A Self-Supervised Mixture-of-Experts Framework for Multi-behavior Recommendation (Online Appendix)

Anonymous Author(s)

A Experimental Details

In this section, we provide several experimental details that are excluded in the main paper due to the space limit.

A.1 Datasets

We leverage three well-known multi-behavior benchmark datasets.

Tmall Dataset. Tmall[1] is a general e-commerce dataset from Alibaba, containing four user behaviors: click, collect, cart, and purchase. Among these, purchase history is typically considered the target behavior.

Taobao Dataset. Taobao[2] is a general e-commerce dataset from Alibaba, containing three user behaviors: click, cart, and purchase. Among these, purchase history is typically considered the target behavior.

JData Dataset. JData[3] is a general e-commerce dataset from JD, containing four user behaviors: click, collect, cart, and purchase. Among these, purchase history is typically considered the target behavior.

A.2 Baseline methods

In our work, we use nine baseline methods, which are two single-behavior approaches and seven multi-behavior approaches. We provide details of each baseline method below.

Single-behavior. Here are two single-behavior methods.

- **MF-BPR** [7]: A matrix factorization-based approach that computes user-item inner products for recommendation and optimizes performance using the BPR loss function.
- **LightGCN** [2]: A simplified GCN-based approach that eliminates unnecessary nonlinearities, resulting in improved recommendation performance with reduced model complexity. We leverage LightGCN for target graph propagation, optimized using the BPR loss sampled from the target behavior.

Multi-behavior. Here are eight multi-behavior methods.

- **LightGCN-G** [2]: We utilize LightGCN for global graph propagation, optimized using the BPR loss sampled from the target behavior.
- **MB-GMN** [14]: MB-GMN leverages a meta-learning paradigm with a meta graph neural network to model behavior heterogeneity, preserving behavior semantics through high-order connectivity.
- **CML** [10]: CML leverages a contrastive meta learning framework to model behavior heterogeneity by aligning multi-behavior views and personalizing supervision through meta-contrastive encoding.
- **CIGF** [9]: CIGF models instance-level high-order relations via compressed graph convolution and mitigates gradient conflict in multi-task learning through behavior-specific separate inputs.
- **CRGCN** [15]: CRGCN models behavior sequences by using a residual GCN module for each behavior. It facilitates information flow between modules through embedding propagation, capturing dependency relationships between behaviors.
- **BCIPM** [17]: BCIPM utilizes a global unified graph pre-training method to learn behavior-specific embeddings, which are aggregated with a behavior-contextualized preference network.
- **MB-HGCN** [16]: MB-HGCN proposes a hierarchical behavior propagation method with multi-task learning for each behavior, combining adaptive embedding aggregation across all behaviors.
- **MULE** [4]: MULE proposes target-guided denoising attention to effectively denoise uncertain interactions and a multi-grained aggregator to integrate behavior embeddings.

A.3 Evaluation details

Metrics. We use Hit-Ratio@K (HR@K) and Normalized-Discounted-Cumulative-Gain@K (NDCG@K) metrics, whose details are as follows:

- **Hit-Ratio@K (HR@K)**: This metric measures how accurately a recommender system ranks the test items within the top K positions of the ranked list.
- **Normalized-Discounted-Cumulative-Gain@K (NDCG@K)**: This metric evaluates the ranking quality by considering both the relevance of the recommended items and their positions in the ranked list.

Protocol. For the recommendation in the original setting, we rank all candidate items for each user based on s^*_{ui} and select the top K items. To evaluate the performance of each expert separately, we also rank items based solely on:

- **Visited Items Performance:** Rank items from $\mathbb{1}[(u,i) \in C_u^{(V)}]s_{ui}^{(V)}$.
- **Unvisited Items Performance:** Rank items from $\mathbb{1}[(u,i) \in C_u^{(U)}]s_{ui}^{(U)}$.

where $\mathbb{1}[\mathsf{cond}]$ is a indicator function that returns 1 if cond is true; otherwise returns 0.

This dual evaluation enables us to assess the model's ability to recommend both interacted and unvisited items effectively.

