λOpt: Learn to Regularize Recommender Models in Finer Levels

Yihong Chen†, Bei Chen‡, Xiangnan He*, Chen Gao†, Yong Li†, Jian-Guang Lou‡, Yue Wang†

† Tsinghua University, ‡ Microsoft Research, * University of Science and Technology of China

Matrix Factorization for Recommendation

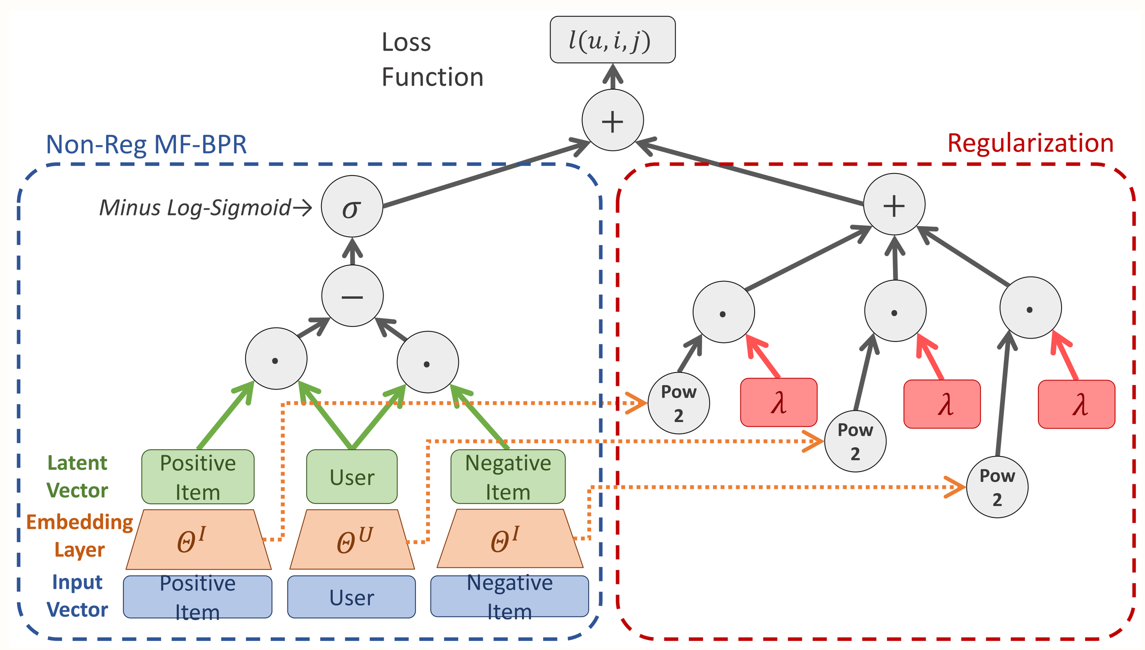

Figure 1: Conventional MF with Bayesian Personalized Ranking (BPR) criterion.

