HOP-Rec: High-Order Proximity for Implicit Recommendation

Radek Bartyzal

Let's talk ML in Prague

18. 10. 2018

Recommender systems

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1m} \\ r_{21} & r_{22} & r_{23} & \dots & r_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{n1} & r_{n2} & r_{n3} & \dots & r_{nm} \end{pmatrix}$$

Figure: User-item matrix of rating values.

Recommendation algorithm types:

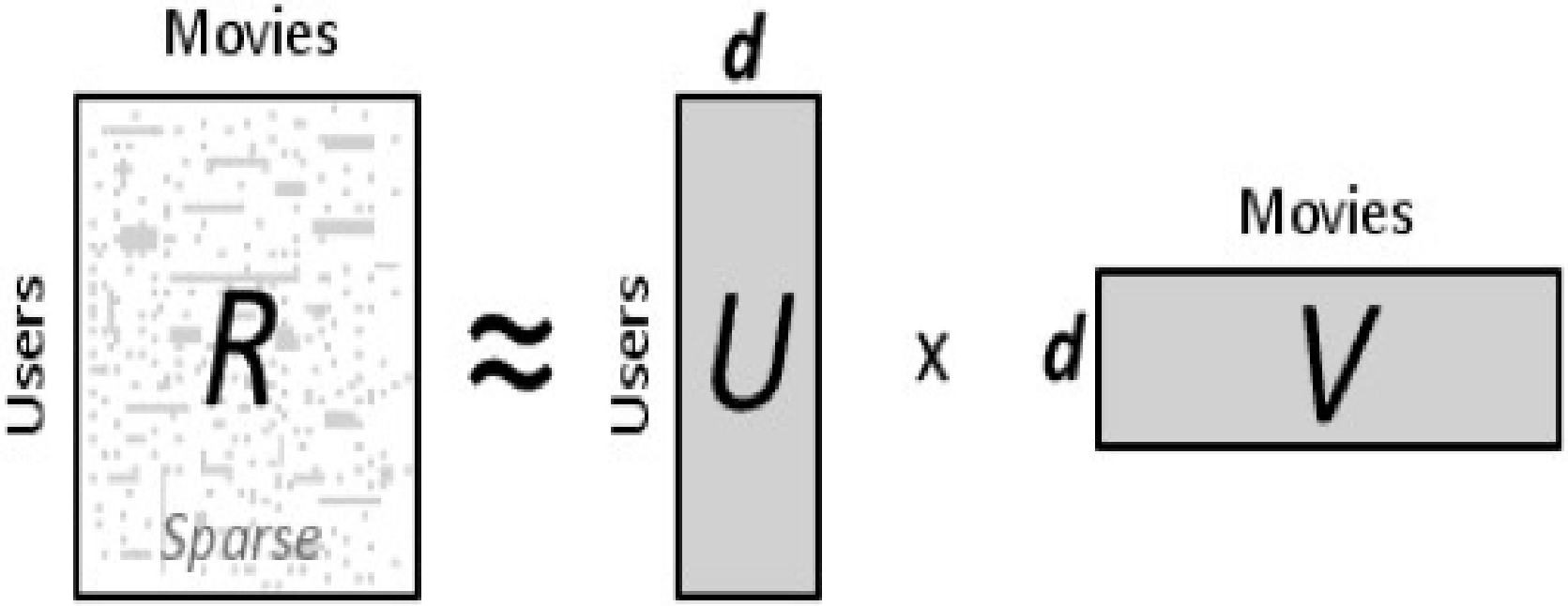
- Factorization based
- Graph based
- Other

Algorithm types

- **Factorization based:** Get lower dimensional vector representation (embedding) of each user and each item. Then use these embeddings as an input to other recommendation algorithms.

- **Graph based:** Construct a bipartite graph from the rating matrix. Ratings are edge weights. Traverse graph in certain ways to get user preferences.