[1]https://tianchi.aliyun.com/dataset/140281
[2]https://tianchi.aliyun.com/dataset/649
[3]https://global.jd.com/

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Table 1: Performance comparison in terms of HR@50 and ND@50 between single-behavior and multi-behavior models under the standard evaluation setting. Boldface indicates the best performance, and the second-best is underlined.

Dataset	Metric	Single-behavior		Multi-behavior								
		MF-BPR	LGCN	LGCN-G	MB-GMN	CML	CIGF	CRGCN	BCIPM	MB-HGCN	MULE	MEMBER(Ours)
Tmall	HR@50	0.0751	0.0464	0.2810	0.1116	0.1053	0.1285	0.2602	0.4315	0.3843	0.4456	**0.7357**
	NDCG@50	0.0264	0.0156	0.1043	0.0359	0.0313	0.0412	0.0951	0.1580	0.1460	0.1834	**0.2716**
Taobao	HR@50	0.0510	0.0308	0.2491	0.1334	0.0870	0.1592	0.1906	0.4506	0.2660	0.3717	**0.5863**
	NDCG@50	0.0186	0.0109	0.1058	0.0512	0.0272	0.0558	0.0751	0.1748	0.1046	0.1770	**0.2414**
Jdata	HR@50	0.4664	0.4748	0.7124	0.4937	0.4430	0.5603	0.7111	0.7864	0.7888	0.7853	**0.8575**
	NDCG@50	0.2250	0.2140	0.3362	0.2160	0.1553	0.2666	0.3535	0.3719	0.3952	0.4573	**0.4974**

Table 2: Inference time comparison of multi-behavior recommendation baselines.

Dataset	MB-GMN	CML	CIGF	CRGCN	BCIPM	MB-HGCN	MULE	Ours
Tmall	254s	8s	26s	8s	6s	8s	14s	11s
Taobao	1095s	25s	134s	14s	14s	13s	27s	25s
JData	178s	7s	16s	3s	3s	3s	9s	6s

Table 3: Memory usage comparison(GB) of multi-behavior recommendation baselines.

Dataset	MB-GMN	CML	CIGF	CRGCN	BCIPM	MB-HGCN	MULE	Ours
Tmall	43.8	17.2	4.4	6.6	3.7	3.9	38.4	3.9
Taobao	47.5	34.6	5.7	5.9	6.3	6.3	32.7	6.5
JData	47.5	34.6	8.1	6.3	7.2	6.5	37.7	7.9

A.4 Machines and Implementation

Our experiments were conducted using an NVIDIA RTX A6000 GPU, equipped with 48GB of VRAM. The implementation was done using PyTorch version 1.13.1, which offers efficient handling of deep learning operations and GPU acceleration. This setup allowed us to handle large datasets and complex model architectures efficiently, ensuring reliable performance during training and evaluation.

B Additional Related Work

GNN-based Recommender Systems. Recently, graph neural networks (GNNs) have been proven effective in various recommendation tasks, including multi-behavior recommendation [1, 2, 6, 8]. GNN-based recommender systems use message passing between nodes to aggregate information from neighbors on graphs encoding observed user-item interactions. This process allows them to capture interaction patterns, leading to improved node representations that are useful for recommendation. Among various GNNs, LightGCN [2] is widely employed in multiple models for its simplicity and effectiveness [11, 12, 18]. For GNN-based methods, it is common to leverage self-supervised learning to enhance GNNs' ability to learn user and item representations. To this end, contrastive self-supervised learning aims to maximize the agreement between different views of graphs [6, 19, 20], and generative self-supervised learning aims to generate or reconstruct parts of graphs [5, 13].

Note that the self-supervised learning strategies used in our method are specially designed to tackle our unique challenges (see Section ??), and they differ significantly from those in existing recommender systems.

C Additional Analysis of MEMBER

In this section, we provide further analysis of MEMBER.