Regularization Tuning Headache

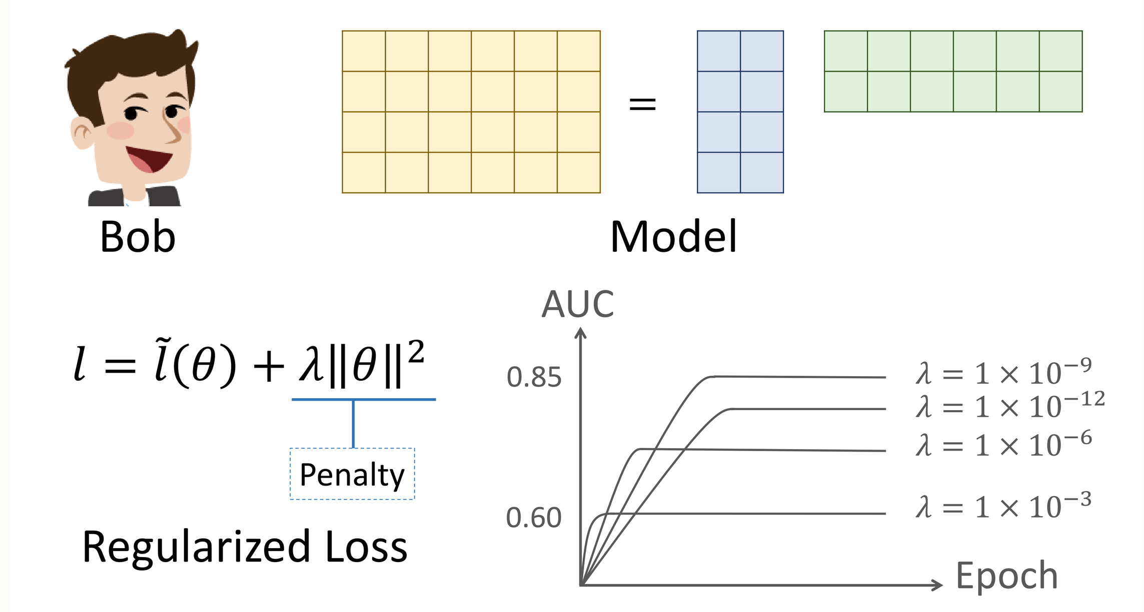

Figure 2: The model can be highly sensitive to the choice of λ.

Our Goal: Find the reasons behind the regularization tuning headache and design methods to automatically regularize recommender models within appropriate computation cost.

Why Hard to Tune the Recommender Models?

Hypothesis 1: Compromise on Regularization Granularity

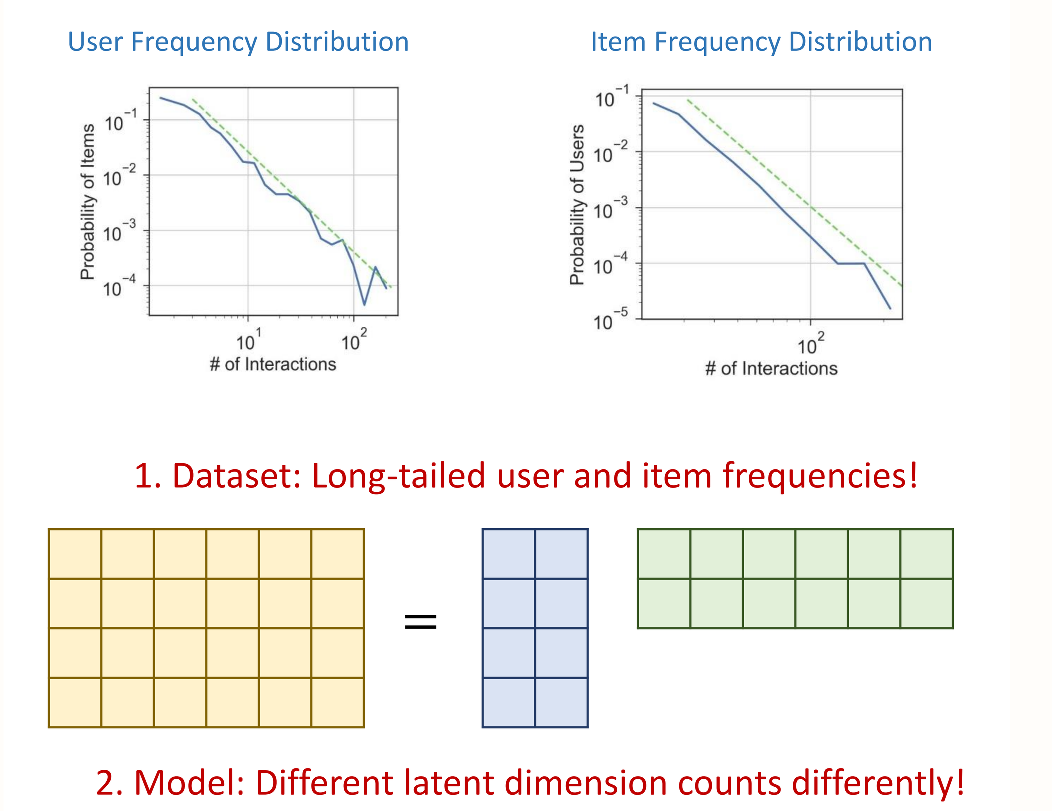

Figure 3: Due to the characteristics of the models and datasets, fine-grained regularization often works better.

