict a user's preferences based on the preferences of other similar users. The core idea here is to identify similarities or patterns among users or items to make personalized recommendations.

Figure 1 illustrates a simple example, Jules and Emma have similar interests for Jobs 1 and 2. However, Emma is also interested in Job 3, therefore Job 3 is recommended for Jules. This is merely an explanation of the idea, the actual algorithm and implementation, on the other hand, is more complex and precise, which will be discussed next.

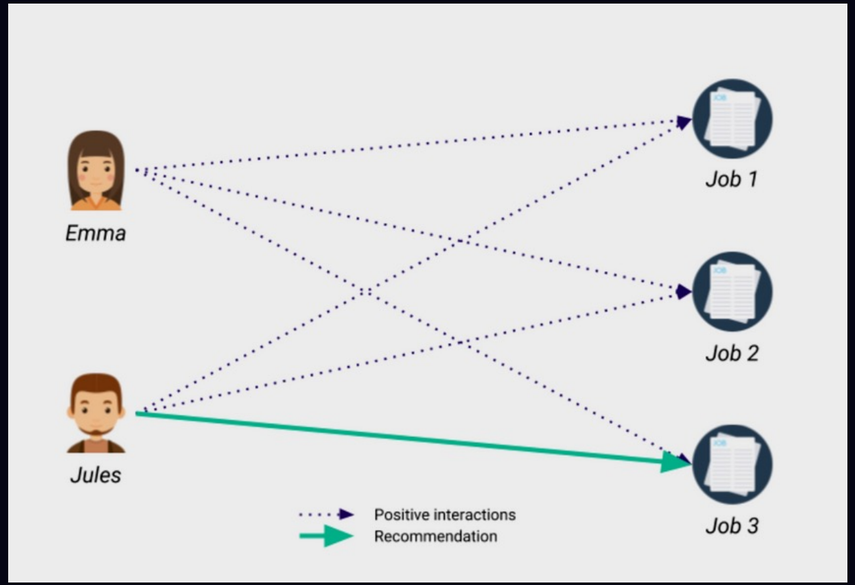

Figure 1

APPROACHES OF COLLABORATIVE FILTERING

- **Model-Based Approaches**
  In this model, training datasets are used to train a pre-defined model. Examples of this approach are Clustering models and Aspect models.

- **Memory-Based Approaches**
  These approaches are the most popular ones. Examples of these approaches are User-based and item-based approaches.

  - **User Based**
    Predicts user ratings based on other similar users' ratings.
  - **Item Based**
    Predicts the ratings of users based on the information of similar items computed.

PEARSON COEFFICIENT CORRELATION

PCC is applied to be able to define the similarity between two users (user a, user u) based on items they have rated in common.

- Similarity between users
  As figure 2 shows, *Sim(a, u)* denotes the similarity between the users, *i* denoted to the set of items which both users rated in common. *ra,i* is the rate user a gave to item *i,* and bar ra is the average rate of user *i.*

- Similarity between items
  The similarity between two items is denoted as *Sim(i, j).* As figure 3 shows, the equation is quite identical to the equation in figure 2.

But what happens if the users happen to have rated a few items identically, but may have different styles or preferences?

$$Sim(a,u) = \frac{\sum_{i \in I(a) \cap I(u)} (r_{a,i} - \overline{r}_a) \cdot (r_{u,i} - \overline{r}_u)}{\sqrt{\sum_{i \in I(a) \cap I(u)} (r_{a,i} - \overline{r}_a)^2} \cdot \sqrt{\sum_{i \in I(a) \cap I(u)} (r_{u,i} - \overline{r}_u)^2}},$$

Figure 2

$$Sim(i,j) = \frac{\sum_{u \in U(i) \cap U(j)} (r_{u,i} - \overline{r}_i) \cdot (r_{u,j} - \overline{r}_j)}{\sqrt{\sum_{u \in U(i) \cap U(j)} (r_{u,i} - \overline{r}_i)^2} \cdot \sqrt{\sum_{u \in U(i) \cap U(j)} (r_{u,j} - \overline{r}_j)^2}},$$

Figure 3

SIGNIFICANCE WEIGHTING

PCC can sometimes overestimate the similarity between users who rated only a few items in common identically. Thus, Significance weighting is introduced, which will devalue any similarity weights that were based on a small number of co-rated items.