Matrix factorization



$$Loss = \left\| R - UV^T \right\|^2 + \lambda \|U\|^2 + \lambda \|V\|^2$$

HOP-Rec

- combine factorized with graph based
- use implicit ratings $= 1$ or unknown

HOP-Rec

Graph:

- nodes = users $\cup$ items
- edges = known implicit ratings, no weights

Random walk through graph:

1. start at user $u = u_0$
2. go through $k-1$ items and $k-1$ users to end at item i_k
3. resulting sequence $S = (u_0, i_1, u_1, \cdots, i_{k-1}, u_{k-1}, i_k)$
4. K typically low = 1 - 3

HOP-Rec

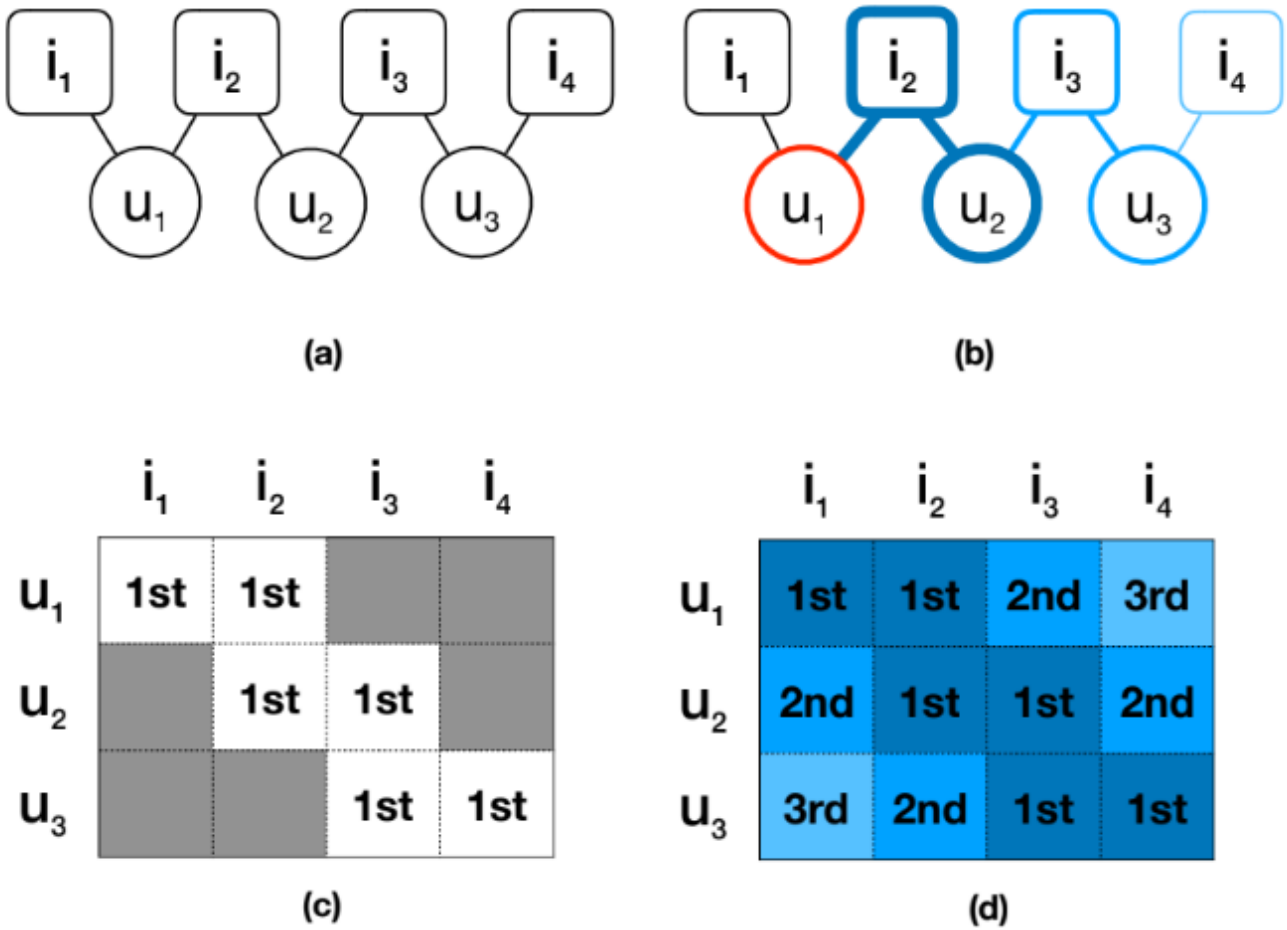


Figure: High-order proximity between users and items within observed interactions [1].

HOP-Rec

$$\mathcal{L}_{HOP} = \sum_{\substack{1 \le k \le K \\ u,(i,i')}} \overbrace{C(k)\, \mathbb{E}_{\substack{i \sim P_u^k \\ i' \sim P_N}}}^{\text{graph model}} \overbrace{\left[\mathcal{F}\left(\theta_u^\intercal \theta_{i'}, \theta_u^\intercal \theta_i\right)\right]}^{\text{factorization model}} + \lambda_\Theta \|\Theta\|_2^2$$

P_u^k = k-order probability distribution for an item sampled from Su
P_N = uniform distribution over all items = i' is random item
$C(k) = \frac{1}{k}$ = weight = how close is the item to the user in graph

$$\mathcal{F}(\theta_u^\intercal \theta_{i'}, \theta_u^\intercal \theta_i) = \mathbb{1}_{\{\theta_u^\intercal \theta_{i'} - \theta_u^\intercal \theta_i > \epsilon_k\}} \log\left[\sigma\left(\theta_u^\intercal \theta_{i'} - \theta_u^\intercal \theta_i\right)\right]$$

