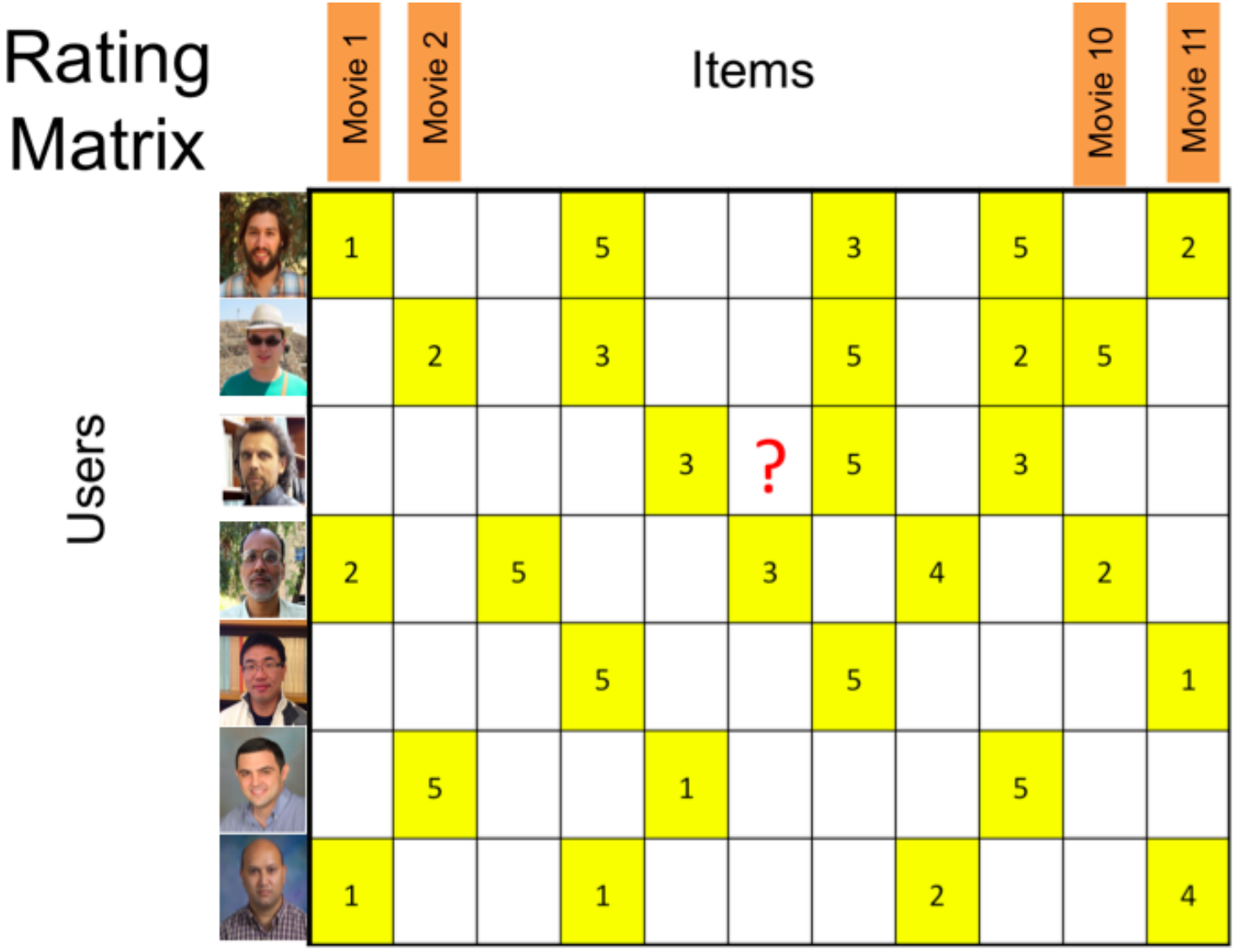
SQL-Rank: A Listwise Approach to Collaborative Ranking

Liwei Wu
Depts of Statistics and Computer Science
UC Davis

ICML'18, Stockholm, Sweden
July 10-15, 2017

Joint work with Cho-Jui Hsieh and James Sharpnack

Recommender Systems: Netflix Problem



Matrix Factorization Approach $R \approx WH^T$

H^T

-0.07	-0.11	-0.53	-0.46	-0.06	-0.05	-0.53	-0.07	-0.35	-0.19	-0.14
0.13	-0.42	0.45	0.17	-0.25	-0.17	-0.18	0.27	-0.59	0.05	0.14
-0.21	-0.43	-0.23	0.16	0.08	0.17	0.57	-0.39	-0.37	-0.08	-0.15

