Data-Free Model Extraction for Black-box Recommender Systems via Graph Convolutions

Zeyu Wang[1,2,3], **Yidan Song**[1,3], **Shihao Qin**[1,3], **Shanqing Yu**[1,3] *, **Yujin Huang**[4], **Qi Xuan**[1,3], **Xin Zheng**[2]*
[1]Zhejiang University of Technology [2]Griffith University
[3]Binjiang Institute of Artificial Intelligence [4]The University of Melbourne
vencent_wang@outlook.com,yushanqing@zjut.edu.cn,
xin.zheng@griffith.edu.au

Abstract

Privacy and security concerns are becoming increasingly critical for recommender systems, as model extraction attack provides an effective way to probe system robustness by replicating the model's recommendation logic — potentially exposing sensitive user preferences and proprietary algorithmic knowledge. Despite the promising performance of existing model extraction methods, they still face two key challenges: *unrealistic assumptions* on the requirement of accessible member or surrogate data and *generalization problem* where surrogate model architecture constraints lead to overfitting on generated data. To tackle these challenges, in this paper, we first thoroughly analyze how the architecture of surrogate models influences extraction attack performance, highlighting the superior effectiveness of the graph convolution architecture. Based on this, we propose a novel **D**ata-free **B**lack-box **G**raph convolution-based **R**ecommender **M**odel **E**xtraction method, dubbed **DBGRME**. Specifically, DBGRME contains: (1) **an interaction generator** to alleviate the need for member data requirements in a data-free scenario; and (2) **a generalization-aware graph convolution-based surrogate model** to capture diverse and complex recommender interaction patterns for mitigating the overfitting issue. Experimental results on various datasets and victim models demonstrate the superiority of our attack in data-free scenarios (e.g., surpassing PTQ data-require methods with 17.4% improvement on LightGCN). Code is available: https://github.com/Vencent-Won/DBGRME.git.

1 Introduction

Recently, as society shifts toward an information-driven paradigm, recommender systems have become crucial solutions to information overload issues [1, 2, 3], with extensive applications in e-commerce [4, 5], social media [6, 7, 8], etc. However, with the popularization of recommender systems, concerns regarding privacy and security have also surfaced [9, 10, 11]. One of the most pressing challenges is the model extraction attack, in which attackers try to reconstruct the victim recommender model or extract its underlying mechanisms. By doing so, attackers can not only gain access to the core technologies of the platform and its commercial value, but also manipulate recommendations [12, 13] by simulating data on user behavior based on the extracted model, potentially affecting user interests and violating their rights. Hence, there is increasing urgency for in-depth research on recommendation system model extraction attacks to explore the robustness of existing recommendation models.

*Corresponding author: Shanqing Yu and Xin Zheng.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

 https://doi.org/10.52202/085713-4680

Model extraction attacks have long been a critical component of the model security field [14, 15, 16, 17], with extensive research conducted in areas such as computer vision [18, 19, 20]. However, research on recommender model extraction attack is still in its early stages. For example, [21, 22] investigate model extraction under white-box settings using the victim model's member data, and [23] explored black-box attacks based on surrogate data drawn from a distribution similar to that of the victim's member data, along with model querying. However, these approaches rely on *unrealistic assumptions* about having complete knowledge of either the member data or high-quality surrogate data, which rarely holds in real-world scenarios. Therefore, this paper aims to explore model extraction attack on recommender systems under more realistic assumptions: black-box settings, no access to member data, and query limitations. Given these constraints, a straightforward approach is to generate synthetic user-item interactions, use them to query the victim model, and then train a surrogate model using the query responses. However, this leads to a key challenge: *the generalization problem*. In previous member data accessible attack algorithms, attackers can access the victim model's member data and only need to select a surrogate model with generalization ability for attack. However, as shown in Figure 1, when applied in data-free settings, these surrogate models exhibit a clear performance gap between synthetic queries and real-world data. This is mainly because they rely solely on user and item identifiers as input features. Lacking real member data, they are forced to resort to synthetic user features, which easily causes overfitting and limits their ability to capture the victim model's true recommendation mechanism.

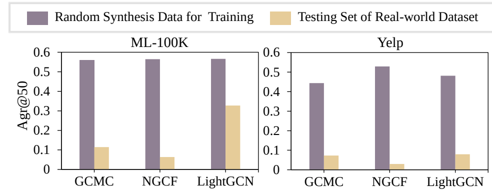


Figure 1: Agr@50 on ML-100K and Yelp with GCMC as the victim model and random data generation. The horizontal axis indicates surrogate models: GCMC, NGCF, and LightGCN.

To address these challenges, this paper begins with a comprehensive study of model extraction attack against recommender systems from multiple dimensions, including attacker knowledge, query budget, attack strategies, and model architectures. Interestingly, we find that surrogate models with architectures identical to the victim model do not always achieve optimal extraction performance. Instead, graph convolution-based surrogate models are well-suited for extraction tasks, thanks to their ability to capture complex interaction patterns within user-item networks. Building on this insight, we introduce a novel **D**ata-free **B**lack-box **G**raph convolution-based **R**ecommender **M**odel **E**xtraction method, dubbed **DBGRME**, consisting of an interaction generator and a generalization-aware graph convolution-based surrogate model. DBGRME mainly generates query data through the generator, obtains feedback from the victim model, and uses this feedback to train a fine-grained surrogate model. Specifically, to overcome the data-free limitation, we introduce an interaction generator that simulates implicit user-item feedback. Its objective is to maximize the disagreement between the surrogate and victim models, thereby driving the surrogate model to explore diverse interaction patterns. Building on this, the generator strategically produces query data that optimizes information extraction from the victim model. Moreover, to achieve efficient model extraction, we develop a generalization-aware graph convolution-based surrogate model for capturing the diverse and complex recommendation interaction, as well as the recommendation pattern of the victim model. Through the collaborative optimization of the generator and the surrogate, DBGRME achieves excellent model extraction performance. In summary, our main contributions are:

- We provide a comprehensive analysis of recommender system model extraction attack from multiple perspectives, including attacker knowledge, query budget, and model types, emphasizing the superiority of graph convolution-based models in extraction performance.
- This paper proposes DBGRME, a novel data-free black-box recommender model extraction method. It develops a graph convolution-based surrogate model that mitigates the overfitting problem, combined with a generator to achieve effective model extraction.
- Extensive experiments on four datasets and five victim models show that our method achieves superior performance in data-free scenarios, even surpassing some that use member data.

