Recommend System Study

Section 3
Collaborative Filtering

2022.07.06

Jihyun Lee

20. Collaborative Filtering Section Introduction

Goal : Personalizing the score
Idea: Rank each item by a score

- e.g. PageRank, Reddit, HN, Bandit
- Call the score s(j) ; j = 1, … ,M , if there are M items

Notations

- Basic algorithm (which we covered) is to make s(j) the average rating for j

$$s(j) = \frac{\sum_{i \in \Omega_j} r_{ij}}{|\Omega_j|}$$

$$\Omega_j = \textit{set of all users who rated item } j$$

$$r_{ij} = \textit{rating user } i \textit{ gave item } j$$

Personalize the score

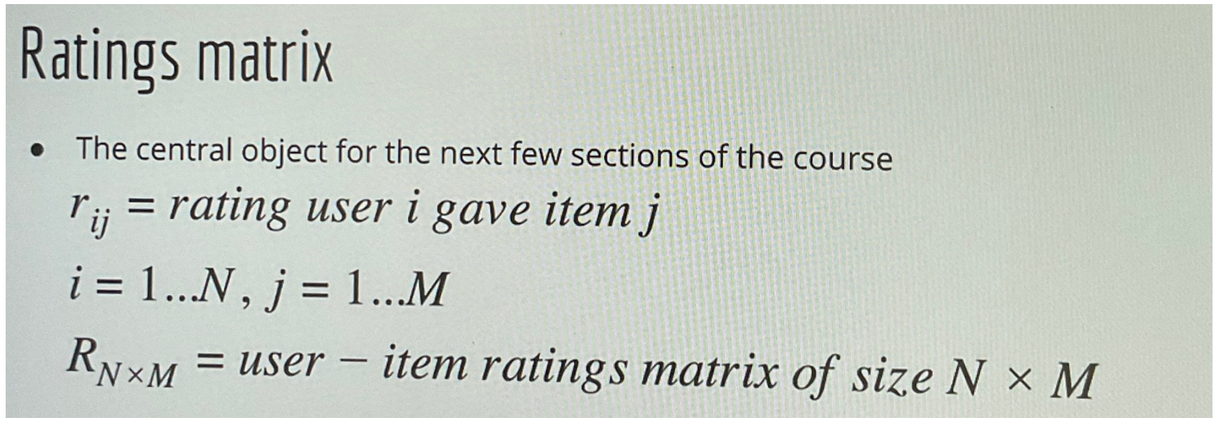
- s(i,j) can depend both on user i and item j
- Note: i' is just an index

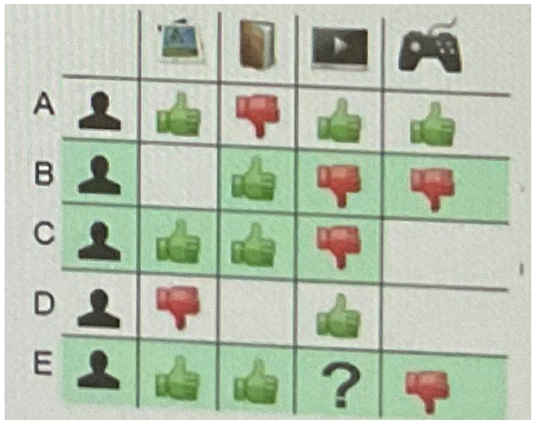
$$s(i,j) = \frac{\sum_{i' \in \Omega_j} r_{i'j}}{|\Omega_j|}$$

$$i = 1 \ldots N,\ N = \textit{number of users}$$

20. Collaborative Filtering Section Introduction

Rankings Matrix





Sparsity

- User-item matrix is sparse because most entries are **empty! (not 0)**
- Most of r(I, j) doesn't exit – this is good. Why?
  - Dense matrix is convenient mathematically, but sucks business-wise
  - It means every user has already seen every movie!
  - Noting to recommend
  - The matrix must be sparse in order to actually have items to recommend