Typically, we use **grid-search** or **babbysitting** to determine λ. In such cases, we set a global λ instead of fine-grained λ as it would otherwise take unaffordable effort or computation cost.

Hypothesis 2: Fixed Regularization Strength Throughout the Model Training Process

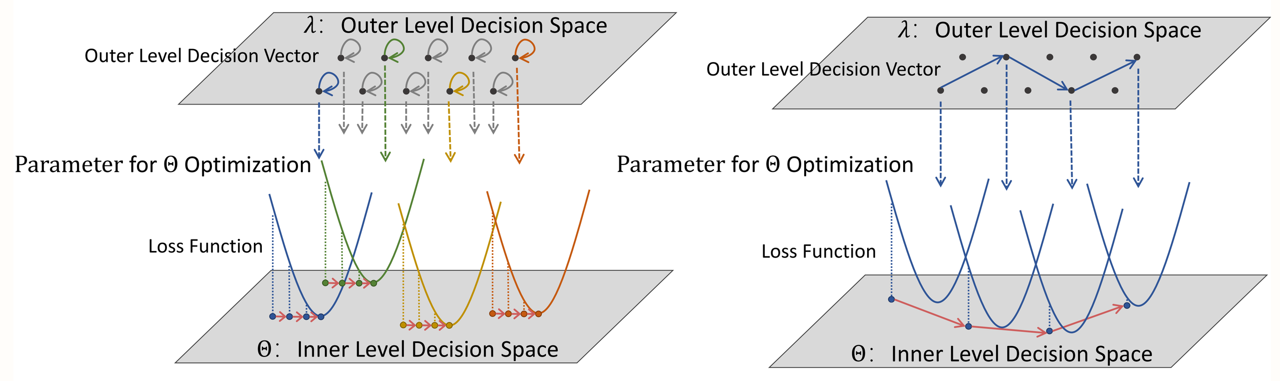

Figure 4: Compared to the fixed approach (left), adaptive regularization (right) can enjoy more efficient exploration in λ space.

How to Tackle the Regularization Tuning Problem for Recommender Models?

Based on the above hypotheses, we propose λOpt to learn to regularize recommender models in finer levels.

MF-BPR with Fine-grained Regularization

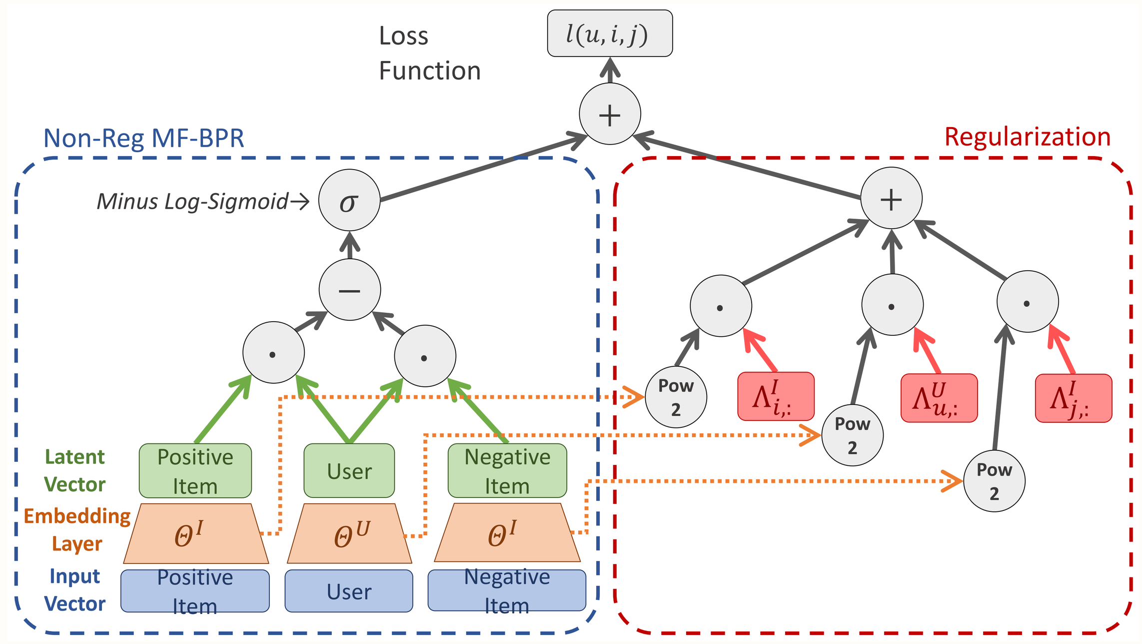

Figure 5: λOpt endows MF-BPR with fine-grained regularization.