2 Background

2.1 Problem Definition

Recommendation is the process of automatically selecting and presenting relevant items to users based on their preferences, behaviors, and contextual information. Formally, given a recommendation interaction dataset $\boldsymbol{D} = \{(u, i)\}^M$ with a user set $\boldsymbol{U} = \{u\}^n$ and an item set $\boldsymbol{I} = \{i\}^m$, where M denotes the amount of interaction in the dataset, n denotes the number of users, and m denotes the number of items. The recommender system then aims to learn a function $f_\theta : \boldsymbol{U} \times \boldsymbol{I} \to \boldsymbol{S}$ that predicts the scores $\boldsymbol{S} \in \mathbb{R}^{n \times m}$ of items $\boldsymbol{I}$ for users $\boldsymbol{U}$. Recommender system model extraction attack refers to the process of extracting the victim model f_θ by querying its API and using the obtained outputs, or by training a surrogate model f_δ with target domain data to replicate its functionality. Formally, given a victim model f_θ, one can visit it to obtain the top-k recommendation list for user u:

$$\boldsymbol{T}_{\theta,u} = \mathrm{TopK}(\boldsymbol{S}_{\theta,u}, k_{\mathrm{eval}}), \tag{1}$$

where k_{eval} is the output length of the victim model. The objective function of surrogate model f_δ is:

$$\arg\max_{\delta} \sum_{u \in U} \mathrm{Agreement}(\boldsymbol{T}_{\delta,u}, \boldsymbol{T}_{\theta,u}), \tag{2}$$

where the $\mathrm{Agreement}(\mathrm{Agr})$ function is defined as:

$$\mathrm{Agreement}(\boldsymbol{T}_{\delta,u}, \boldsymbol{T}_{\theta,u}) = \frac{|\boldsymbol{T}_{\delta,u} \cap \boldsymbol{T}_{\theta,u}|}{|\boldsymbol{T}_{\theta,u}|}. \tag{3}$$

$\mathrm{Agr}@k_{\mathrm{eval}}$ represents the top-k_{eval} agreement between the victim model and the surrogate model.

2.2 Attack Overview

Attacker's Knowledge. From the data perspective, according to the attacker's access to member data, we classify scenarios into full data, partial data, and data-free settings, where the partial data is divided based on the user set of the whole dataset. From the model perspective, depending on the attacker's access to the victim model, we categorize scenarios into black-box and white-box settings. In the black-box setting, the attacker can only visit the model API and obtain the final output (i.e., the top-k recommendation list). In contrast, the white-box setting grants the attacker access to both the outputs and model architecture, etc. Additionally, we define a query as the input of a user along with their historical interactions. If either the user or the interaction history changes, it is considered a new query; otherwise, it is not counted as a new query.

Attack Mode. As shown in Table 1, according to both member data accessibility and query capabilities, model extraction attack can be categorized into three modes: 1) Partial target data (PTD): The attacker holds a subset of the target domain data but cannot query the victim model. The surrogate model is trained solely on the available data to approximate the victim's behavior. 2) Partial Target Data with Query (PTQ): The attacker not only accesses partial data but is also allowed to query the victim model. The surrogate model is initially trained on the available data and further refined through query feedback, achieving closer replication of the victim's behavior. 3) Data-free: The most challenging setting, where the attacker has no access to the target domain/surrogate data and operates under a limited query budget. In this case, the attacker must generate data to extract knowledge from the victim model.

Table 1: Attacker's knowledge and ability across different attack modes.

Mode	Partial Target Data	Query
PTD	✓	
PTQ	✓	✓
Data-Free		✓

In this paper, all experiments are conducted under black-box settings. Unlike previous works that assume access to full data, partial data, or surrogate datasets, we target a more challenging and realistic scenario: performing model extraction in a data-free setting with limited query budgets.

3 Comprehensive Investigation of Recommender System Model Extraction

In this section, we conduct a comprehensive investigation of the recommender model extraction from multiple perspectives, including the knowledge and capabilities of the attackers. Specifically, at the data level, we analyze how accessible member data rate and query budget impact the effectiveness

 https://doi.org/10.52202/085713-4680

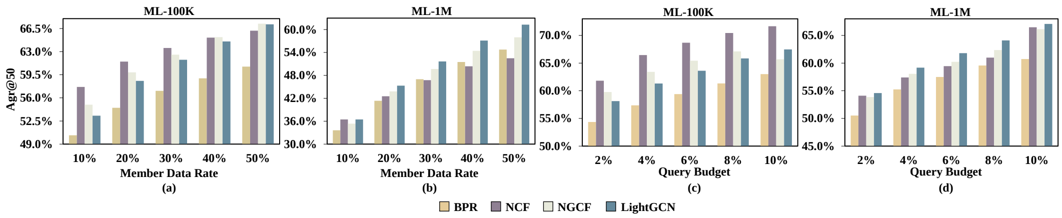


Figure 2: The Agr@50 of PTD and PTQ with varying member data rates and query budgets across four victim models. Among them, (a-b) show the PTD results on ML-100K and ML-1M, (c-d) show the PTQ results on ML-100K and ML-1M.

of model extraction; at the model level, we investigate the influence of model architectures on the model extraction task. Based on the above investigation, we identify the superiority of graph convolution-based recommender models in model extraction tasks.

3.1 The Analysis of Data Perspective

To investigate the impact of accessible member data rate and query budget on model extraction performance, we conduct experiments under the PTD and PTQ attack mode using BPR [24], NCF [25], NGCF [25], and LightGCN [26] as victim and surrogate models. Specifically, in the member data rate experiments, we analyze the extraction effectiveness under an accessible data rate of $\{10\%, 20\%, 30\%, 40\%, 50\%\}$; in the query budget experiments, based on accessible member data rate 10%, we explore the impact of a query budget of $\{20\%, 40\%, 60\%, 80\%, 100\%\}$ on extraction effectiveness. Moreover, for consistency, all victim and surrogate models are configured according to their papers. Figure 2 presents the experimental results on ML-100K and ML-1M datasets. For each victim model, we report the extraction results corresponding to the best-performing surrogate model. As shown in Figure 2(a), the extraction performance consistently improves with an increasing member data rate. Similarly, Figure 2(b) demonstrates that a larger query budget leads to better extraction outcomes. Notably, attackers with access to only 10% of member data and sufficient query budget eventually outperform those with 50% of member data but no querying capability. Based on these observations, we can draw the following conclusions: 1) Extraction performance is positively correlated with the attacker's knowledge (i.e., the accessible member data rate) and capabilities (i.e., query budget); 2) Querying the victim model significantly enhances attack performance.