C.1 Time Complexity Analysis

We provide an encoding time complexity analysis of MEMBER with respect to the input size. Specifically, MEMBER receives a global graph $\mathcal{G} = (\mathcal{V} = \bigcup_{m\in\mathcal{M}} \mathcal{V}^{(m)}, \mathcal{E} = \bigcup_{m\in\mathcal{M}} \mathcal{E}^{(m)})$ and behavior-specific graphs $\mathcal{G} = (\mathcal{V}^{(m)}, \mathcal{E}^{(m)}), \forall m \in \mathcal{M}$. We encode each graph separately with LightGCN [2]. Note that for a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, LightGCN'S encoding time complexity is $O(|\mathcal{V}|+|\mathcal{E}|)$ with respect to the input graph size [3]. Thus, encoding time complexity of MEMBER is equivalent to

$$O\left(|\bigcup_{m\in\mathcal{M}} \mathcal{V}^{(m)}| + |\bigcup_{m\in\mathcal{M}} \mathcal{E}^{(m)}|\right) + \sum_{m\in\mathcal{M}} O(|\mathcal{V}^{(m)}| + |\mathcal{E}^{(m)}|), \tag{1}$$

$$\equiv O\left(\sum_{m\in\mathcal{M}} |\mathcal{V}^{(m)}| + |\mathcal{E}^{(m)}|\right). \tag{2}$$

After LightGCN encoding, we average the embeddings from the behavior-specific graphs. This computation has the complexity of $O\left(|\bigcup_{m\in\mathcal{M}} \mathcal{V}^{(m)}| + |\bigcup_{m\in\mathcal{M}} \mathcal{E}^{(m)}|\right)$. Since this complexity is bounded by Eq (2), still the overall complexity is equivalent to Eq (2). This is the end of the encoding, and thus, the encoding time complexity is equivalent to $O\left(\sum_{m\in\mathcal{M}} |\mathcal{V}^{(m)}| + |\mathcal{E}^{(m)}|\right)$.

C.2 Inference time Comparison

As shown in Table 2, we measured the inference time of multi-behavior recommender models and found that among the three best-performing models (MB-HGCN, MULE, and Ours), our method achieved the second-fastest inference time.

C.3 Memory Usage Comparison

As shown in Table 3, we measured the inference time of multi-behavior recommender models and found that among the three best-performing models (MB-HGCN, MULE, and Ours), our method achieved the second-fastest inference time.

C.4 Additional Real-world Data Analysis

We present an additional real-world data analysis on Taobao and Jdata in Figure 1 and 2, respectively. The results confirm that our observation remains consistent across these two datasets.

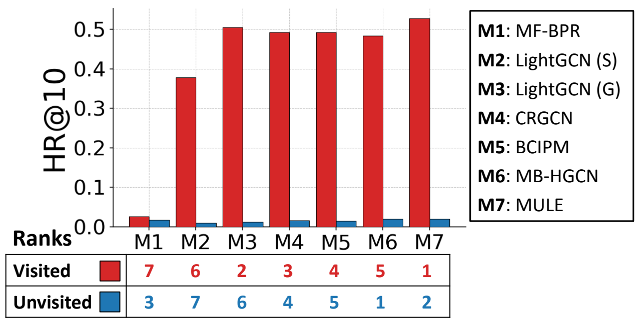

Figure 1: Performance comparison of visited vs. unvisited item recommendation on the Taobao dataset across two single-behavior and five multi-behavior recommender systems. The black line denotes the HR@10 performance gap, and Rank (lower is better) indicates model effectiveness in each setting.

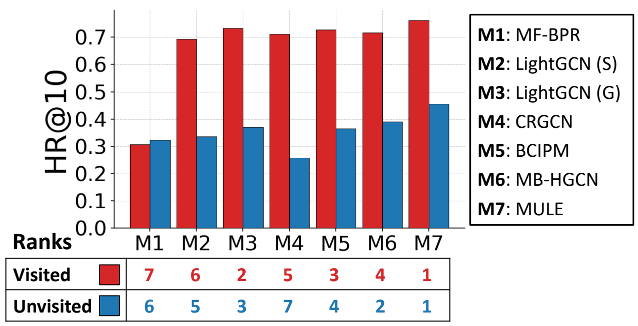

Figure 2: Performance comparison of visited vs. unvisited item recommendation on the Jdata dataset across two single-behavior and five multi-behavior recommender systems. The black line denotes the HR@10 performance gap, and Rank (lower is better) indicates model effectiveness in each setting.

