CS 768: Learning With Graphs **Autumn 2020-2021**

Lecture 11: Collaborative Filtering

Instructor: Prof. Abir De *Scribe: Soumya Chatterjee*

11.1 Recommendation Systems

11.1.1 Formal Model

- C is the set of customers
- S is the set of items
- Utility function $u: C \times S \to R$
  - R is the set of ratings
  - R is totally ordered
  - Eg. 0-5 star ratings, real number in $[0, 1]$

11.1.1.1 Utility Matrix

	Avatar	LOTR	Matrix	Pirates
Alice	1		0.2	
Bob		0.5		0.3
Carol	0.2		1	
David				0.4

Utility Matrix are generally sparse. Recommender systems try to predict these missing values.

11.1.2 Key Problems

1. Gathering known ratings for utility matrix
   - Explicit - Ask people to rate items. Is simple and gives good quality data but does not scale. Most people do not leave ratings or reviews.
   - Implicit - Learn ratings from user actions. Is much more scalable but knowing low ratings is difficult with implicit methods
   - Generally a combination is used
2. Extrapolate unknown ratings from the known ones. Mainly the high rating
   - Key problem here is the fact that U is sparse
   - There is also a cold start problem with new users/items
3. Evaluating extrapolation methods

11.2 Content-based Recommender Systems

Main idea: Recommend items to customer x similar to previous items rated highly by x

Item Profile: For each item, an item profile is created which is a set of features for that item. For example, for recommending news articles, TF-IDF features can be used.

TF-IDF

Let f_{ij} be the frequency of term i in document j. The term frequecy TF_{ij} is given by

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$$

Now, if n_i documents contain the term i and N is the total number of documents, the inverse document frequency of term i is given by

$$IDF_i = \log \frac{N}{n_i}$$

Using these, the TF-IDF score of term i in document j is given by

$$w_{ij} = TF_{ij} \times IDF_i$$

User Profiles: can be constructed by using a weighted average of the item profiles of the items the user has rated where the weight is the rating given to that item. A variant of this is to mean normalize the weights by the average rating of the user.

Making predictions: The similarity between user with user profile $\mathbf{x}$ and item with profile $\mathbf{i}$ is given by the cosine similarity $U(\mathbf{x}, \mathbf{i}) = \frac{\mathbf{x} \cdot \mathbf{i}}{|\mathbf{x}|\,|\mathbf{i}|}$. For an user, find the cosine similarity with all products and recommend those with high similarity.

Pros: (1) Data from other users not needed; (2) Able to recommend to users with unique tastes; (3) Able to recommend new and unpopular items; (4) Can provide explanations for recommendations

Cons: (1) Finding appropriate features is hard; (2) Overspecialization - never recommends items outside user's profile; (3) Cold-start for new users - no user profile for new users

11.3 Collaborative Filtering

11.3.1 Main Idea

Consider user x who has rated some items. Now, find the set N of top-k other users whose ratings are *similar* to x's ratings. x's ratings for other items can be estimated based on rating of users in N.

11.3.2 Similarity between users

For doing this, we need to define similarity $\text{sim}(x, y)$ between users x and y with rating vectors r_x and r_y. The rating vector r_x contains the ratings given by user x to all the items in the catalog.

We want sim to capture the intuition that if two users have similar ratings for some items, they will have higher sim values than two user with dissimilar rating for some items. Choices for sim function are as follows:

- Jaccard Similarity: $\text{sim}(A, B) = \frac{|r_A \cap r_B|}{|r_A \cup r_B|}$
  The problem with this is that it completely ignores ratings
- Cosine similarity: $\text{sim}(A, B) = \cos(r_A, r_b)$
  To calculate this, we need to fill in missing values in the rating vectors. We can fill all of them with 0. But this leads to the following issue.
  Consider that all the rating vectors have values in $[0, 5]$ (say). Now, since rating vectors are generally sparse, any pair of vectors will have lots of zeros and thus, very close sim values.
- Centered cosine or Pearson Correlation:
  Change the 0-filled rating vectors in cosine similarity by normalizing by subtracting means from each rating vector. Thus the average rating of each user is made 0.
  This definition captures the intuition better by treating missing ratings as *average* and not negative.
  It also helps is handling differences between *tough raters* and *easy raters* via the mean normalization.

11.3.3 Making Rating Predictions

Let r_x be the rating vector of user x. Say we want to predict the rating user x give to item i. For this consider the set N of k users most similar to x who have rated item i. The prediction for user x's rating for item i can given by

$$r_{xi} = \frac{\sum_{y \in N} \text{sim}(x, y) \cdot r_{yi}}{\sum_{y \in N} \text{sim}(x, y)}$$

11.3.4 Another view: Item-Item Collaborative Filtering

The above discussed method is called User-User Collaborative Filtering. A dual method is called Item-Item Collaborative Filtering in which, for predicting the rating user x will give to item i, we find the set $N(i; x)$ of top k other similar items rated by x and then estimate rating for item i based on ratings for similar items *i.e.*

$$r_{xi} = \frac{\sum_{j \in N(i;x)} \text{sim}(i, j) \cdot r_{xj}}{\sum_{j \in N(i;x)} \text{sim}(i, j)}$$

Here the feature vector for item i is the collection of ratings given by different users to this item.

11.3.5 Item-Item vs User-User

In theory, user-user and item-item are dual approaches but in practice, item-item outperforms user-user in many use cases.

This is because items are *simpler* than users. Items belong to to a small set of genres but users can have varied tastes across genres. Thus item similarity is more meaningful than user similarity.

11.3.6 Complexity

Most expensive step is finding k most similar users (or items). For each user this is $O(|U|)$ where U is the utility matrix. We could precompute the neighbours N for each user but this too takes a $O(k \cdot |U|)$ time which is again quite expensive.

This can be avoided by doing near-neighbour search in high dimensions (Locality Sensitive Hashing), or clustering and then finding neighbours within clusters, or dimensionality reduction.

11.3.7 Pros and Cons

Pros: (1) Works for any kind of item since no feature selection is needed

Cons:

(1) Cold start: Need enough users in the system to find a match

(2) Sparsity: Since the user/ratings matrix is sparse, it is hard to find users who have rated the same items

(3) First rater problem: Cannot recommend an unrated item (eg. new items and esoteric items)

(4) Popularity bias: Tends to recommend popular items since they will occur in neighbourhoods of many items

11.4 Hybrid Methods

- Add content-based methods to collaborative filtering: Item profiles can be used to overcome new item problem, and demographics can be used to deal with new user problem
- Implement two or more different recommenders and combine predictions (possibly using a linear model). *E.g.*, global baseline + collaborative filtering

11.4.1 Global Baseline Estimate

Suppose that user x has rated no item similar to item i. So N is empty and no prediction can be made about x's rating for i using collaborative filtering.

We could however predict x's rating for i as follows:

Let the mean rating of all the items be μ. Let the rating deviation of user x be b_x (avg. rating given by user x - μ) and rating deviation of item i be b_i. We could estimate user x's rating of i as

$$b_{xi} = \mu + b_x + b_i$$

11.4.2 Combining Global Baseline with Collaborative Filtering

This method uses a global baseline estimate with a collaborative filtering refinement term as follows

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} \text{sim}(i,j) \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} \text{sim}(i,j)}$$