3.2 The Analysis of Model Perspective

To explore the model extraction capabilities across different model architectures, we conduct cross-extraction experiments under the PTD attack mode using three classical recommender models: BPR, NCF, and LightGCN, which represent matrix factorization, deep learning, and graph convolution-based architectures, respectively. Experiments are performed on the ML-100K with accessible member data rates of 10% and 50%. To ensure fair comparison, all surrogate and victim models are configured identically. The experimental results are presented in Figure 3, where the horizontal axis denotes the surrogate models, the vertical axis denotes the victim models, and darker cells indicate better extraction effectiveness. From the perspective of varying member data rates, increasing the accessible member data leads to consistently improved surrogate performance across different victim models. Furthermore, by examining the heatmaps horizontally, we observe that LightGCN, when used as the surrogate model, consistently achieves the best extraction performance—even when the victim and surrogate architectures are identical. This finding highlights the superiority of graph convolution-based recommender models in model extraction tasks.

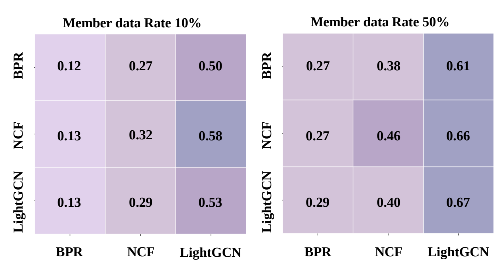


Figure 3: Agr@50 of cross model extraction with member data rate $\{10\%, 50\%\}$. Horizontal/vertical denotes the surrogate/victim model.

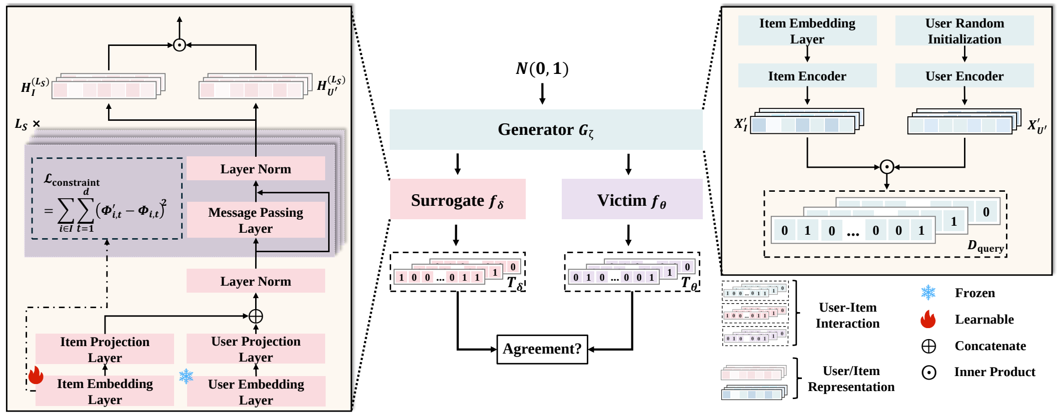


Figure 4: The framework of DBGRME. Among them, the left part depicts the workflow of the surrogate model, the middle part depicts the workflow of the model extraction and the right part depicts the workflow of the generator.

Moreover, to further explore the role and necessity of graph convolution in the model extraction task, we conducted an ablation study by removing the convolution layers of the surrogate during the testing phase for different victim models. The comparison results under PTD setting are shown in Table 2, where w/o Conv represents the LightGCN without the graph convolution layers. Surprisingly, the performance of models without convolution layers exhibits a sharp decline across victim models, highlighting the superiority of the graph convolution mechanism in recommendation model extraction.

Table 2: The ablation results about graph convolution mechanism on ML-100k.

victim model	BPR	NCF	NGCF	LightGCN
w/o Conv	0.0458	0.0383	0.0453	0.0481
LightGCN	0.5034	0.5766	0.5497	0.5330

4 Data-free Black-box Graph Convolution-based Recommender Model Extraction

To effectively extract a black-box recommender model in the data-free sceneries, we propose a novel **D**ata-free **B**lack-box **G**raph convolution-based **R**ecommender **M**odel **E**xtraction method, dubbed **DBGRME**, consisting of an interaction generator based on user-item implicit feedback modeling, and a surrogate model composed of a generalization-aware module and a graph convolution module. As Figure 4 described, DBGRME utilizes the generator to synthesize virtual user-item interactions, which are then applied to query the victim model. Subsequently, the surrogate model is trained by aligning its outputs with those of the victim model, thereby approximating its behavior. In this section, we provide a detailed introduction of the generator and surrogate model within the attack framework. The algorithm description of the entire training process is in Appendix A.

4.1 Generator

In data-free scenarios, attackers must generate query data and access the victim model to obtain feedback for training the surrogate model. A straightforward approach under the data-free black-box setting is to randomly create interaction data for querying. However, feedback from such random data carries limited information, requiring a larger query budget to achieve effective extraction. Moreover, in real-world scenarios, queries often face strict limitations. To address these challenges, inspired by classical graph generative models, we design an interaction generator that synthesizes user-item embeddings to simulate implicit feedback and generate interaction data for model extraction.