(O1) Performance gap between visited and unvisited items. All methods experience significant absolute performance drop in the unvisited-item recommendation compared to the visited-item recommendation.

(O2) No one-size-fits-all model. The rankings based on performance for visited-item recommendation differ from those for unvisited-item recommendation. This discrepancy demonstrates the challenge of achieving strong recommendation performance for both item types within a single recommender system.

D Additional Experimental Results of MEMBER

We now provide additional experimental results of MEMBER.

D.1 Additional Results of MEMBER

We compare the performance of MEMBER against those of baseline methods in terms of additional evaluation metrics. We use HR@50 and NDCG@50 evaluation metrics (refer to Sec. **??** for details of HR and NDCG). As shown in Tables 1, MEMBER outperforms other methods in all settings. Thus, the superiority of MEMBER is not limited to particular evaluation metrics.

Table 4: RQ3. Effectiveness of the key components of MEMBER under the standard evaluation protocol. The best performance is highlighted in bold, and the second-best one is underlined. H@10: Hit Ratio@10. N@10: NDCG@10.

Datasets	Tmall		Taobao		JData	
Metrics	**H@10**	**N@10**	**H@10**	**N@10**	**H@10**	**N@10**
MEMBER-MoE.	0.1404	0.0747	0.1748	0.1014	0.5072	0.3203
MEMBER-$\mathcal{L}_{\text{SSL}}^{(V)}$.	0.3646	0.1745	0.3022	0.1554	0.6515	0.4193
MEMBER-$\mathcal{L}_{\text{CL}}^{(U)}$.	0.3699	0.1813	0.3067	0.1579	0.6196	0.3843
MEMBER-$\mathcal{L}_{\text{GEN}}^{(U)}$.	**0.3766**	**0.1853**	0.3184	0.1705	0.6277	0.3964
MEMBER	0.3764	0.1850	**0.3371**	**0.1808**	**0.6618**	**0.4425**

Table 5: Effectiveness of various gating function. The best performance is highlighted in bold. H@10: Hit Ratio@10. N@10: NDCG@10.

Datasets	Tmall		Taobao		JData	
Metrics	**H@10**	**N@10**	**H@10**	**N@10**	**H@10**	**N@10**
MEMBER w/ AVG.	0.1404	0.0747	0.1748	0.1014	0.5072	0.3203
MEMBER w/ L.G.	0.2465	0.2769	0.2920	0.1616	0.5887	0.3549
MEMBER	**0.3764**	**0.1850**	**0.3371**	**0.1808**	**0.6618**	**0.4425**

We also present the ablation study results for key components in Table 4. Our method consistently outperforms its variants, with the mixture-of-experts architecture proving to be the most critical component.

D.2 Gating function analysis

Recall that MEMBER combines the scores of the two experts based on the item types; using the score of the visited-item expert for visited items, while using that of the unvisited-item expert for unvisited items. In this section, we demonstrate that this simple intuitive approach outperforms the below approaches:

- **MEMBER w/ AVG.** This approach combines the score via mean pooling. Formally, the score for recommending item i to user u is computed as $s^*_{ui} = (s^{(V)}_{ui} + s^{(U)}_{ui})/2$.
- **MEMBER w/ L.G.** This approach learns the adequate score weights for respective experts. Formally, the score for recommending item i to user u is computed as $s^*_{ui} = \alpha^{(V)}_u s^{(V)}_{ui} + \alpha^{(U)}_u s^{(U)}_{ui}$, where $[\alpha^{(V)}_u, \alpha^{(R)}_u] \in (0,1)^2$ such that $\alpha^{(V)}_u + \alpha^{(U)}_u = 1$ holds. Here, $[\alpha^{(V)}_u, \alpha^{(U)}_u]$ are learnable user-specific parameters, which are obtained with the Softmax function.