Figure: Ranking loss = indicator * pairwise logistic loss [1].

Indicator = if predicted rating of random item is higher than for the known item, then optimize their embeddings to change that

Training

1. Do this for each user u:
2. sample item i from random walk
3. sample random item i'
4. optimize embeddings of u, i, i' by Asynchronous SGD to predict higher rating for (u, i) than for (u, i')

Whats the point?
Instead of giving SGD just the known ratings and then all the rest as unknown we increase the number of 'known' examples by selecting higher-order neighbor items and present them to SGD with lower weight than the original first-order known ratings.

Results

	CiteUlike			MovieLens-1M		
	P@10	R@10	MAP@10	P@10	R@10	MAP@10
MF	4.1%	13.1%	6.7%	17.7%	13.1%	11.7%
BPR	3.8%	14.2%	6.4%	18.1%	13.2%	12.5%
WARP	5.4%	18.3%	9.1%	24.8%	18.5%	18.5%
K-OS	5.6%	19.4%	9.5%	23.0%	17.3%	16.4%
$RP^3(\beta)$	5.9%	21.2%	3.2%	22.8%	17.2%	14.2%
HOP	5.9%	**21.3%**	***10.8%**	***25.9%**	***20.5%**	***19.6%**
%Improv.	0.0%	0.5%	13.7%	4.4%	10.8%	5.9%

Results

MovieLens-20M			Amazon-Book		
P@10	R@10	MAP@10	P@10	R@10	MAP@10
14.9%	14.0%	11.3%	0.7%	3.7%	1.4%
13.3%	14.3%	10.4%	1.0%	5.3%	2.5%
20.7%	21.4%	17.2%	1.4%	7.6%	3.2%
19.6%	20.5%	15.7%	1.5%	7.9%	3.5%
17.3%	19.4%	10.3%	-	-	-
*21.2%	*22.3%	*17.9%	1.5%	7.9%	*3.6%
2.4%	4.2%	4.1%	0.0%	0.0%	2.9%

Sources

1. Yang, Jheng-Hong, et al. "HOP-rec: high-order proximity for implicit recommendation." Proceedings of the 12th ACM Conference on Recommender Systems. ACM, 2018.