As illustrated in Figure 4, given the synthesized user number n' and the generator G_ζ with parameters ζ, we first initialize item embeddings $\boldsymbol{X_I} \in \mathbb{R}^{m\times d}$ using the $\mathrm{Embedding}$ layer, and initialize user with a random distribution $\boldsymbol{X_{U'}} \sim N(0,1), \boldsymbol{X_{U'}} \in \mathbb{R}^{n'\times d}$. The embeddings can be input into the encoder of items and users to get the representations of users $\boldsymbol{X'_{U'}}$, items $\boldsymbol{X'_I}$, where the encoder consists of L_G layers MLP. Then, we can get the rate matrix $\boldsymbol{S}_\zeta$ with the inner product of the

 https://doi.org/10.52202/085713-4680

representations of users and items, where $\boldsymbol{S}_{\zeta,u}$ denotes the generated rate vector of user $u \in U'$. And we can acquire the synthesized quer data $D_{\text{query}} = \{\boldsymbol{T}_{\zeta,u}\}_{u=1}^{n'}$ by sampling the top-k_{gen} items for each user, where the k_{gen} is a hyperparameter.

Inspired by the adversarial dynamics in GANs, we design the generator to maximize the disagreement between the surrogate and victim models, encouraging it to produce samples near decision boundaries where the models differ most. To achieve efficient model extraction, the synthesized queries should maximize useful information for training the surrogate model. Therefore, we train the generator by maximizing the Bayesian Personalized Ranking (BPR) Loss, which captures the disagreement between the victim and surrogate predictions:

$$\mathcal{L}_{\text{generator}} = -\sum_{u \in \boldsymbol{U}'} \sum_{i \in \boldsymbol{T}_{\theta,u}} \sum_{j \notin \boldsymbol{T}_{\theta,u}} -\ln\left(S_{\delta,ui} - S_{\delta,uj}\right), \tag{4}$$

where $S_{\delta,ui}$ denotes the prediction score of user u to item i with surrogate f_δ based on input D_{query}.

4.2 Surrogate Model

In attacks where member data is accessible, attackers can simply adopt a generalizable recommender architecture as a surrogate model, relying solely on user identifiers for representation learning. However, under the data-free scenario, such surrogate models—with independently learned user-specific embeddings—often fall into the overfitting trap: while the surrogate may perform well on synthetic data, it ultimately fails to capture the true member data distribution and the underlying recommendation logic of the victim model.

Building on the above analysis and the findings from Section 3, we propose a novel generalization-aware graph convolution-based surrogate model architecture. As illustrated in Figure 4, our surrogate model consists of two main components: the generalization-aware module and the graph convolution module. Specifically, considering the ineffectiveness of individualized training for synthesized user embeddings, the generalization-aware module enhances model generalization by freezing user embeddings and introducing a feature projection layer. This design transforms the previous user-specific learning paradigm into capturing platform-level latent recommendation patterns. Formally, given synthesized query data D_{query} with synthesized user set $\boldsymbol{U}' = \{u\}^{n'}$, the embeddings of synthesized users and items can be denoted as $\boldsymbol{E}_{\boldsymbol{U}'}$ and $\boldsymbol{E}_{\boldsymbol{I}}$ with corresponding weights $\boldsymbol{\psi}_{\boldsymbol{U}'} \in \mathbb{R}^{n' \times d}$ and $\boldsymbol{\psi}_{\boldsymbol{I}} \in \mathbb{R}^{m \times d}$, where n' denotes the number of synthesized users, d denotes the embedding dimension and the user embedding weights are frozen, $\boldsymbol{\psi}_{\boldsymbol{U}'} \leftarrow \text{constant}$. Then, one can apply the user and item embeddings to the feature projection layer and get the projection representation $\boldsymbol{E}^{'''}$:

$$\boldsymbol{E}^{'}_{\boldsymbol{U}'} = \text{FeedForward}(\boldsymbol{E}_{\boldsymbol{U}'}) \in \mathbb{R}^{n' \times d}, \qquad \boldsymbol{E}^{'}_{\boldsymbol{I}} = \text{FeedForward}(\boldsymbol{E}_{\boldsymbol{I}}) \in \mathbb{R}^{m \times d}, \tag{5}$$

$$\boldsymbol{E}^{''}_{\boldsymbol{U}'} = \text{FeedForward}(\phi(\boldsymbol{E}^{'}_{\boldsymbol{U}'})) \in \mathbb{R}^{n' \times d}, \qquad \boldsymbol{E}^{''}_{\boldsymbol{I}} = \text{FeedForward}(\phi(\boldsymbol{E}^{'}_{\boldsymbol{I}})) \in \mathbb{R}^{m \times d}, \tag{6}$$

$$\boldsymbol{E}^{'''} = \text{LayerNorm}([\boldsymbol{E}^{''}_{\boldsymbol{U}'} \oplus \boldsymbol{E}^{''}_{\boldsymbol{I}}]) \in \mathbb{R}^{(n'+m) \times d}, \tag{7}$$

where $\boldsymbol{E}'$ and $\boldsymbol{E}''$ represent the process variables, the $\oplus$ denotes the concatenate operation, $\phi(\cdot)$ denotes the nonlinearity function, $\text{FeedForward}(\cdot)$ denotes feed forward layer, and $\text{LayerNorm}(\cdot)$ denotes layer normalization. This approach replaces the individualized optimization of user embeddings with the feature projection mechanism, effectively capturing the underlying behavior patterns of users and items. Then, the surrogate model utilizes the graph convolution backbone to capture the user-item interactions. For training stability, the representations update with symmetric normalization:

$$\boldsymbol{z}_u^{(l)} = \sum_{i \in \mathcal{N}(u)} \frac{1}{\sqrt{|\mathcal{N}(u)|}\sqrt{|\mathcal{N}(i)|}} \boldsymbol{h}_i^{(l)} + \boldsymbol{e}_u^{'''}, \qquad \boldsymbol{z}_i^{(l)} = \sum_{u \in \mathcal{N}(i)} \frac{1}{\sqrt{|\mathcal{N}(u)|}\sqrt{|\mathcal{N}(i)|}} \boldsymbol{h}_u^{(l)} + \boldsymbol{e}_i^{'''}, \tag{8}$$

$$\boldsymbol{h}_u^{(l+1)} = \text{LayerNorm}(\boldsymbol{z}_u^{(l)}), \qquad \boldsymbol{h}_i^{(l+1)} = \text{LayerNorm}(\boldsymbol{z}_i^{(l)}), \tag{9}$$

where $\boldsymbol{e}''' \in \boldsymbol{E}'''$ represents the projection representation of user or item, and is simultaneously utilized as the initial hidden state $h^{(0)}$, $\boldsymbol{z}_i^{(l)}$ and $\boldsymbol{z}_u^{(l)}$ are the process variables of graph convolution, $\mathcal{N}(u)$ denotes the set of items that are interacted by user u, $\mathcal{N}(i)$ denotes the set of users that interact with item i. After L layers update, one can acquire the final representation of user $\boldsymbol{h}_u^{(L)}$, item $\boldsymbol{h}_i^{(L)}$,

Table 3: Comparison of Agr@50 and R@50 obtained from various attack. Among them, the "black-box" corresponds the recommendation performance R@50 of victim models, the bolden highlights the best extraction performance, and the underline indicates the sub-optimal performance.