Other architectural designs of these variants are the same as that of MEMBER. As shown in Table **??**, MEMBER outperforms these two variants, demonstrating the effectiveness of our score combination strategy in multi-behavior recommendation.

D.3 Hyperparameter Sensitivity Analysis

TODO

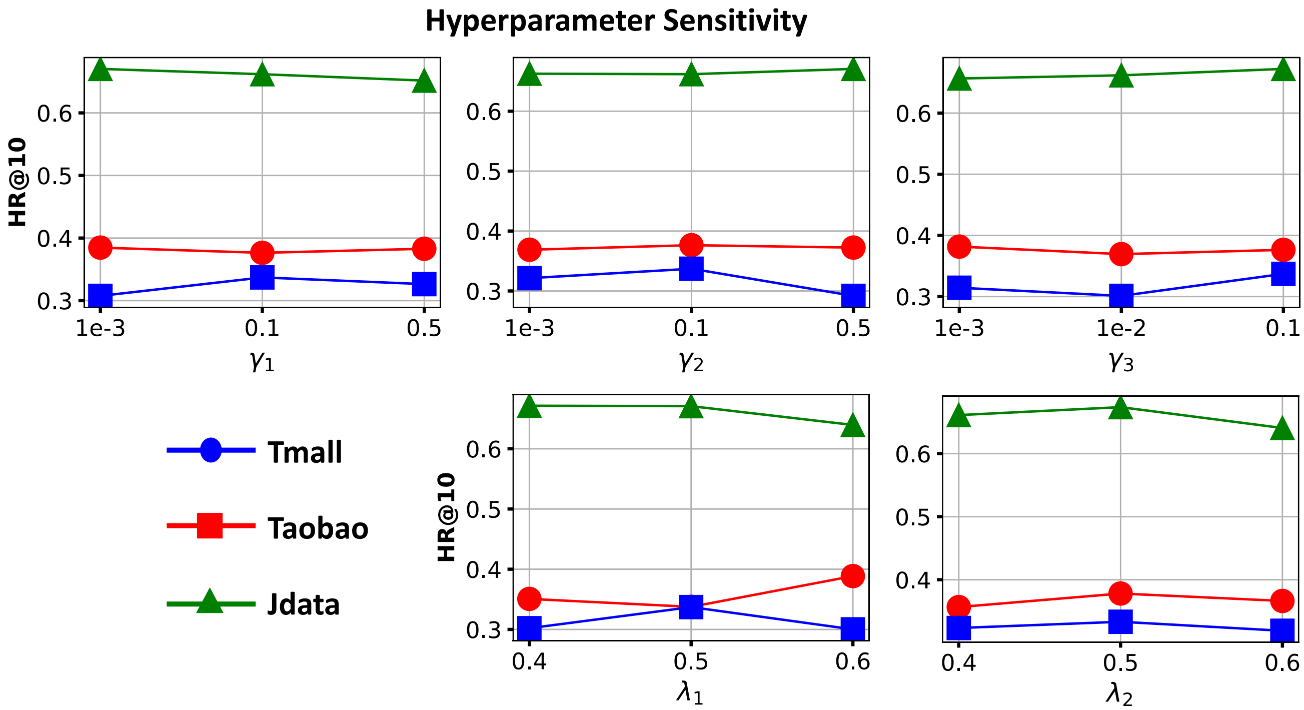

Figure 3: TODO

Table 6: Effectiveness of various gating function. The best performance is highlighted in bold. H@10: Hit Ratio@10. N@10: NDCG@10.

Datasets	Tmall		Taobao		JData	
Metrics	H@10	N@10	H@10	N@10	H@10	N@10
MEMBER w/ AVG.	0.1404	0.0747	0.1748	0.1014	0.5072	0.3203
MEMBER w/ L.G.	0.2465	0.2769	0.2920	0.1616	0.5887	0.3549
MEMBER	**0.3764**	**0.1850**	**0.3371**	**0.1808**	**0.6618**	**0.4425**

Table 7: Effectiveness of various gating function. The best performance is highlighted in bold. H@10: Hit Ratio@10. N@10: NDCG@10.

