Item-based CF

Making predictions

- Rationale for predicting the rating for item p for user u
  - Select the set of item neighbors based on similarity to item p
  - Consider the ratings of user u for all the neighbors i ($\boldsymbol{r_{u,i}}$)
  - Combine them together into a weighted average
  - Use the item neighbor similarity as the weight

$$pred(u,p) = \frac{\sum_{i \in ratedItem(u)} sim(i,p) * r_{u,i}}{\sum_{i \in ratedItem(u)} sim(i,p)}$$

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Comparing User-based and Item-based

- Item-based method often provide more relevant recommendations
- User-based method often provide more diverse recommendations
  - If the items are not diverse, then if the user does not like the first item, she might not also like the rest
- Item-based methods can provide a concrete reason for the recommendation
  - *Because you watched "Secrets of the Wings," [the recommendations are] <List>*

Strengths of Neighborhood-Based Methods

- Easy to implement and debug
- Easy to justify why a specific item is recommended
  - The interpretability of item-based methods is particularly notable
- The recommendations are relatively stable with the addition of new items and users
- It is also possible to create incremental approximations of these methods

Item-based CF

Making predictions for ranking task

- The goal is finding k most similar items to the ones that user u interacted with
  - Predicting the exact rating value user u might give to item p does not matter
  - Instead, a *relevance score* is predicted for user u for item p
  - Summing up the similarity values between item p and those user u interacted

Item-based CF

Making predictions for ranking task

UI rating matrix

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	0	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Item-Item similarity matrix

	Item1	Item2	Item3	Item4	Item5
Item1	0	0.1	0.4	0.6	0.2
Item2	0.1	0	0.2	0.3	0.5
Item3	0.4	0.2	0	0.3	0.5
Item4	0.6	0.3	0.3	0	0.4
Item5	0.2	0.5	0.5	0.4	0

Multiply

This operation computes weighted sum of similarity values between seen items and target item

To emphasize only similar items, *k* most similar items can be considered

Item-based CF

Making predictions for ranking task

UI rating matrix

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	0	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Item-Item similarity matrix

	Item1	Item2	Item3	Item4	Item5
Item1	0	0.1	0.4	0.6	0.2
Item2	0.1	0	0.2	0.3	0.5
Item3	0.4	0.2	0	0.3	0.5
Item4	0.6	0.3	0.3	0	0.4
Item5	0.2	0.5	0.5	0.4	0

Multiply

$$pred(u,p) = \sum_{j=0}^{|I|} R_{u,j} S_{j,p}$$

Item-based CF

Making predictions for ranking task

UI rating matrix

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	0	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Item-Item similarity matrix

	Item1	Item2	Item3	Item4	Item5
Item1	0	0.1	0.4	0.6	0.2
Item2	0.1	0	0.2	0.3	0.5
Item3	0.4	0.2	0	0.3	0.5
Item4	0.6	0.3	0.3	0	0.4
Item5	0.2	0.5	0.5	0.4	0

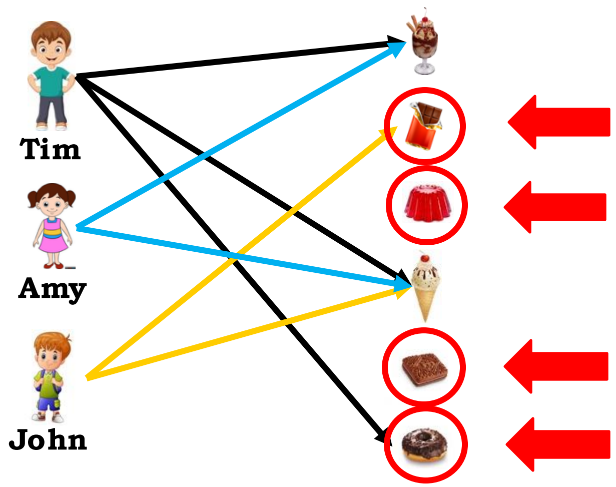
Multiply

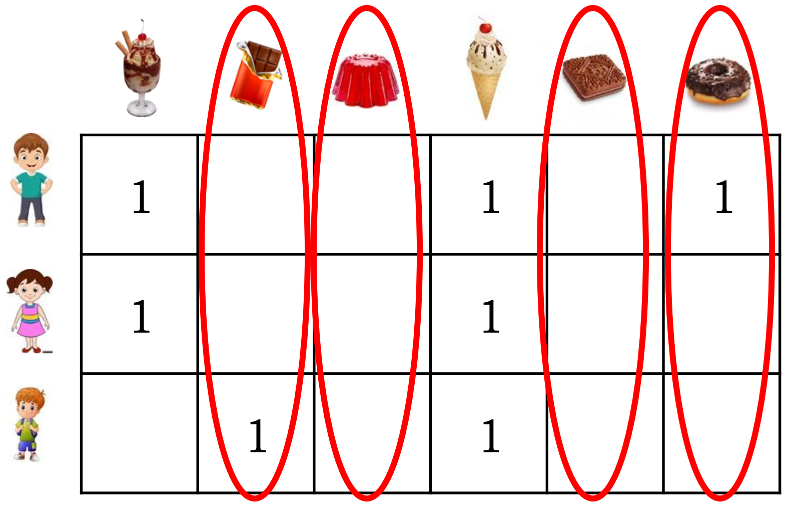
$$pred(u,p) = \sum_{j=0}^{|I|} R_{u,j} S_{j,p}$$

- This computation is done for all unseen items
- k items with the highest score form the recommendation list to user u

Limitations of Neighborhood-Based Methods

- Similarity computation is sometimes not accurate/possible due to *sparsity* issue
  - Sparsity refers to the percentage of *missing values* in UI matrix
  - When highly sparse, no neighbor can be found for an item
  - Therefore, the similarity computation and rating prediction is not possible



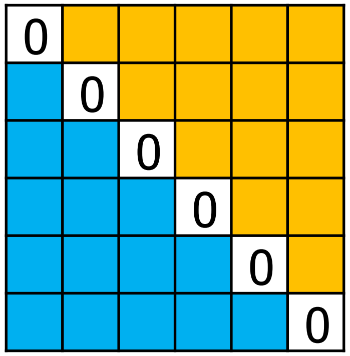


1			1		1
1			1		
	1		1		

Limitations of Neighborhood-Based Methods

- Creation of similarity matrix is computationally expensive
  - Similarity value needs to be computed beween all pairs of items
  - E.g., when number of items is 6:
    - Total entries: $6 \times 6 = 36$
    - Main diagonal entries are zero: $36 - 6 = 30$
    - Similarity matrix is *symmetry*: $\frac{30}{2} = 15$
  - Therefore, for n items, the number of required computation is

$$\frac{(n \times n) - n}{2}$$



  - *For 100,000 items, it requires 4,999,950,000 computations!*

SLIM algorithm

Ning and Karypis, ICDM 2011

- Sparse LInear Method
- SLIM improves item-based neighborhood model in creating item-item similarity matrix by addressing the aforementioned limitations
- Instead of computing similarity value between each pair of items, it learns the similarity matrix through an optimization process