Datasets	Victim Models	Black-box	Partial Data				Data-Free											
			PTD		PTQ		Random-Attack		Random-DBGRME		Generator-Attack		DFME		DBGRME	
		R@50	R@50	Agr@50	R@50	Agr@50	R@50	Agr@50	R@50	Agr@50	R@50	Agr@50	R@50	Agr@50	R@50	Agr@50
ML-100K	BPR	0.3499	0.3260	0.5034	0.3426	0.5936	0.1969	0.2724	0.2136	0.2945	0.1974	0.2735	0.1447	0.2181	**0.2172**	**0.2976**
	NCF	0.3586	0.3260	0.5766	0.3471	0.6856	0.2115	0.3074	0.2168	0.3195	0.2129	0.3076	**0.2191**	0.3034	0.2182	**0.3197**
	GCMC	0.3815	0.3260	0.5158	0.3498	0.6202	0.2215	0.3270	0.2789	0.4036	0.2445	0.3472	0.1867	0.2579	**0.3173**	**0.4707**
	NGCF	0.3832	0.3260	0.5497	0.3504	0.6454	0.1206	0.2106	0.2867	0.4781	0.1669	0.2808	0.1867	0.3373	**0.2946**	**0.4891**
	LightGCN	0.3921	0.3260	0.5330	0.3524	0.6378	0.3417	0.6495	0.3535	0.7514	0.3491	0.6532	0.3339	0.5803	**0.3599**	**0.7727**
ML-1M	BPR	0.3739	0.2581	0.3363	0.3614	0.5633	0.0626	0.0798	**0.2037**	**0.2673**	0.0543	0.0647	0.0487	0.0609	0.2022	0.2664
	NCF	0.3779	0.2581	0.3647	0.3585	0.5788	0.2027	0.2972	0.2025	0.3174	0.2021	0.3021	0.2001	0.2911	**0.2034**	**0.3185**
	GCMC	0.4156	0.2581	0.3407	0.3802	0.5828	0.2021	0.3092	0.2042	0.3130	0.2022	0.3095	0.2000	0.2982	**0.2347**	**0.3467**
	NGCF	0.4253	0.2581	0.3535	0.3823	0.5911	0.1986	0.2634	**0.2076**	0.2702	0.2027	0.2719	0.1978	0.2626	0.2034	**0.2773**
	LightGCN	0.4337	0.2581	0.3647	0.3911	0.6281	0.2897	0.4179	0.3427	0.5572	0.2840	0.4211	0.2074	0.2775	**0.3596**	**0.6047**
Yelp	BPR	0.0833	0.0794	0.2387	0.0961	0.4098	0.0229	0.0575	**0.0245**	**0.0705**	0.0218	0.0589	0.0238	0.0593	0.0235	0.0703
	NCF	0.0777	0.0794	0.2175	0.0945	0.3288	0.0188	0.1459	0.0221	0.1994	0.0155	0.1445	0.0136	0.1470	**0.0232**	**0.2004**
	GCMC	0.1059	0.0794	0.2423	0.0963	0.4686	0.0437	0.0797	0.0371	0.1131	0.0423	0.0999	0.0509	0.1504	**0.0618**	**0.1942**
	NGCF	0.1065	0.0794	0.2597	0.0976	0.4815	0.0523	0.1578	0.0562	0.1973	0.0513	0.1697	0.0558	0.1462	**0.0652**	**0.2872**
	LightGCN	0.1266	0.0794	0.2850	0.1087	0.4571	0.0927	0.4768	0.1008	0.5751	0.0951	0.4884	0.0707	0.2830	**0.1037**	**0.6062**
Gowalla	BPR	0.1857	0.1864	0.3545	0.2167	0.5108	0.0571	0.0642	**0.0675**	**0.0719**	0.0591	0.0587	0.0678	0.0697	0.0654	0.0704
	NCF	0.1853	0.1864	0.3231	0.2106	0.3589	0.0626	0.0738	**0.0667**	0.0753	0.0581	0.0739	0.0472	0.0584	0.0659	**0.0763**
	GCMC	0.2079	0.1864	0.3489	0.2214	0.4897	0.0523	0.0707	0.1136	0.1679	0.0702	0.1277	0.1106	0.1528	**0.1350**	**0.2252**
	NGCF	0.2408	0.1864	0.3407	0.2226	0.4783	0.1231	0.1779	0.1428	0.2482	0.1297	0.1783	0.0789	0.1237	**0.1662**	**0.2828**
	LightGCN	0.2758	0.1864	0.3589	0.2366	0.5146	0.2071	0.5663	0.2014	0.5871	0.2145	0.5747	0.1609	0.3665	**0.2177**	**0.6154**

and compute the user-item rate $S_{\delta,ui}$. To align the output of the surrogate model and victim model to achieve model extraction, we apply the BPR loss as the loss function of the surrogate model:

$$\mathcal{L}_{BPR} = \sum_{u \in \boldsymbol{U}'} \sum_{i \in \boldsymbol{T}_{\theta,u}} \sum_{j \notin \boldsymbol{T}_{\theta,u}} - \ln\left(S_{\delta,ui} - S_{\delta,uj}\right) + \lambda ||\boldsymbol{E}||^2, \tag{10}$$

where λ is the regularization constant. Moreover, to balance the model architecture and prevent the aggressive optimization of item embeddings, we add a constraint to limit the magnitude of item embedding weight updates:

$$\mathcal{L}_{\text{constraint}} = \sum_{i \in \boldsymbol{I}} \sum_{t=1}^{d} (\psi'_{i,t} - \psi_{i,t})^2, \tag{11}$$

where $\psi_{i,t} \in \mathbb{R}$ denotes the t-th dimensional value of $\boldsymbol{\psi}_i$.