W

-8.72	0.03	-1.03
-7.56	-0.79	0.62
-4.07	-3.95	2.55
-3.52	3.73	-3.32
-7.78	2.34	2.33
-2.44	-5.29	-3.92
-1.78	1.90	-1.68

1			5			3		5		2
	2		3			5		2	5	
				3	?	5		3		
2		5			3		4		2	
			5			5				1
	5			1				5		
1			1				2			4

Collaborative Filtering: Matrix Factorization Approach

Latent Factor model fit the ratings directly in the objective function:

$$\min_{\substack{W\in\mathbb{R}^{n\times r}\\ H\in\mathbb{R}^{m\times r}}} \sum_{(i,j)\in\Omega} (R_{ij} - \boldsymbol{w}_i^T \boldsymbol{h}_j)^2 + \frac{\lambda}{2}\left(\|W\|_F^2 + \|H\|_F^2\right),$$

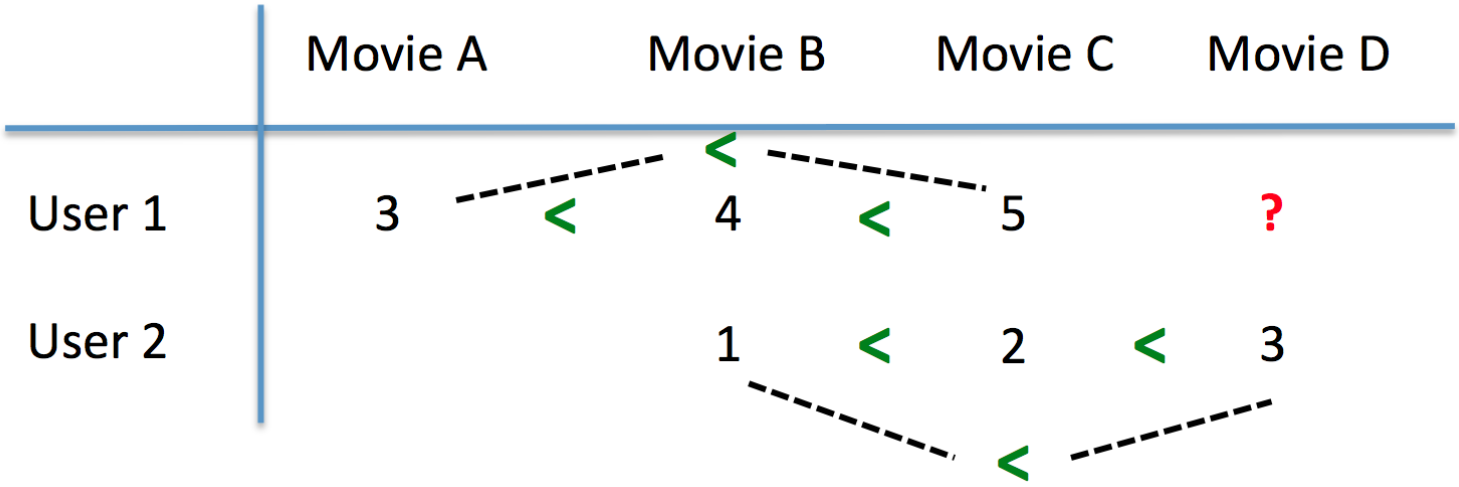
- $\Omega = \{(i,j) \mid R_{ij} \text{ is observed}\}$
- Regularized terms to avoid over-fitting

Criticisms for Pointwise Methods:

- Calibration drawback: users have different standards for their ratings
- Performance measured by quality of rating prediction, not the ranking of items for each user

Collaborative Filtering: Collaborative Ranking Approach

- 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++)