Alternating Optimization

Regularization tuning can be regarded as a bi-level optimization problem

$$\min_{\Lambda} \sum_{(u',i',j')\in S_V} l(u',i',j' | \arg\min_{\Theta} \sum_{(u,i,j)\in S_T} l(u,i,j|\Theta,\Lambda)),$$

At iteration t,

- Fix Λ, Optimize Θ -> almost the same as conventional MF-BPR except that λ is fine-grained
- Fix Θ, Optimize Λ -> find a Λ which achieves the smallest validation loss

Fix Θ, Optimize Λ

Taking a greedy perspective, we look for Λ which can minimize the next-step validation loss

- If we keep using current Λ for next step, we would obtain $\bar{\Theta}_{t+1}$
- Given $\bar{\Theta}_{t+1}$, our aim is $\min_\Lambda l_{S_V}(\bar{\Theta}_{t+1})$

But how to obtain $\bar{\Theta}_{t+1}$ without influencing the normal Θ update? Simulate the MF update!

1. Obtain the gradients by combining the non-regularized part and penalty part
$$\frac{\partial l_{S_T}}{\partial \Theta_t} = \frac{\partial \overline{l_{S_T}}}{\partial \Theta_t} + \frac{\partial \Omega}{\Theta_t}.$$
Note that Λ is the only variable here.
2. Simulate the operations that the MF optimizer would take
$\bar{\Theta}_{t+1} = f(\Theta_t, \frac{\partial l_{S_T}}{\partial \Theta_t})$ where f denotes the MF update function.

To avoid obscure derivation of gradients introduced by the MF optimizer and fine-grained regularization, we rely on auto-differentiation to implement λOpt. That is, we first prepare the non-regularized gradients using a simulate forward & backwards on training set. Then we use a Λ-network (where the weights is Λ) to do all the aforementioned gradient combination, MF optimizer simulation, and computation of validation loss. After a forward & backward pass of Λ-network, we get the Λ for next step.

