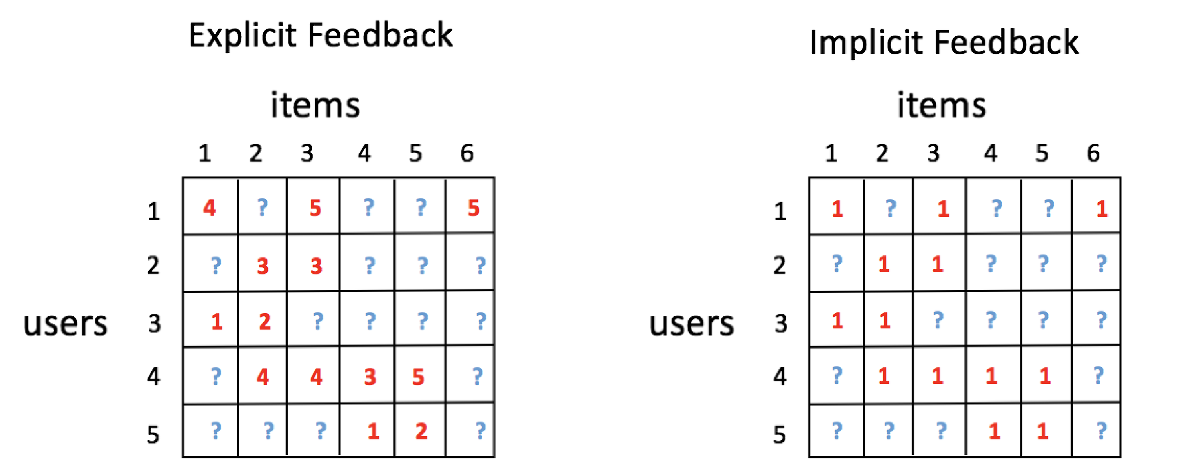
SQL-Rank: A Listwise Approach to Collaborative Ranking

Liwei Wu, Cho-Jui Hsieh, and James Sharpnack
{ liwu, chohsieh and jsharpna}@ucdavis.edu

Problem

- Collaborative Ranking
  - Recommender system problem
  - Focus on ranking of items rather than ratings in the model
  - Performance measured by ranking order of top k items for each user
- State-of-arts are using pairwise loss (such as BPR, and Primal-CR++).
- **But pairwise loss is not the only ranking loss.**
- We will show a new listwise loss works better than pairwise loss in collaborative ranking with implicit feedback.



Stochastic Queuing (SQ)

We denote the set of valid permutations as $\mathcal{S}(R,\Omega)$, where Ω is the set of all pairs (i,j) such that $R_{i,j}$ is observed.

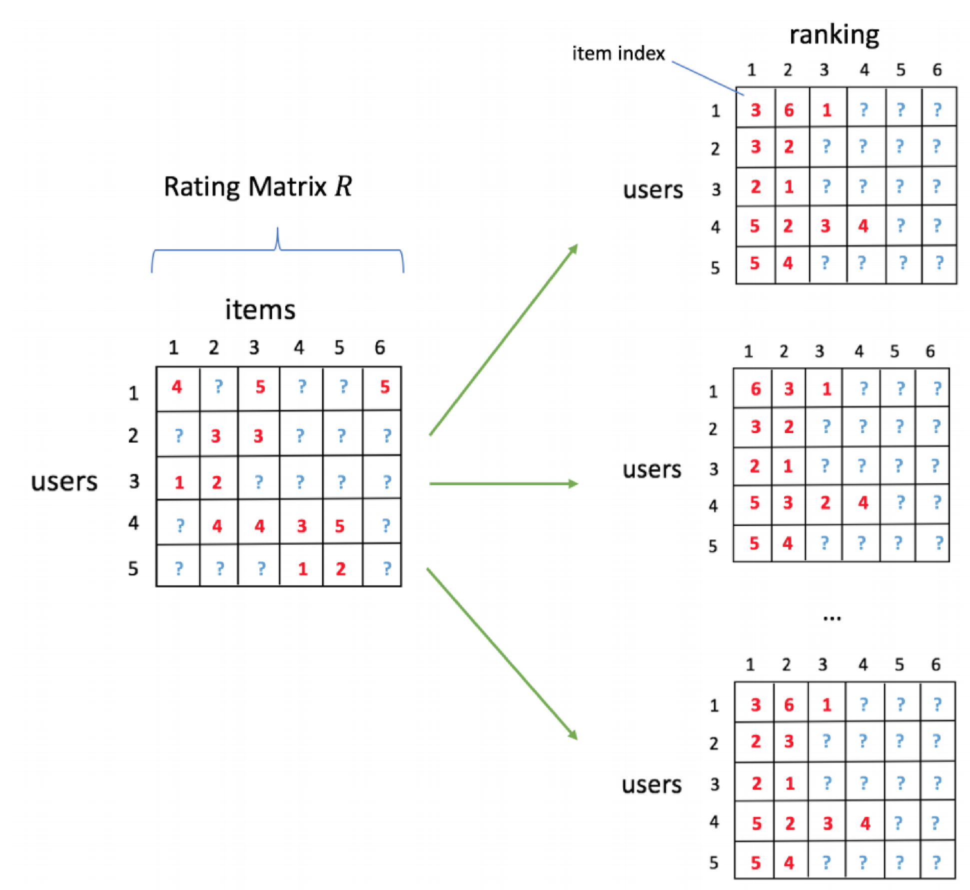


Figure 1. Demonstration of Stochastic Queuing Process—the rating matrix R (left) generates multiple possible rankings Π's (right), $\Pi \in \mathcal{S}(R,\Omega)$ by breaking ties randomly.