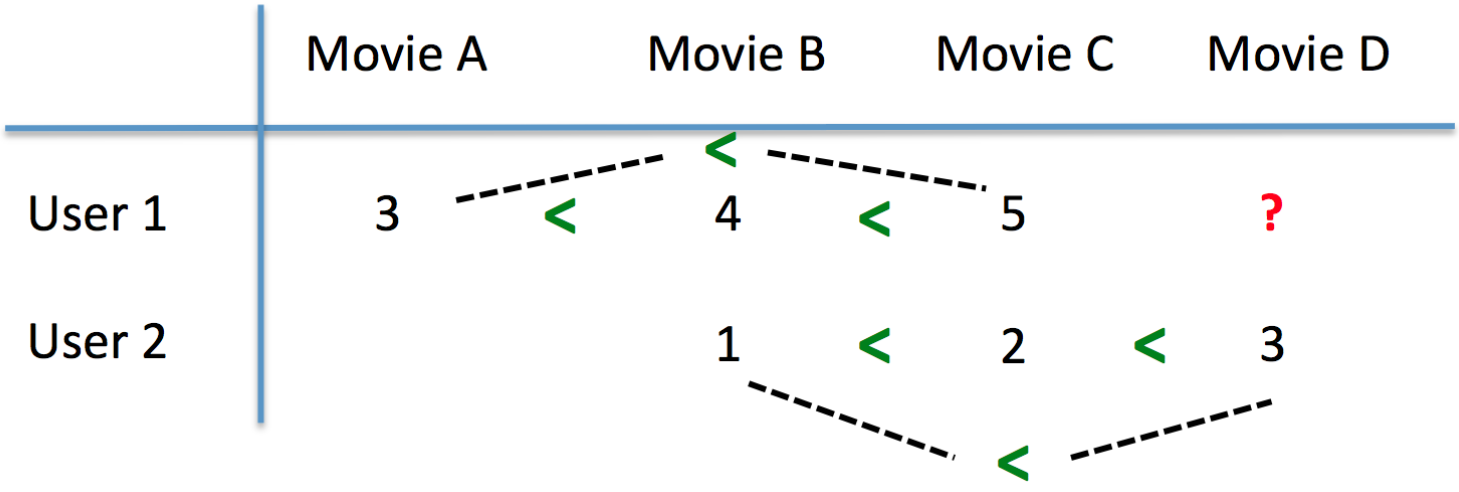


- With the same data size, **ranking loss outperforms point-wise loss**

  But pairwise loss is not the only ranking loss.

Collaborative Filtering: Collaborative Ranking Approach

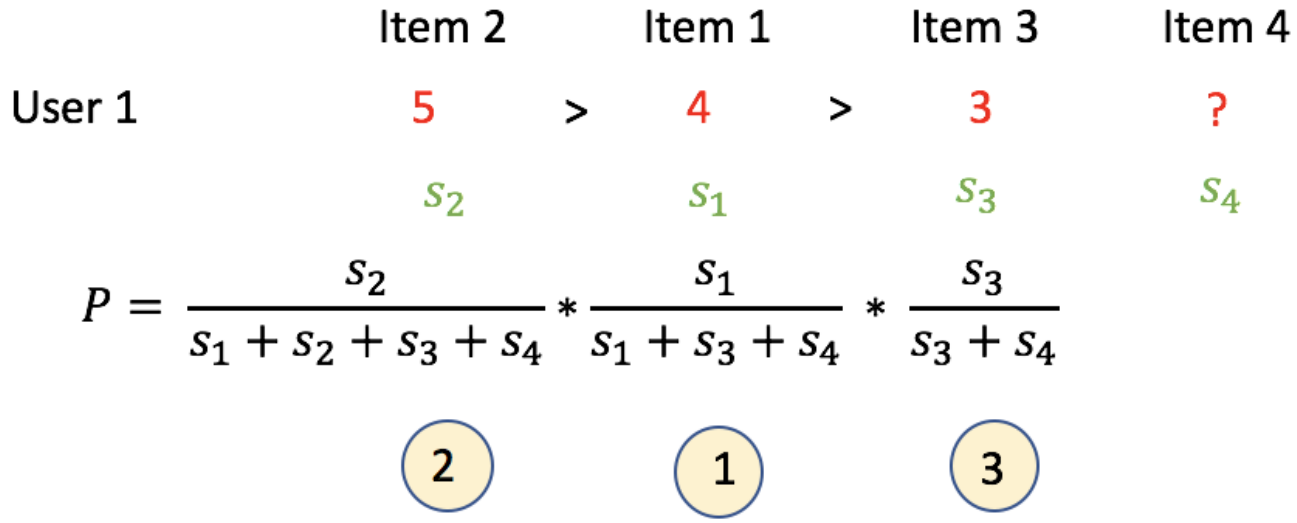
- Focus on ranking of items rather than ratings in the model
- Performance measured by ranking order of top k items for each user
- State-of-arts are using pairwise loss (such as BPR and Primal-CR++)



- With the same data size, **ranking loss outperforms point-wise loss**

  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.