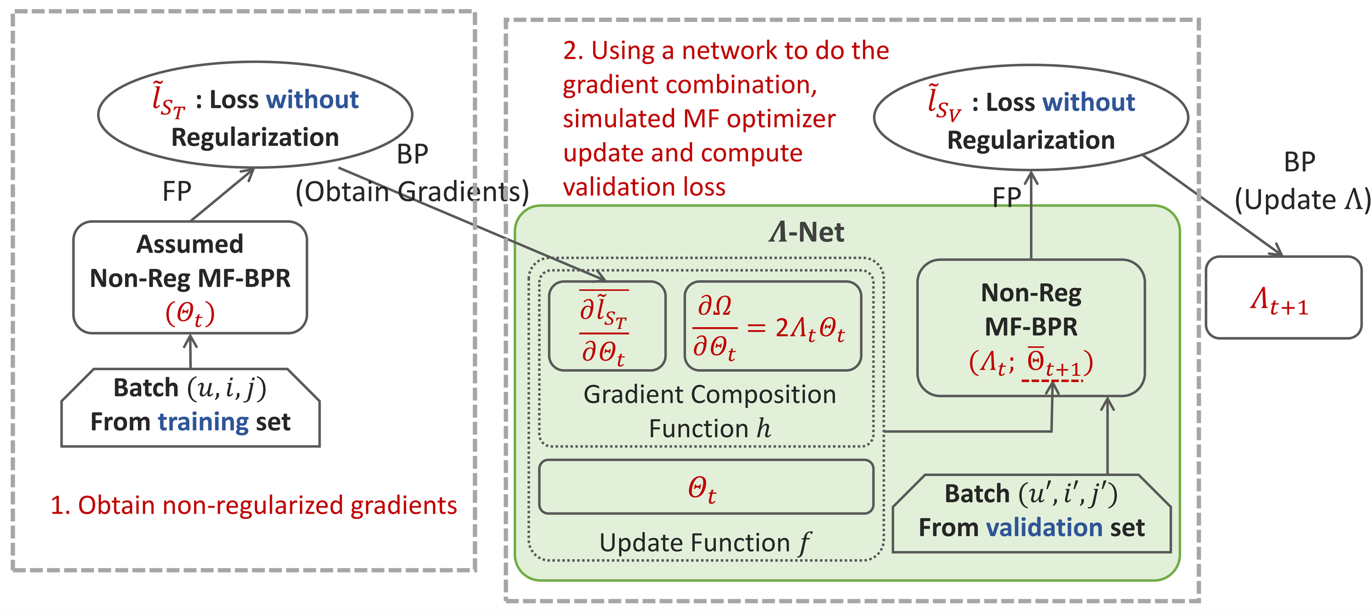

Figure 6: λOpt: fix Θ, update Λ.

Results

Performance Comparison

Method	Amazon Food Review					MovieLens 10M				
	AUC	HR@50	HR@100	NDCG@50	NDCG@100	AUC	HR@50	HR@100	NDCG@50	NDCG@100
SGDA [26]	0.8130	0.1786	0.3857	0.1002	0.1413	0.9497	0.2401	0.3706	0.0715	0.0934
AMF [15]	0.8197	0.3541	0.4200	0.2646	0.2552	0.9495	0.2625	0.3847	0.0787	0.0985
NeuMF [16]	0.8103	0.3537	0.4127	0.2481	0.2218	0.9435	0.2524	0.3507	0.0760	0.0865
MF-λFix	0.8052	0.3482	0.4163	0.2251	0.2217	0.9497	0.2487	0.3779	0.0727	0.0943
MF-λOpt -D	0.8109	0.2134	0.3910	0.1292	0.1543	0.9501	0.2365	0.3556	0.0715	0.0909
-DU	0.8200	0.3694	0.4814	0.2049	0.2570	0.9554	0.2743	0.4109	0.0809	0.1031
-DI	0.8501	0.2966	0.4476	0.1642	0.2039	0.9516	0.2648	0.3952	0.0804	0.1013
-DUI	**0.8743**	**0.4470**	**0.5251**	**0.2946**	**0.2920**	**0.9575**	**0.3027**	**0.4367**	**0.0942**	**0.1158**