5 Experiments

In this section, we conduct a comprehensive experimental evaluation of the proposed DBGRME to validate its effectiveness in black-box recommender model extraction and its generalization performance in a data-free scenario. The experiments consist of four key aspects: First, we perform a comparative study to assess the extraction performance of DBGRME across different victim models and datasets, comparing it with existing methods. Second, we conduct a query budget analysis to investigate the impact of query limitations on the performance of the surrogate model. Third, we performed an ablation study to analyze the contribution of each key module. Finally, we further analyze the behavior of the generator to gain deeper insights into its role in model extraction.

5.1 Experimental Setup

Datasets. We use four popular recommendation datasets to evaluate our methods: ML-100K, ML-1M [27], Yelp [28] and Gowalla [29]. We follow the preprocessing in [26] to process the rating data into implicit feedbacks, and split them into 8:2 as training set and testing set for victim model training. Also, the testing set will be used to test the model extraction attack performance.

Victim Model. We choose five classical recommender models BPR [24], NCF [25], GCMC [30], NGCF [25], and LightGCN [26] to verify the performance of different attack algoritms. In particular, they are all trained with the settings that their paper recommended.

Baselines. We compare DBGRME against six baselines: PTD, PTQ, Random-Attack, Random-DBGRME, Generator-Attack, and DFME [31]. Regarding the surrogate model, PTD, PTQ, Random-Attack, Generator-Attack, and DFME all use traditional surrogate models (BPR, NCF, GCMC, NGCF,

 https://doi.org/10.52202/085713-4680

and LightGCN) for model extraction, while Random-DBGRME employs the surrogate of DBGRME. In terms of member data rate, PTD and PTQ utilize 10% of member data, whereas Random-Attack, Random-DBGRME, Generator-Attack, and DFME operate under a data-free setting. As for querying, PTQ can leverage member data to query victim models, Random-Attack and Random-DBGRME rely on random generation, Generator-Attack uses the generator we developed, and DFME utilizes its autoregressive generation method. Notably, we record the best results of PTD, PTQ, Random-Attack, Generator-Attack, and DFME across different surrogate models. Moreover, the detailed experimental settings are provided in the Appendix B.

5.2 Main Results

To validate the effectiveness of DBGRME, we conduct extensive experiments under two attack scenarios using six model extraction methods across four publicly available datasets and five victim models. For comprehensive evaluation, we adopt Recall@50 (R@50) and Agr@50 as the primary metrics. Specifically, Recall assesses the similarity in recommendation performance between the surrogate model and the victim model, while Agr quantifies the effectiveness of the surrogate model in capturing the victim model's recommendation mechanism. The experimental results are presented in Table 3, where the Black-box denotes the performance of the black-box victim model, and the bolden/ underlined values indicate the best/suboptimal results under the data-free setting.

Based on the observations, we can derive the following insights: First, in terms of overall extraction performance, DBGRME exhibits exceptional effectiveness in data-free scenarios and consistently achieves the best results. Notably, it is capable of approaching or even surpassing attacks conducted under the PTD and PTQ assumptions, particularly, averaging 17.4% improvement over the PTQ with the victim model LightGCN. Second, from the perspective of surrogate model architectures, both Random-DBGRME and DBGRME, which are based on the surrogate we proposed, consistently outperform traditional surrogate models, emphasizing the strong generalization capabilities of the surrogate mdoel of DBGRME. Additionally, generator-based methods outperform those relying on randomly generated query data, confirming the effectiveness of our proposed training framework. However, BPR and NCF present exceptions due to the constraints imposed by their query mechanisms when handling new data. The overfitting issue is particularly severe for traditional surrogate models on these two victim models, whereas our surrogate model architecture alleviates this problem to a certain extent, further highlighting the superiority of our approach.

5.3 The Analysis of Querying Budget

To investigate the impact of query budget on extraction performance, we conduct extraction experiments under varying query budgets for both Random-DBGRME and DBGRME. The experimental results on ML-100K, ML-1M, and victim model LightGCN are presented in Figure 5. First, from an overall perspective, DBGRME consistently outperforms Random-DBGRME, and DBGRME is more sensitive to changes in the query budget, further demonstrating the superiority of our proposed method. Interestingly, we find that increasing the query budget does not always lead to a continuous improvement in extraction performance. Instead, after reaching a certain threshold, the performance tends to plateau. Additionally, the results indicate that different victim models exhibit varying sensitivities to query budget changes, implying that model-specific characteristics play a role in determining the effectiveness of query-based extraction. These findings highlight the importance of balancing query efficiency and extraction performance when performing black-box model extraction.

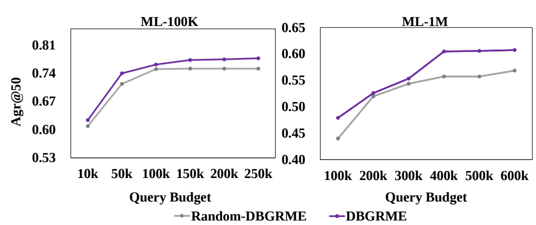


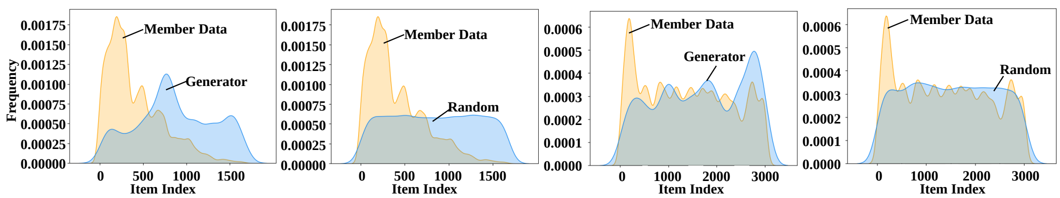
Figure 5: Agr@50 on ML-100K and ML-1M across different query budgets.