Collaborative Ranking: Listwise Loss



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

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

where π is a particular permutation (or ordering) of the $\bar{m}$ items, s are underlying true scores for all items, and φ is some increasing function.

SQL-Rank: Stochastically Queuing

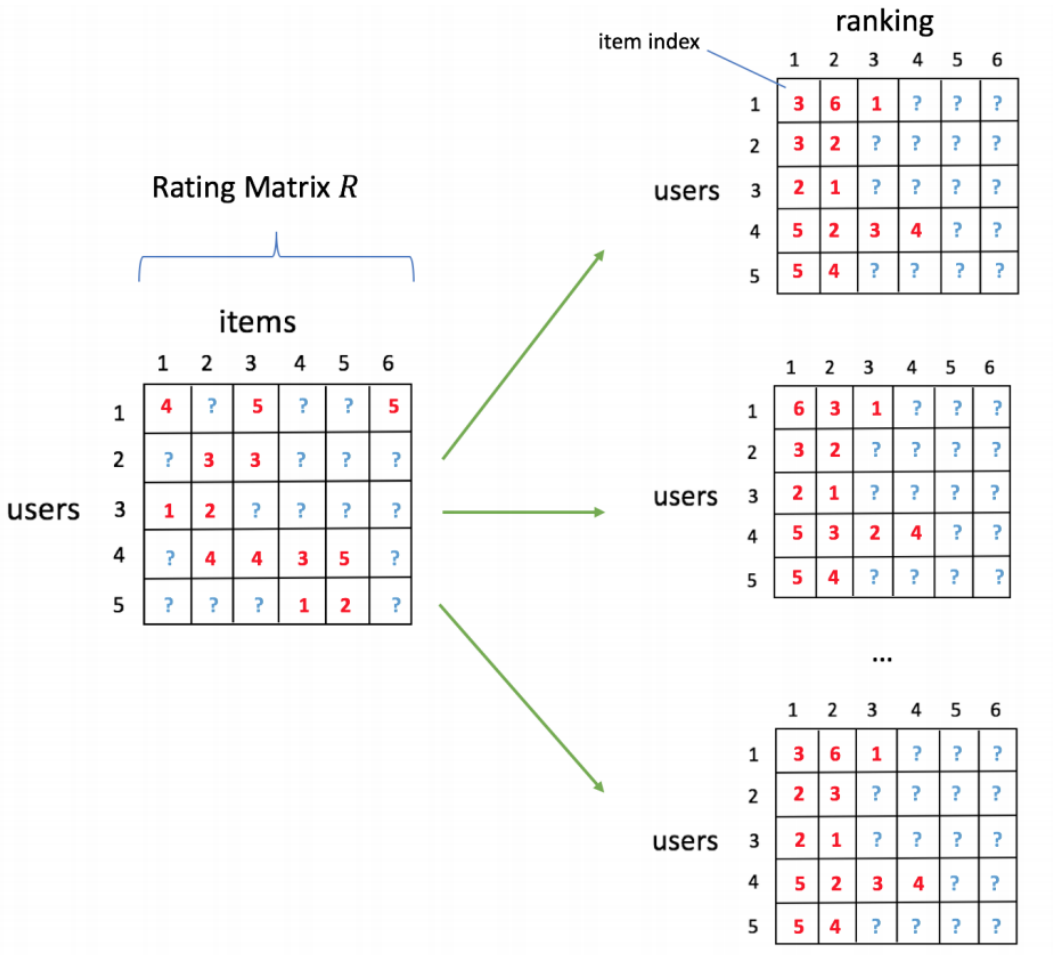


Figure 1. Demonstration of Stochastic Queuing Process—the rating matrix R (left) generates multiple possible rankings Π's (right), $\Pi \in \mathcal{S}(R, \Omega)$ by breaking ties randomly.

Explicit Feedback versus Implicit Feedback

Explicit Feedback

users \ items	1	2	3	4	5	6
1	4	?	5	?	?	5
2	?	3	3	?	?	?
3	1	2	?	?	?	?
4	?	4	4	3	5	?
5	?	?	?	1	2	?

