Steps for Item Based Collaborative Filtering

Item-based collaborative filtering

n_{items}

n_{users}

	item_1	item_2	...	item_19	item_20	...	...
user_1	-	5	...	-	1	...	...
user_2	4	-	...	-	-	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
user_57	-	5	...	2	4	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...

$n_{users} \times n_{items}$

$\longrightarrow$

n_{items}

n_{items}

	item_1	item_2	...	item_19	item_20	...	...
item_1	1	0.23	...	-0.47	0.9	...	...
item_2	0.23	1	...	0.13	0.35	...	...
...	...	...	1	...	...	...	...
item_19	-0.47	0.13	...	1	0.01	...	...
item_20	0.9	0.35	...	0.01	1	...	...
...	...	...	...	...	...	1	...
...	...	...	...	...	...	...	1

$\boxed{n_{items} \times n_{items}}$

Item-based collaborative filtering

- Basic idea:
  - Use the similarity between items (and not users) to make predictions
- Example:
  - Look for items that are similar to Item5
  - Take Alice's ratings for these items to predict the rating for Item5

	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

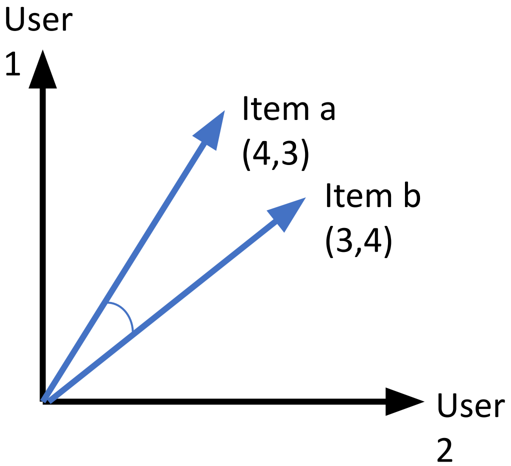
Analytics Vidhya

Similarity Measure

- Cosine Similarity Produces better results in item-to-item filtering
- Ratings are seen as vector in n-dimensional space
- Similarity is calculated based on the angle between the rating vectors

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

User	Item	Rating
User 1	Item a	3
User 2	Item a	4
User 1	Item b	4
User 2	Item b	3



$$\frac{(4,3).(3,4)}{\sqrt{4^2+3^2}*\sqrt{4^2+3^2}} = \frac{12}{25} = 0.48$$

Analytics Vidhya

Adjusted Cosine Similarity Measure

- Adjusted cosine similarity
  - take average user ratings into account, transform the original ratings
  - U: set of users who have rated both items a and b

$$sim(\vec{a},\vec{b}) = \frac{\sum_{u\in U}(r_{u,a})(r_{u,b})}{\sqrt{\sum_{u\in U}(r_{u,a})^2}\sqrt{\sum_{u\in U}(r_{u,b})^2}} \quad \Rightarrow \quad sim(\vec{a},\vec{b}) = \frac{\sum_{u\in U}(r_{u,a}-\bar{r_u})(r_{u,b}-\bar{r_u})}{\sqrt{\sum_{u\in U}(r_{u,a}-\bar{r_u})^2}\sqrt{\sum_{u\in U}(r_{u,b}-\bar{r_u})^2}}$$

Making predictions

- Prediction function:

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

- Neighborhood size is typically also limited to a specific size

- Not all neighbors are taken into account for the prediction

Making predictions

	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

sim = 0.99 sim = 0.94

$$pred(u,p) = \frac{\sum_{i \in ratedItem(u)} sim(i,p) * r_{u,i}}{\sum_{i \in ratedItem(u)} sim(i,p)} = \frac{0.99*5+0.94*4}{0.99+0.94} = 4.51$$

Data sparsity problems

- Cold start problem
  - How to recommend new items? What to recommend to new users?
- Straightforward approaches
  - Ask/force users to rate a set of items
  - Use another method (e.g., content-based, demographic or simply non-personalized) initially
- Alternatives
  - Use better algorithms (beyond nearest-neighbor approaches)
  - Example:
    - In nearest-neighbor approaches, the set of sufficiently similar neighbors might be too small to make good predictions

Memory-based and model-based approaches

- User-based CF is said to be "memory-based"
  - the rating matrix is directly used to find neighbors / make predictions
  - does not scale for most real-world scenarios
  - large e-commerce sites have tens of millions of customers and millions of items
- Model-based approaches
  - based on some pre-processing or "model-learning" phase
  - at run-time, only the learned model is used to make predictions
  - models are updated / re-trained periodically
  - *item*-based CF is an example for model-based approaches

Item based collaborative Filtering for Unary Ratings

- So far, we have seen how to use item-item over rating data
- This also works well for unary data – mostly implicit ratings such as clicks, song plays, purchases etc.
- Matrix to represent data
  - Logical (1/0) user-item purchase matrix
  - Purchase Count Matrix
- Standard Mean Centering is not useful here as we have 1s and 0s
- Solution: Normalise User Vectors to unit vectors
- Weighted Average doesn't work for Unary Data so just sum the neighbour similarities