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

RMIT UNIVERSITY

Practical Data Science COSC2670

Adjusted Weighted Slope One Scheme (AWSS)
Fit, predict and recommend items to users

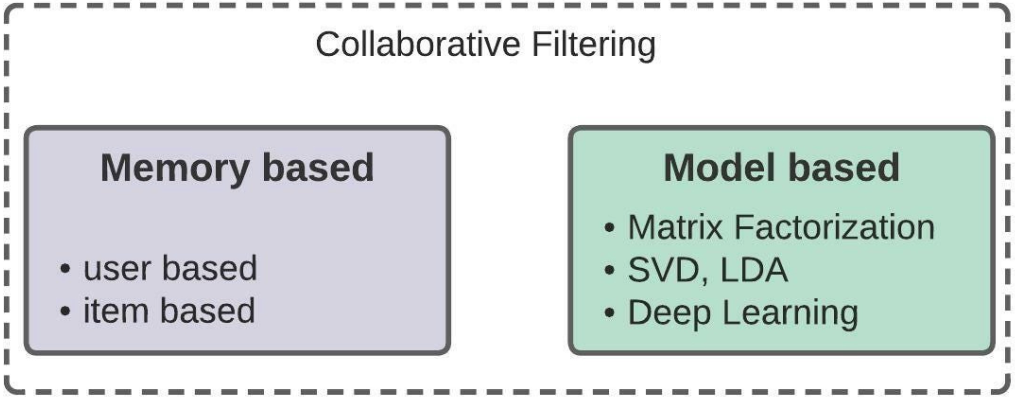
Mrwan Fahad A Alhandi s3969393

Date: 24/06/2023

RMIT UNIVERSITY

Overview of Prediction Algorithms

- Memory vs Model based



Overview of Prediction Algorithms

1- Per User Average

$$\boxed{P(u) = \bar{u}}$$

2- Bias From Mean

$$P(u)_i = \bar{u} + \frac{1}{card(S_i(\chi))} \sum_{v \in S_i(\chi)} v_i - \bar{v}.$$

3- Adjusted Cosine Similarity

$$P(u)_i = \frac{\sum_{j \in S(u)} |\mathrm{sim}_{i,j}| (\alpha_{i,j} u_j + \beta_{i,j})}{\sum_{j \in S(u)} |\mathrm{sim}_{i,j}|}$$

4- Pearson

$$P(u)_i = \bar{u} + \frac{\sum_{v \in S_i(\chi)} \gamma(u,v)(v_i - \bar{v})}{\sum_{v \in S_i(\chi)} |\gamma(u,v)|}$$

Slope One Scheme

Equation:

$$f(x) = x + b \text{ or } f(u_i) = u_i + dev_{j,i}$$

Deviation:

$$\text{dev}_{j,i} = \sum_{u \in S_{j,i}(\chi)} \frac{u_j - u_i}{card(S_{j,i}(\chi))}.$$

Prediction Algorithm With No Average:

$$P(u)_j = \frac{1}{card(R_j)} \sum_{i \in R_j} (\text{dev}_{j,i} + u_i)$$

Prediction Algorithm With Average Term:

$$P^{S1}(u)_j = \bar{u} + \frac{1}{card(R_j)} \sum_{i \in R_j} \text{dev}_{j,i}.$$

RMIT UNIVERSITY

Weighted Slope One

$$P^{wS1}(u)_j = \frac{\sum_{i \in S(u)-\{j\}} (\text{dev}_{j,i} + u_i) \boxed{c_{j,i}}}{\sum_{i \in S(u)-\{j\}} \boxed{c_{j,i}}}$$

where $\boxed{c_{j,i} = card(S_{j,i}(\chi)).}$

- If we want to predict L item for user A

- User A → J and K

- 2000 users → J and L

- 20 users → K and L

- J is better predictor for A L item than A rating for K

Class AWSS (Adjusted Weighted Slope Scheme)

This class have three methods:

- Fit
- Predict
- Recommend_items

```python
class AWSS:
    """
    This class is to implement the AWSS algorithm which stand for Adjusted Weighted Slope Scheme.
    The class has three methods:
    1. fit: To train the model by calculating the deviation matrix
    2. predict: To predict the Loading... or the test dataset
    3. recommend_movies: To recommend movies for a given user
    """
    def __init__(self, train_ds, n_users = n_users, n_items = n_items, limbda = 0.05, GAMMA = 30, EPSILON = 1e-6):
        """
        self.train: The training dataset
        self.n_users: The number of users
        self.n_items: The number of items
        self.limbda: Model hyperparameter to control the two parts of the equation. First: ∑u∈Sj,i(χ) ( (uj − ui) / card(Sj,i(χ
        self.GAMMA: Model hyperparameter to control the contribution of similarity between users. A higher GAMMA would give mor
        self.EPSILON: A small value to avoid division by zero
        self.devs: The deviation matrix
        self.counts: The number of users who rated both items
        self.preds: The prediction matrix
        """
        self.train_ds = train_ds
        self.n_users = n_users
        self.n_items = n_items
        self.limbda = limbda
        self.GAMMA = GAMMA
        self.EPSILON = EPSILON
        self.devs = None
        self.counts = None
        self.preds = None

    def fit(self):
        """
        This method is to train the model by calculating the deviation matrix.
        Based on the following equation:
```

fit method

Full Equation:

$$dev_{j,i} = \lambda \sum_{u \in S_{j,i}(\chi)} \frac{u_j - u_i}{card(S_{j,i}(\chi))} + (1-\lambda) \frac{\sum_{u \in S_{j,i}(\chi)} ((u_j - u_i) \cdot exp(sim(u,u')))}{\sum_{u \in S_{j,i}(\chi)} (exp(sim(u,u')) \cdot card(S_{j,i}(\chi)))},$$

Two new things:

- A new term to the equation which represents the influence of the similarity between users who rated both items
- If limda is 0, the first term will turn of and the second term will have full involvement in the calculation

RMIT UNIVERSITY

Deviation: Term 1 Of Equation

Term 1 Of Equation:

$$\sum_{u \in S_{j,i}(\chi)} \frac{u_j - u_i}{card(S_{j,i}(\chi))}$$

	Item 1	Item 2	Item 3	Item 4
User 1	3	5		1
User 2	4		4	2
User 3		1	5	3
User 4	5	4		

If we considered a single iteration example:

j: item 1

i: item 2

We calculate **(uj - ui)** for each user in **S1,2(χ)**:

- For User 1, **u1 - u2** = 3 - 5 = -2.
- For User 4, **u1 - u2** = 5 - 4 = 1.

So, **dev1,2** = **ΣuЄS1,2(χ) (uj - ui)** / **card(S1,2(χ))** = (-2 + 1) / 2 = -0.5.

Deviation: Term 2 Of Equation

Term 2 Of Equation:

$$\frac{\sum_{u \in S_{j,i}(\chi)} ((u_j - u_i) \cdot exp(sim(u, u')))}{\sum_{u \in S_{j,i}(\chi)} (exp(sim(u, u')) \cdot card(S_{j,i}(\chi)))}$$

	Toy Story	Star Wars	Similarity to User 1	Exponential similarity
User1	5	4	1	2.72
User2	4	3	0.9	2.46
User3	5	5	0.8	2.23

The computation for the numerator would be:

((5-4) * 2.72) + ((4-3) * 2.46) + ((5-5) * 2.23) = 2.72 + 2.46 = 5.18

For the denominator:

(2.72 * 3) + (2.46 * 3) + (2.23 * 3) = 23.49

Hence, the resulting second term of deviation for "Toy Story" and "Star Wars" ratings, considering the affinities to User1, is calculated as 5.18 / 23.49 = 0.22.

Where sim(u,u') is the centered cosine similarity.

RMIT UNIVERSITY

predict method

$$P^{wS1}(u)_j = \frac{\sum_{i \in S(u)-\{j\}} (\text{dev}_{j,i} + u_i) \boxed{c_{j,i}}}{\sum_{i \in S(u)-\{j\}} \boxed{c_{j,i}}}$$

where $\boxed{c_{j,i} = card(S_{j,i}(\chi)).}$

recommend_items method

```python
def recommend_items(self,user_id, top_n=10):
    """
    This method is to recommend movies for a given user.
    """
    # Make sure predictions have been calculated
    preds = AWSS.predict(self)

    # Get the user's predicted ratings
    user_ratings = preds[user_id]

    # Get the indices of the user's top n ratings
    top_n_indices = np.argsort(user_ratings)[-top_n:]

    # Get the movie names corresponding to the top n indices
    movie_names = df_movies[df_movies['movie_id'].isin(top_n_indices)]['movie_title'].values

    return movie_names
```

RMIT UNIVERSITY

Thank you for listening

RMIT UNIVERSITY