Model

- Permutation probability for a single user's top k ranked items:

$$P_s^{(k,\bar{m})}(\pi) = \prod_{j=1}^{\min\{k,\bar{m}\}} \frac{\phi(s_{\pi_j})}{\sum_{l=j}^{\bar{m}} \phi(s_{\pi_l})}.$$

, where π is a particular permutation (or ordering) of the m items, s are underlying true scores for all items, and ϕ is some increasing function.
- Can easily be extended to multiple users even with a lot of ties (e.g. 0/1 implicit feedback data).
- Minimize the negative log-likelihood:

$$\min_{X \in \mathcal{X}} -\log \sum_{\Pi \in \mathcal{S}(R,\Omega)} P_X^{(k,\bar{m})}(\Pi)$$

- The non-convex version can easily be optimized using SGD:

$$\sum_{\Pi \in \mathcal{S}(R,\Omega)} \underbrace{-\sum_{i=1}^{n} \sum_{j=1}^{\bar{m}} \log \frac{\phi(u_i^T v_{\Pi_{ij}})}{\sum_{l=j}^{\bar{m}} \phi(u_i^T v_{\Pi_{il}})}}_{f(U,V)} + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2),$$

$g = \log \phi$ is the sigmoid function.
- For implicit feedback data, we sample $\rho\tilde{m}$ unobserved entries uniformly and append to the back of the list $\rightarrow \bar{m} = (1+\rho)\tilde{m}$ (For each user (row of R), assume there are $\tilde{m}$ 1's).

Theory

The problem of the constrained form

$$\hat{X} := \arg\min -\log P_X(\Pi) \text{ such that } X \in \mathcal{X}. \quad (1)$$

The personalized setting:

$$X_{ij} = u_i^\top v_j, u_i, v_j \in \mathbb{R}^r, \|U\|_F \le c_u, \|V\|_F \le c_v. \quad (2)$$

Corollary 1. *Consider the minimizer, $\hat{X}$, to the constrained optimization, (1). Suppose that there exists a $X^\star \in \mathcal{X}$ such that $\Pi_i \sim P_{X_i^\star}$ independently for all $i = 1, \ldots, n$. If $\log\phi$ is 1-Lipschitz, then in the personalized ranking setting, (2), the KL-divergence between the estimate and the truth is bounded:*

$$D(X^\star, \hat{X}) = O_{\mathbb{P}}\left(\sqrt{\frac{rm}{n}} \ln m\right).$$

Results

- Baseline Methods
  - Explicit methods:
    - LIST-MF (Shi et al., 2010): another listwise method which utilizes the cross entropy loss
    - Primal-CR++ (Wu et al., 2017): our newly proposed pairwise method
    - MF (Koren, 2008): classical matrix factorization method
  - Implicit methods:
    - Weighted-MF (Hu et al., 2008): the weighted matrix factorization algorithm by putting different weights on 0 and 1's
    - BPR (Rendle et al., 2009): the Bayesian personalized ranking method motivated by MLE
- Experimental results

Dataset	Method	NDCG@10	P@1	P@5	P@10
Movielens1m	SQL-Rank	0.75076	**0.50736**	**0.43692**	**0.40248**
	List-MF	0.73307	0.45226	0.40482	0.38958
	Primal-CR++	**0.76826**	0.49365	0.43098	0.39779
	MF	0.74661	0.00050	0.00096	0.00134
Yahoo music	SQL-Rank	0.66150	**0.14983**	**0.12144**	**0.10192**
	List-MF	0.67490	0.12646	0.11301	0.09865
	Primal-CR++	0.66420	0.14291	0.10787	0.09104
	MF	**0.69916**	0.04944	0.03105	0.04787

Explicit Feedback Methods

Dataset	Method	P@1	P@5	P@10
Movielens1m	SQL-Rank	**0.73685**	**0.67167**	0.61833
	Weighted-MF	0.54686	0.49423	0.46123
	BPR	0.69951	0.65608	**0.62494**
Amazon	SQL-Rank	0.04255	**0.02978**	**0.02158**
	Weighted-MF	0.03647	0.02492	0.01914
	BPR	**0.04863**	0.01762	0.01306
Yahoo music	SQL-Rank	**0.45512**	**0.36137**	**0.30689**
	Weighted-MF	0.39075	0.31024	0.27008
	BPR	0.37624	0.32184	0.28105
Foursquare	SQL-Rank	**0.05825**	**0.01941**	**0.01699**
	Weighted-MF	0.02184	0.01553	0.01407
	BPR	0.03398	0.01796	0.01359

Implicit Feedback Methods

Method	P@1	P@5	P@10
With SQ	**0.73685**	**0.67167**	**0.61833**
Without SQ	0.62763	0.58420	0.55036

Effectiveness of SQ

- We find that our method achieves better precision results on most explicit or implicit datasets.
- We find that stochastic queuing (SQ) greatly improves recommendation results.

Conclusion

- We propose a new collaborative filtering algorithm using listwise loss.
- Our algorithm is faster and more accurate than the state-of-the-art methods on implicit feedback data.
- We provide a theoretical framework for analyzing listwise methods.

Source Code

- Julia codes: https://github.com/wuliwei9278/SQL-Rank