As figure 4 shows, */Ia ∩ Iu/* is the items rated by both users, and we use Min to ensure that the interval and the values are always in the range of 0 to 1.

$$Sim'(a,u) = \frac{Min(|I_a \cap I_u|, \gamma)}{\gamma} \cdot Sim(a,u),$$

Figure 4

$$Sim'(i,j) = \frac{Min(|U_i \cap U_j|, \delta)}{\delta} \cdot Sim(i,j),$$

Figure 5

SIMILAR NEIGHBORS SELECTION

Similar neighbor selection is one of the main steps in predicting missing data. The solution given in the report introduces two thresholds called ITA and THETA, these thresholds are used to overcome the flaws of Top-N neighbor algorithms. in brief, the idea of thresholds is that if the similarity between the neighbor and the current user is greater than the threshold ITA, then this neighbor is selected as a similar user, Figure 6 illustrates it.

$$S(u) = \{u_a | Sim'(u_a, u) > \eta, u_a \neq u\},$$

Figure 6

Figure 7 shows the equation for item based, the main difference is it uses THETA, and it can be a different value. Further, the selection of these thresholds is an important step because higher thresholds will cause shortage of similar users and low thresholds will give too many similar users

$$S(i) = \{i_k | Sim'(i_k, i) > \theta, i_k \neq i\},$$

Figure 7

HOW DOES THE SOLUTION PREDICT MISSING VALUES?

The solution provided in the report starts off the process of predicting missing values by finding the set of similar users or items for the current active user. Meaning, the first step of predicting missing values is calculating the PCC values for User-based as well as the PCC values for the item-based. Here are the steps in sequence:

1- Calculating the PCC values of the User-based and Item-based
2- Finding the set of similar users or items.
3- Predict the missing values based on those ratings of similar users or items depending on certain conditions that will be discussed in the next slide.

The solution in this report solves the missing values problem in a different way, it utilizes both User-based and item-based Collaborative Filtering to find the missing values.
In other words, the solution specifies a set of conditions for the missing value to be predicted, and based on the satisfied condition, the approach (item or user based) for the missing value is decided. The reason for this is because if we use one approach, then we are neglecting valuable information that can help make the prediction more accurate. The next slide will demonstrate how.

HOW DOES THE SOLUTION PREDICT MISSING VALUES?

As mentioned before, there is a set of conditions which will decide how to predict a certain missing value. First, we will denote the set of similar users as *S(u)* and denote the set of similar items as *S(i)*. There are 4 conditions in order to predict the missing values and they are:

1- if *S(u)* and *S(i)* are not empty, if this is the case for the current user, we use the equation in figure 10.

2- if *S(u)* is not empty and *S(i)* is empty, then we use the equation in figure 9.

3- if *S(u)* is empty and *S(i)* is not empty, we use the equation in figure 10.

4- if *S(u)* and *S(i)* are both empty, we use the equation in figure 11.

This way, we are utilizing user-based and item-based collaborative filtering, and we don't waste any valuable information, and thus the solution is more accurate in predicting the missing values.

$$P(r_{u,i}) = \lambda \times (\overline{u} + \frac{\sum_{u_a \in S(u)} Sim'(u_a, u) \cdot (r_{u_a,i} - \overline{u}_a)}{\sum_{u_a \in S(u)} Sim'(u_a, u)}) + (1 - \lambda) \times (\overline{i} + \frac{\sum_{i_k \in S(i)} Sim'(i_k, i) \cdot (r_{u,i_k} - \overline{i}_k)}{\sum_{i_k \in S(i)} Sim'(i_k, i)}),$$

Figure 8

$$P(r_{u,i}) = \overline{u} + \frac{\sum_{u_a \in S(u)} Sim'(u_a, u) \cdot (r_{u_a,i} - \overline{u}_a)}{\sum_{u_a \in S(u)} Sim'(u_a, u)}.$$

Figure 9

$$P(r_{u,i}) = \overline{i} + \frac{\sum_{i_k \in S(i)} Sim'(i_k, i) \cdot (r_{u,i_k} - \overline{i}_k)}{\sum_{i_k \in S(i)} Sim'(i_k, i)}.$$

Figure 10

$$P(r_{u,i}) = 0.$$

Figure 11

WHY CAN THE SOLUTION TACKLE THE MISSING DATA PROBLEM?

The provided solution has new approaches to solve certain common problems when predicting the missing data. First, it introduced the significance weighting factor when calculating the PCC values, this significance weighting factor ensures that there is no possible overestimation of similarity of those users who have few item ratings in common, this helps in predicting the data more accurately.

The second reason is that the solution introduced thresholds, to avoid any dissimilarity between the user and its neighbors. Meaning, it compares each neighbor's similarity with the current user, and if the similarity is not greater than the threshold, the neighbor is not selected anymore, which increases the accuracy of the predictions. Further, another reason why the solution can tackle the missing data problem is that it introduced conditions to be able to make use of both user based and item based without wasting valuable information.

HOW IS THE SOLUTION IMPLEMENTED?

STEP 1

Calculate the Pearson Coefficient Correlation for both user-based and item-based Collaborative Filtering

STEP 2

Find the set of similar users or similar items for the current user.

STEP 3

Make the 4 conditions that decide how to predict the missing value

STEP 4

Predict the missing values

HOW IS THE SOLUTION IMPLEMENTED?