Datasets	Tmall		Taobao		JData	
Metrics	H@10	N@10	H@10	N@10	H@10	N@10
MEMBER w/ AVG.	0.1404	0.0747	0.1748	0.1014	0.5072	0.3203
MEMBER w/ L.G.	0.2465	0.2769	0.2920	0.1616	0.5887	0.3549
MEMBER	**0.3764**	**0.1850**	**0.3371**	**0.1808**	**0.6618**	**0.4425**

References

[1] Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. 2022. LightGCL: Simple Yet Effective Graph Contrastive Learning for Recommendation. In *ICLR*.

[2] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*.

[3] Geon Lee, Kyungho Kim, and Kijung Shin. 2024. Revisiting LightGCN: Unexpected Inflexibility, Inconsistency, and A Remedy Towards Improved Recommendation. In *RecSys*.

[4] Seunghan Lee, Geonwoo Ko, Hyun-Je Song, and Jinhong Jung. 2024. MuLe: Multi-Grained Graph Learning for Multi-Behavior Recommendation. In *CIKM*.

[5] Chaoliu Li, Lianghao Xia, Xubin Ren, Yaowen Ye, Yong Xu, and Chao Huang. 2023. Graph Transformer for Recommendation. In *SIGIR*.

[6] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. 2022. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *WWW*.

[7] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.

[8] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*.

[9] Guo Wei, Meng Chang, Yuan Enming, He Zhicheng, Zhang Yingxue, Chen Bo, Hu Yaochen, Li Ruiming, Tang an Xiu, and Zhang Rui. 2023. Compressed Interaction Graph based Framework for Multi-behavior Recommendation. In *WWW*.

[10] Wei Wei, Chao Huang, Lianghao Xia, Yong Xu, Jiashu Zhao, and Dawei Yin. 2022. Contrastive Meta Learning with Behavior Multiplicity for Recommendation. In *WSDM*.

[11] Yinwei Wei, Wenqi Liu, Fan Liu, Xiang Wang, Liqiang Nie, and Tat-Seng Chua. 2023. Lightgt: A light graph transformer for multimedia recommendation. In *SIGIR*.

[12] Yinwei Wei, Xiang Wang, Qi Li, Liqiang Nie, Yan Li, Xuanping Li, and Tat-Seng Chua. 2021. Contrastive learning for cold-start recommendation. In *MM*.

[13] Lianghao Xia, Chao Huang, Chunzhen Huang, Lin Kangyi, Tao Yu, and Kao Ben. 2023. Automated Self-Supervised Learning for Recommendation. In *WWW*.

[14] Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Bo Zhang, and Liefeng Bo. 2020. Multiplex behavioral relation learning for recommendation via memory augmented transformer network. In *SIGIR*.

[15] Mingshi Yan, Ziyoung Chen, Chen Gao, Zing Sun, Fan Liu, Fuming Sun, and Haojie Li. 2023. Cascading Residual Graph Convolutional Network for Multi-behavior Recommendation. *TOIS* 1 (2023), 1–24.

[16] Mingshi Yan, Zhiyong Cheng, Jing Sun, Fuming Sun, and Yuxin Peng. 2023. MB-HGCN: A hierarchical graph convolutional network for multi-behavior recommendation. *arXiv preprint arXiv:2306.10679* (2023).

[17] Mingshi Yan, Fan Liu, Jing Sun, Fuming Sun, Zhiyong Chen, and Yahong Han. 2024. Behavior-Contextualized Item Preference Modeling for Multi-Behavior Recommendation. In *SIGIR*.

[18] Yuhao Yang, Chao Huang, Lianghao Xia, and Chenliang Li. 2022. Knowledge graph contrastive learning for recommendation. In *SIGIR*.

[19] Junliang Yu, Xin Xia, Tong Chen, Lizhen Cui, Nguyen Quoc Viet Hung, and Hongzhi Yin. 2024. XSimGCL: Towards extremely simple graph contrastive learning for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 36, 2 (2024), 913–926.

[20] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 2022. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *SIGIR*.