5.4 Ablation Study

To clarify the contributions of each module in DBGRME, we conducted ablation studies on the generalization-aware (GA) module and the constraint loss. The experimental results, summarized in Table 4, show the performance on the ML-100k and ML-1M datasets across five victim models

Table 4: The ablation results on ML-100k and ML-1M. The bolden is the best performance.

Datasets	Victim Models	Black-box	w/o GA		w/o constraint		w/o GA&constraint		DBGRME	
		R@50	R@50	Agr@50	R@50	Agr@50	R@50	Agr@50	R@50	Agr@50
ML-100K	BPR	0.3499	0.1795	0.2805	0.1887	0.2923	0.1749	0.2768	**0.2172**	**0.2976**
	NCF	0.3586	0.2091	0.3119	0.2166	0.3168	0.2115	0.3047	**0.2182**	**0.3197**
	GCMC	0.3815	0.3035	0.4652	0.3097	0.4726	0.3036	0.4628	**0.3173**	**0.4894**
	NGCF	0.3832	0.2508	0.3849	0.2890	0.4686	0.2177	0.3506	**0.2946**	**0.4891**
	LightGCN	0.3921	0.3340	0.7404	0.3509	0.7657	0.353	0.7400	**0.3599**	**0.7727**
ML-1M	BPR	0.3739	0.1434	0.1767	0.1980	0.2534	0.0871	0.1049	**0.2022**	**0.2664**
	NCF	0.3779	0.1965	0.2903	0.1983	0.2966	0.1935	0.2877	**0.2034**	**0.3185**
	GCMC	0.4156	0.2053	0.3120	0.2289	0.3296	0.2033	0.3106	**0.2347**	**0.3467**
	NGCF	0.4253	0.1008	0.1277	0.2015	0.2657	0.0993	0.1266	**0.2034**	**0.2773**
	LightGCN	0.4337	0.3219	0.5528	0.3429	0.5616	0.3235	0.5464	**0.3596**	**0.6047**



(a) The results on ML-100K dataset. (b) The results on ML-1M dataset.

Figure 6: The visualization of data distribution of member data and generated data with different generation methods.

(BPR, NCF, GCMC, NGCF, and LightGCN). In Table 4, **w/o GA** indicates the removal of the generalization-aware module, where the model instead uses the Embedding Layer for user and item encoding and learning; **w/o constraint** refers to the removal of constraints on item parameter updates, and **w/o GA&constraint** means both the GA module and the constraint on item parameter updates are ablated. The results show a significant drop in performance for all variants of DBGRME, indicating that both the GA module and the constraint loss contribute positively to the performance of DBGRME, further validating the effectiveness of our method. Specifically, the "w/o GA&constraint" variant has the greatest impact on model performance, followed by "w/o GA," while "w/o constraint" shows the least impact. This phenomenon makes sense because the GA module is designed to address the overfitting issue that may arise in prior extraction attacks, and the constraint loss helps the GA module by restricting aggressive updates to item parameters while enhancing the optimization of feature mapping layers within the GA module. The synergistic effect of both components achieves optimal extraction performance.

5.5 The Analysis of Generator

To deeply analyze the role of the generator in the model extraction experiments, we conducted a visual analysis of the data distribution of virtual query data and the member data of the victim model. Figure 6 presents the experimental results on the ML-100K and ML-1M datasets using LightGCN as the victim model, where the horizontal axis represents the item index and the vertical axis represents the item interaction frequency. It is evident that, compared to the randomly generated data, the distribution of the data generated by the generator is more concentrated and aligns more closely with the member data. It exhibits similar peak variations and trends in multiple key regions, indicating that the generator can produce data that is more informative for training the surrogate model, which aligns with our training objective.

6 Related Work

Model extraction attack [15] aim to reconstruct or approximate a target model by visiting its API, often under black-box conditions. Although extensive research has been conducted on model extraction in domains such as computer vision [20, 18], the exploration of model extraction attack in recommender systems remains relatively limited and underdeveloped. Early research on model extraction in recommender systems typically assumes access to either the full data/partial member data [21, 22], or a surrogate dataset [23] that shares a similar distribution. However, such assumptions

 https://doi.org/10.52202/085713-4680

are often impractical in real-world scenarios due to data privacy and platform restrictions. DFME [31] proposes an autoregressive data synthesis strategy tailored for sequential recommender systems under data-free and black-box settings. While effective in the data-free regime, autoregressive data generation typically suffers from low diversity and limited generalizability. Moreover, its design is inherently more compatible with sequence-based models, and may not generalize well to general interaction-based recommender systems, such as collaborative filtering, etc.

7 Potential Defense

Here, we discuss the future extension of potential countermeasures from three perspectives: detection-based defense, model-level defense, and legal & ethical considerations.

Detection-based defense. Recent attacks [31, 32] often rely on autoregressive querying or generator-driven interaction synthesis. Monitoring query frequency, interaction entropy, and behavioral anomalies can help detect suspicious users. Platforms may then apply query rate limiting or introduce noise to hinder extraction.

Model-level defense. This approach involves injecting perturbations into the model's representation or output space to mislead attackers. However, in hard-label recommendation settings, such defenses require careful calibration to maintain model accuracy while degrading extraction effectiveness.

Legal & ethical considerations. In addition to technical defenses, platform operators should implement responsible deployment practices such as strict API access control, policy enforcement and legal safeguards. Raising public awareness and establishing regulatory frameworks can further mitigate the misuse of recommendation models.

In addition, this study is conducted on benchmark datasets with the aim of improving the robustness of the system and promoting awareness of potential vulnerabilities.

8 Conclusion

In this paper, We begin with a comprehensive investigation of recommender model extraction and identify the superiority of the graph convolution layer in recommender model extraction. Inspired by these key insights, we propose a novel data-free black-box graph convolution-based recommender model extraction method, termed DBGRME. Specifically, our approach introduces two key components: (1) to handle the data-free challenge, we introduce a user-item implicit feedback-based interaction generator with the training objective that generates data to maximize the disagreement between the surrogate model and victim model, and (2) to mitigate the overfitting problem, we develop a generalization-aware graph convolution-based surrogate model, consisting of a generalization-aware module and the graph convolution backbone. Based on the collaboration of these two components, DBGRME can effectively capture the recommendation mechanism of the victim model. Extensive experiments on four datasets with five victim models demonstrated that DBGRME achieves superior model extraction performance in data-free scenarios. Specifically, our method achieved an Agr@50 0.7727 and R@50 0.3599 (the victim model 0.3921) on ML-100K with the victim model LightGCN, surpassing the performance of traditional extraction methods based on the member data accessible assumption. Moreover, a limitation of this work is that we have not yet explored defense strategies against recommender model extraction attack, an important avenue for future research. Introducing differential privacy presents a promising direction; however, it is primarily suited for soft-label settings. In hard-label scenarios, its application necessitates a careful balance between preserving the recommendation and robustness.