STEP 1

Code wise, the implementation of the first step of the solution is by doing the following code in figure 12

This code calculates the PCC values for the user-based Collaborative Filtering approach.
Further, the code for calculating the PCC values for the item-based CF is almost identical except that we use np.Transpose, which interchanges the rows and columns of the data.

$$Sim(a,u) = \frac{\sum_{i \in I(a) \cap I(u)} (r_{a,i} - \overline{r}_a) \cdot (r_{u,i} - \overline{r}_u)}{\sqrt{\sum_{i \in I(a) \cap I(u)} (r_{a,i} - \overline{r}_a)^2} \cdot \sqrt{\sum_{i \in I(a) \cap I(u)} (r_{u,i} - \overline{r}_u)^2}},$$

Figure 13

```python
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_sqrt + EPSILON)

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

        np_user_pearson_corr[i][j] = weighted_sim

np_user_pearson_corr
```

Figure 12

HOW IS THE SOLUTION IMPLEMENTED?

STEP 2

The second step of the solution is to implement the conditions that decide what the prediction equation is. As figure 14 shows, the first if-statement checks if the sets of similar users our items is empty or not, and then it takes equation 8 if the condition is satisfied.

The implementation of *S(u)*

```
for (i, j), rating in np.ndenumerate(train_ds.values):

    if rating > 0:

        #============USER-BASED============#

        new_user_ids = np.array([])
        new_user_ids = new_user_ids.astype(int)
        sim_user_ids = np.argsort(imputed_trained_pearson_corr[i])[-(K+1):-1]

        for x in sim_user_ids:

            if imputed_trained_pearson_corr[i][x] > ITA:

                new_user_ids = np.append(new_user_ids, x)

            sim_val = imputed_trained_pearson_corr[i][new_user_ids]

            sim_users = imputed_train_ds.values[new_user_ids]
```

Figure 14

$$S(u) = \{u_a | Sim'(u_a, u) > \eta, u_a \neq u\},$$

Figure 15

HOW IS THE SOLUTION IMPLEMENTED?

STEP 3 & STEP 4

The third step of the solution is to implement the conditions that decide what the prediction equation is. As figure 16 below shows, the first if-statement checks if the sets of similar users and items are empty or not, and then it takes equation 8 if the condition is satisfied.

We will denote the set of similar users as *S(u)*, and *S(i)* for the set of similar items.

STEP 4 is the actual prediction for the missing value.

```
#Equation 8
if new_user_ids.size != 0 and new_item_ids.size != 0:

    train_ds_pred[i][j] = LAMBDA * (user_mean + user_based_pred)
                        + (1 - LAMBDA) * (item_mean + item_based_pred)
    train_ds_pred[i][j] = np.clip(train_ds_pred[i][j], 0, 5)

#Equation 9
elif new_user_ids.size != 0 and new_item_ids.size == 0:

    train_ds_pred[i][j] = (user_mean + user_based_pred)
    train_ds_pred[i][j] = np.clip(train_ds_pred[i][j], 0, 5)

#Equation 10
elif new_user_ids.size == 0 and new_item_ids.size != 0:

    train_ds_pred[i][j] = (item_mean + item_based_pred)
    train_ds_pred[i][j] = np.clip(train_ds_pred[i][j], 0, 5)


#Equation 11
elif new_user_ids.size == 0 and new_item_ids.size == 0:

    train_ds_pred[i][j] = 0
```

Figure 16

1- if *S(u)* and *S(i)* are not empty

$$P(r_{u,i}) = \lambda \times (\overline{u} + \frac{\sum_{u_a \in S(u)} Sim'(u_a, u) \cdot (r_{u_a,i} - \overline{u}_a)}{\sum_{u_a \in S(u)} Sim'(u_a, u)}) + (1 - \lambda) \times (\overline{i} + \frac{\sum_{i_k \in S(i)} Sim'(i_k, i) \cdot (r_{u,i_k} - \overline{i}_k)}{\sum_{i_k \in S(i)} Sim'(i_k, i)}),$$

Equation 8

2- if *S(u)* is not empty and *S(i)* is empty

$$P(r_{u,i}) = \overline{u} + \frac{\sum_{u_a \in S(u)} Sim'(u_a, u) \cdot (r_{u_a,i} - \overline{u}_a)}{\sum_{u_a \in S(u)} Sim'(u_a, u)}.$$

Equation 9

3- If *S(u)* is empty and *S(i)* is not empty

$$P(r_{u,i}) = \overline{i} + \frac{\sum_{i_k \in S(i)} Sim'(i_k, i) \cdot (r_{u,i_k} - \overline{i}_k)}{\sum_{i_k \in S(i)} Sim'(i_k, i)}.$$

Equation 10

4- if *S(u)* and *S(i)* are both empty

$$P(r_{u,i}) = 0.$$

Equation 11

THANK YOU

REFERENCES

RMIT Practical Data Science Tutorial 9 & 10

RMIT Practical Data Science Lecture 9 & 10

RMIT Practical Data Science Pre-recorded videos week 9 & 10

Ankita P 2020, gitconnected, "The Mathematics Behind Recommendation systems"
https://levelup.gitconnected.com/the-mathematics-of-recommendation-systems-e8922a50bdea