Sparseness & Activeness

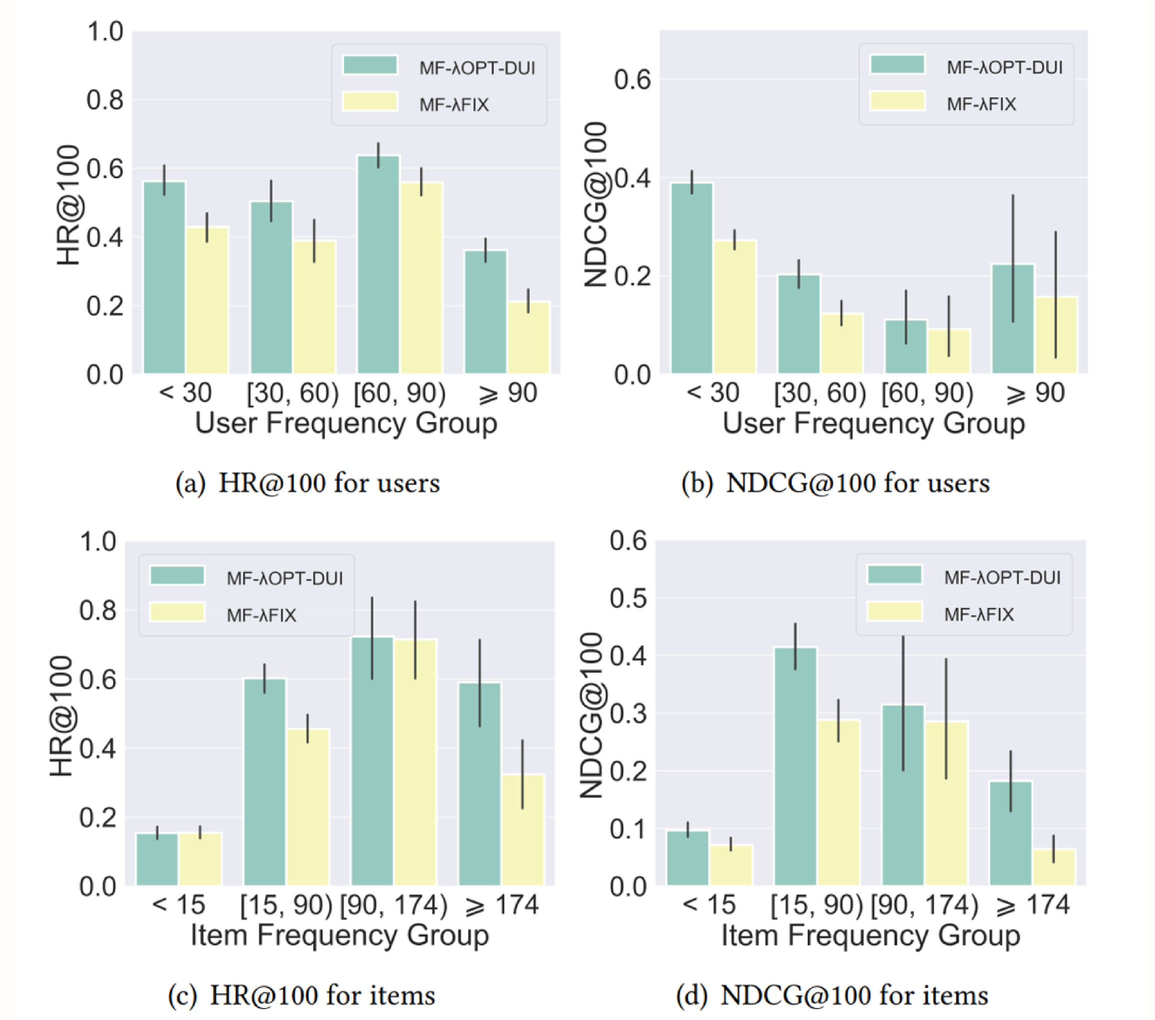

Figure 7: λOpt addresses both the sparse and active users.

Analysis of λ-trajectory

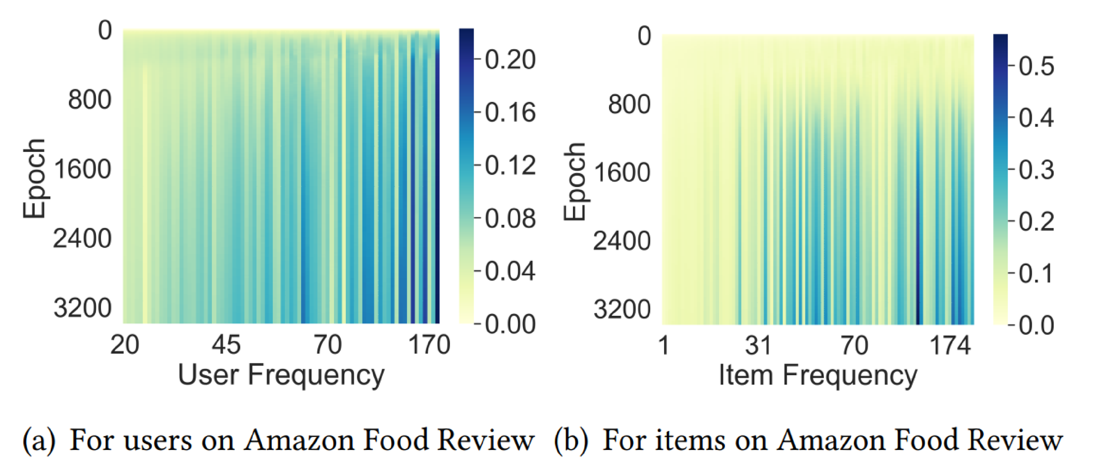

Figure 8: λOpt generates different λ-trajectories for different users/items.

25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Anchorage, Alaska