20. Collaborative Filtering Section Introduction

Goal

- We want to guess what. You might rate a movie you haven't seen yet
  - E.g. if we think you might rate some movie a 5, we definitely want you to watch that movie

$$s(i,j) = \hat{r}(i,j) = \textit{guess what user } i \textit{ might rate item } j$$

Regression

- We want to predict a real number, so objective is MSE (mean-squared error)
- Outline
  - User-User Collaborative Filtering
  - Item-Item Collaborative Filtering

0. Collaborative Filtering

많은 유저들로부터 모은 취향 정보들을 기반으로 하여, 스스로 예측하는 기술

- 어떤 특정한 인물 A 가 한가지 이슈에 관해서 인물 B 와 같은 의견을 갖는다면, 다른 이슈에 대해서도 비슷한 의견을 가질 확률이 높을 것이라는 사실에 기반 (집단 지성)
- 기본적으로 collaborative filtering 의 종류는 3가지
  - **<u>Memory-based</u>**
  - Model-based
  - Hybrid

- **Memory-based Collaborative Filtering**
  - 유사도를 기반으로 동작
  - 사용자-사용자 간의 유사도를 기준으로 하는 경우 **(User-Based)**
  - 아이템-아이템 간의 유사도를 기준으로 하는 경우 **(Item-Based)**
  - 사용자 기반 vs 아이템 기반
    - 결론부터 말하자면, 아마존, 넷플릭스를 비롯한 서비스에서는 대부분 아이템 기반 활용
    - 유저가 아이템을 평가하는 순간, 다른 아이템을 추천할 수 있어야 하는데, 매 평가시마다 유사도 정보를 업데이트하는 것은 현실적으로 어려움.
    - 아이템 기반에서는 일정 기간마다 유사도를 업데이트 하는 것으로도 적절한 UX 제공
    - 대부분의 경우 사용자에 비해 아이템 수가 적기 때문에, 아이템 간의 관계 데이터가 발생할 확률이 높다. 따라서 데이터가 누적될수록 추천의 정확도가 높아질 가능성이 더 큼.

0. Collaborative Filtering

User-Based (사용자 기반)

- **<u>유사도 : 두 사용자가 얼마나 유사한 항목 (아이템) 을 선호했는지.</u>**
- 한 사용자가 평가한 영화들의 점수를 벡터로 나타냄.
  - 재석의 평가 점수 (5,4,4,3,-)
- 이 때 두 사용자 간의 유사도 : 두 벡터간의 유사도 (**코사인 유사도, 피어슨 유사도**)

사용자 기반, 아이템 기반의 협업 필터링을 설명하기 앞서 간단한 예시 상황을 가정하도록 한다. 다음은 어떤 사용자들이 영화의 평점을 (0~5) 사이의 값으로 매긴 결과이다. 빈칸은 평가를 하지 않은 항목이다.

	공조	더 킹	라라랜드	컨택트	너의 이름은
재석	5	4	4	3	
명수	1	0	1		4
하하	4	4		5	3
준하		2	1	4	3
세형	4		4	4	2
광희	4	2	3		1

출처 : https://scvgoe.github.io/2017-02-01-%ED%98%91%EC%97%85-%ED%95%84%ED%84%B0%EB%A7%81-%EC%B6%94%EC%B2%9C-%EC%8B%9C%EC%8A%A4%ED%85%9C-(Collaborative-Filtering-Recommendation-System)/

0. Collaborative Filtering | 유사도 (similarity)

벡터 간 유사도

- 어떤 벡터와 어떤 벡터가 가장 비슷하고, 가장 비슷하지 않은지 판단

- E.g. 상품 x,y,z 가 있다고 가정
  - 각 상품의 속성들을 숫자로 잘 뽑으면, 상품들의 속성 값들을 벡터로 표현할 수 있음.
  - 아이템마다 벡터화를 해주고나면, 아이템간 유사도를 구할 수 있음.
  - 어떤 아이템이 어떤 아이템과 가장 비슷한가? 비슷하지 않은가?
  - 벡터를 이용하여 계산할 수 있는 유사도