Implicit Feedback

users \ items	1	2	3	4	5	6
1	1	?	1	?	?	1
2	?	1	1	?	?	?
3	1	1	?	?	?	?
4	?	1	1	1	1	?
5	?	?	?	1	1	?

SQL-Rank: Objective Function

- 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:

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

  $g = \log \varphi$ 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).

Experiments: 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

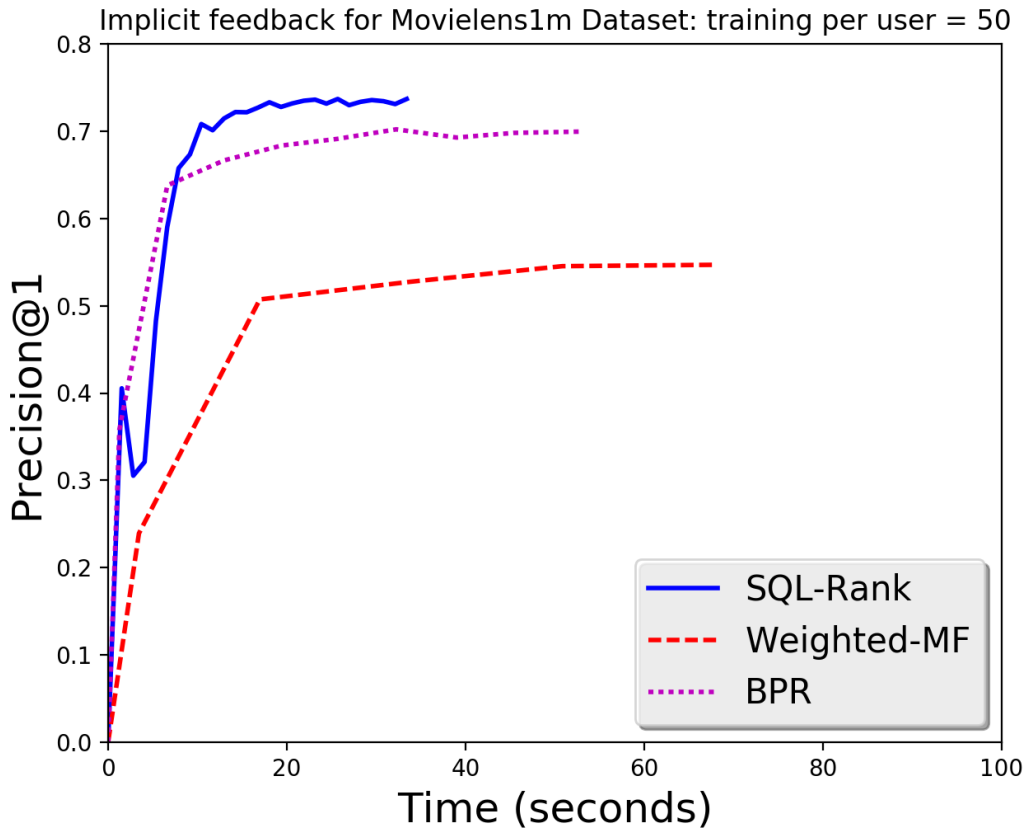
Experiments: Implicit Feedback

- Comparing implicit feedback methods on various datasets.

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

Experiments: Training Time

- Our algorithm has time complexity of $O(nnz \cdot r)$, which is linear to the observed ratings (nnz is the number of nonzero elements in the rating matrix and r is the rank).



Experiments: Effectiveness of Stochastic Queuing

- Our way of handling ties and missing data is very effective and improves the ranking results by a lot.

Table: Effectiveness of Stochastic Queuing Process on Movielen1m dataset.

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

Experiments: Effectiveness of using the Full List

- Using permutation probability for full lists is better than using the top k probability for top k partial lists in the likelihood loss.

Table: Comparing different k on Movielens1m data set using 50 training data per user.

k	NDCG@10	P@1	P@5	P@10
5	0.64807	0.39156	0.33591	0.29855
10	0.67746	0.43118	0.34220	0.33339
25	0.74589	0.47003	0.42874	0.39796
50 (FULL LIST)	**0.75076**	**0.50736**	**0.43692**	**0.40248**

Conclusions

- 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.
- Relationship with production recommender systems?

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

- Our poster: Poster 51 in hall B, starting tonight at 6:15pm.

Thank You!