X = [1,0,2,3,2]

Y = [1,0,3,2,1]

Z = [0,1,3,1,1]

- 종류 (3가지)
  - Jaccard Similarity
  - Cosine Similarity
  - Pearson Similarity

0. Collaborative Filtering | 유사도 (similarity)

1. Jaccard Similarity

- 합집합과 교집합 사이으 | 비율

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

```
from math import *

def jaccard_similarity(x, y):
    intersection_cardinality = len(set.intersection(*[set(x), set(y)]))
    union_cardinality = len(set.union(*[set(x), set(y)]))

    return intersection_cardinality / float(union_cardinality)

print(jaccard_similarity(['A', 'B', 'C', 'D'],['A', 'C', 'F', 'G']))
```

2. Cosine Similarity

- 두 벡터의 코사인 각도 계산
- -1 (유사하지 않음) ~ 1 (유사함)

$$cosine(X_r, X_s) = \frac{X_r \cdot X_s}{\|X_r\| \|X_s\|} = \frac{\sum_{i=1}^{n} X_{ri} X_{si}}{\sqrt{\sum_{i=1}^{n} X_{ri}^2} \sqrt{\sum_{i=1}^{n} X_{si}^2}}$$

```
import numpy as np

def cosine_similarity(a, b):
    return np.dot(a, b) / (np.linalg.norm(a) * (np.linalg.norm(b)))

print(cosine_similarity([3, 10, 7, 2], [2, 15, 10, 12]))
```

3. Pearson Similarity

- 벡터의 표본평균으로 normalization 을 하고, 코사인 유사도를 구함.
- -1 (음의 상관관계) ~ 1 (양의 상관관계)
- 0 : 상관관계 없음 (독립)

$$r_{rs} = \frac{\sum_{i=1}^{n}(X_{ri} - \overline{X_r})(X_{si} - \overline{X_s})}{\sqrt{\sum_{i=1}^{n}(X_{ri} - \overline{X_r})^2} \sqrt{\sum_{i=1}^{n}(X_{si} - \overline{X_s})^2}}$$

```
from scipy.stats import pearsonr

def pearson_similarity(a, b):
    return np.dot((a - np.mean(a)), (b - np.mean(b))) / ((np.linalg.norm(a -
np.mean(a))) * (np.linalg.norm(b - np.mean(b))))

display(pearson_similarity([3, 10, 7, 2], [2, 15, 10, 12])
        , pearsonr([3, 10, 7, 2], [2, 15, 10, 12])
        , np.corrcoef([3, 10, 7, 2], [2, 15, 10, 12]))
```

```
0.5756777782280114
(0.5756777782280114, 0.42432222177198864)
array([[1.        , 0.57567778],
       [0.57567778, 1.        ]])
```

출처 : https://abluesnake.tistory.com/100

0. Collaborative Filtering

User-Based (사용자 기반)

- **유사도 : 두 사용자가 얼마나 유사한 항목 (아이템) 을 선호했는지.**
- 이 때 두 사용자 간의 유사도 : 두 벡터간의 유사도 (**코사인 유사도, 피어슨 유사도**)

코사인 유사도를 이용해 명수와 준하의 유사도를 구해보자. 유사도를 구할 때에는 두 사용자가 공통으로 평가한 항목에 대해서만 계산한다. A=(0, 1, 4), B=(2, 1, 3)이고, 코사인 유사도는 다음과 같이 정의된다.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$, where A_i and B_i are components of vector A and B respectively.

$$similarity = \frac{(0,1,4)\cdot(2,1,3)}{\sqrt{0^2+1^2+4^2}\times\sqrt{2^2+1^2+3^2}} = \frac{13}{\sqrt{17}\times\sqrt{14}} \cong 0.84$$

유사도	재석	명수	하하	준하	세형	광희
재석	1.00	0.84	0.96	0.82	0.98	0.98
명수	0.84	1.00	0.61	0.84	0.63	0.47
하하	0.96	0.61	1.00	0.97	0.99	0.92
준하	0.82	0.84	0.97	1.00	0.85	0.71
세형	0.98	0.63	0.99	0.85	1.00	0.98
광희	0.98	0.47	0.92	0.71	0.98	1.00

이때 세형의 더 킹에 대한 평가 점수를 예측해보고 싶다고 할 때, 우리는 여러 방법을 적용할 수 있다. 세형과 가장 유사한 몇명의 점수를 이용하여 예측 점수로 구할 수도 있고, 전체를 대상으로 유사도 기반의 weighted sum 값을 예측 점수로 사용할 수도 있다. 후자의 경우로 세형의 더 킹에 대한 평가 점수를 예측해보자.

$$prediction = \frac{0.98\times4+0.63\times0+0.99\times4+0.85\times2+0.98\times2}{0.98+0.63+0.99+0.85+0.98} = 2.60$$

0. Collaborative Filtering

Item-Based (아이템 기반)

- **<u>유사도 : 아이템들에 대한 유사도 계산.</u>**
- 이 때 두 아이템 간의 유사도 : 두 벡터간의 유사도 (**코사인 유사도, 피어슨 유사도**)

	공조	더 킹	라라랜드	컨택트	너의 이름은
재석	5	4	4	3	
명수	1	0	1		4
하하	4	4		5	3
준하		2	1	4	3
세형	4		4	4	2
광희	4	2	3		1

아이템 기반도 앞선 사용자 기반과 매우 유사한 과정을 거친다. 다만 앞에서는 사용자에 대한 유사도를 계산했다면, 여기서는 아이템들에 대한 유사도를 계산한다. 마찬가지로 두 아이템에 대한 사용자들의 평가 점수를 벡터로 나타내보자. 공조와 라라랜드의 유사도를 구하려고 한다면, 이 둘을 모두 평가한 사용자는, 재석, 명수, 세형, 광희다. 각각 (5, 1, 4, 4)와 (4, 1, 4, 3)이다.

$$similarity = \frac{5\times4+1\times1+4\times4+4\times3}{\sqrt{5^2+1^2+4^2+4^2}\ \sqrt{4^2+1^2+4^2+3^2}} = 0.99$$

공조와 라라랜드의 유사도는 0.99로 상당히 높은 유사도를 보인다. 이는 즉 공조를 좋아하는 사람은 라라랜드를 좋아할 확률이 높다는 말로 풀이 될수 있고, 그 반대로도 해석할 수 있다.

출처 : https://scvgoe.github.io/2017-02-01-%ED%98%91%EC%97%85-%ED%95%84%ED%84%B0%EB%A7%81-%EC%B6%94%EC%B2%9C-%EC%8B%9C%EC%8A%A4%ED%85%9C-(Collaborative-Filtering-Recommendation-System)/

0. Collaborative Filtering

고찰

- 두 아이템에 대해서 유사도를 구하려고 하는데, 이 두 영화를 모두 평가한 사람이 한 명 뿐인 경우에 유사도가 1이 나옴.
- 이렇게 적은 데이터에 기반한 정보는 추천의 정확도를 떨어뜨릴 가능성이 높음. 이러 경우를 방지하기 위해, 최소 평가 인원의 수를 정함. 두 아이템을 모두 평가한 사람의 수가 5명 이상일 때부터 유사도를 계산하기로 하고, 5명 미만의 공통 평가 인원수를 갖는 두 아이템에 대한 유사도를 0으로 처리.

출처 : https://scvgoe.github.io/2017-02-01-%ED%98%91%EC%97%85-%ED%95%84%ED%84%B0%EB%A7%81-%EC%B6%94%EC%B2%9C-%EC%8B%9C%EC%8A%A4%ED%85%9C-(Collaborative-Filtering-Recommendation-System)/

THANK YOU

YOUR NAME