Acknowledgements. This work is supported in part by the Key R&D Program of Zhejiang Province under Grant 2024C01025, by the Zhejiang Province Natural Science Foundation under Grant LBMHZ25F020002, by Key Technology Research and Development Program Project of Hangzhou under Grant 2024SZD1A23, 2025SZD1A41.

References

[1] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys (CSUR)*, 52(1):1–38, 2019.

[2] Muhammad Aljukhadar, Sylvain Senecal, and Charles-Etienne Daoust. Information overload and usage of recommendations. In *Proceedings of the ACM RecSys 2010 Workshop on User-Centric Evaluation of Recommender Systems and Their Interfaces (UCERSTI), Barcelona, Spain*, pages 26–33, 2010.

[3] Yating Liu, Xin Zheng, Yi Li, and Yanqing Guo. Test-time adaptation on recommender system with data-centric graph transformation. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2025.

[4] Brent Smith and Greg Linden. Two decades of recommender systems at amazon. com. *Ieee internet computing*, 21(3):12–18, 2017.

[5] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pages 158–167, 2000.

[6] Minghao Zhao, Qilin Deng, Kai Wang, Runze Wu, Jianrong Tao, Changjie Fan, Liang Chen, and Peng Cui. Bilateral filtering graph convolutional network for multi-relational social recommendation in the power-law networks. *ACM Trans. Inf. Syst.*, 40(2), September 2021.

[7] Flora Amato, Vincenzo Moscato, Antonio Picariello, and Giancarlo Sperlí. Recommendation in social media networks. In *2017 IEEE Third International Conference on Multimedia Big Data (BigMM)*, pages 213–216. IEEE, 2017.

[8] Xin Zheng, Miao Zhang, Chunyang Chen, Quoc Viet Hung Nguyen, Xingquan Zhu, and Shirui Pan. Structure-free graph condensation: From large-scale graphs to condensed graph-free data. *Advances in Neural Information Processing Systems*, 36, 2024.

[9] Yujin Huang, Han Hu, and Chunyang Chen. Robustness of on-device models: Adversarial attack to deep learning models on android apps. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 101–110. IEEE, 2021.

[10] Yujin Huang and Chunyang Chen. Smart app attack: hacking deep learning models in android apps. *IEEE Transactions on Information Forensics and Security*, 17:1827–1840, 2022.

[11] Shanqing Yu, Jintao Zhou, Meng Zhou, Yidan Song, Jiaxiang Li, Zeyu Wang, Qi Xuan, Silu Mu, and Xiaolei Qian. Fine-grained mutation operators for community hiding using genetic algorithms. *Applied Soft Computing*, page 113767, 2025.

[12] Haoyang Li, Shimin Di, and Lei Chen. Revisiting injective attacks on recommender systems. *Advances in Neural Information Processing Systems*, 35:29989–30002, 2022.

[13] Yang Yu, Qi Liu, Likang Wu, Runlong Yu, Sanshi Lei Yu, and Zaixi Zhang. Untargeted attack against federated recommendation systems via poisonous item embeddings and the defense. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4854–4863, 2023.

[14] Daryna Oliynyk, Rudolf Mayer, and Andreas Rauber. I know what you trained last summer: A survey on stealing machine learning models and defences. *ACM Computing Surveys*, 55(14s):1–41, 2023.

[15] Maria Rigaki and Sebastian Garcia. A survey of privacy attacks in machine learning. *ACM Computing Surveys*, 56(4):1–34, 2023.

[16] Yujin Huang, Zhi Zhang, Qingchuan Zhao, Xingliang Yuan, and Chunyang Chen. Themis: Towards practical intellectual property protection for post-deployment on-device deep learning models. In *34th USENIX Security Symposium (USENIX Security 25)*, 2025.

[17] Yujin Huang, Terry Yue Zhuo, Qiongkai Xu, Han Hu, Xingliang Yuan, and Chunyang Chen. Training-free lexical backdoor attacks on language models. In *Proceedings of the ACM Web Conference 2023*, pages 2198–2208, 2023.

 https://doi.org/10.52202/085713-4680

[18] Sunandini Sanyal, Sravanti Addepalli, and R Venkatesh Babu. Towards data-free model stealing in a hard label setting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15284–15293, 2022.

[19] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4954–4963, 2019.

[20] Sanjay Kariyappa, Atul Prakash, and Moinuddin K Qureshi. Maze: Data-free model stealing attack using zeroth-order gradient estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13814–13823, 2021.

[21] Konstantina Christakopoulou and Arindam Banerjee. Adversarial attacks on an oblivious recommender. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 322–330, 2019.

[22] Minghong Fang, Neil Zhenqiang Gong, and Jia Liu. Influence function based data poisoning attacks to top-n recommender systems. In *Proceedings of The Web Conference 2020*, pages 3019–3025, 2020.

[23] Zhihao Zhu, Rui Fan, Chenwang Wu, Yi Yang, Defu Lian, and Enhong Chen. Model stealing attack against recommender system. *arXiv preprint arXiv:2312.11571*, 2023.

[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, page 452–461, Arlington, Virginia, USA, 2009. AUAI Press.

[25] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019.*, pages 165–174, 2019.

[26] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020.

[27] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015.

[28] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 549–558, 2016.

[29] Eunjoon Cho, Seth A Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090, 2011.

[30] Rianne Van Den Berg, N Kipf Thomas, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2(8):9, 2017.

[31] Zhenrui Yue, Zhankui He, Huimin Zeng, and Julian McAuley. Black-box attacks on sequential recommenders via data-free model extraction. In *Proceedings of the 15th ACM conference on recommender systems*, pages 44–54, 2021.

[32] Yihao Wang, Jiajie Su, Chaochao Chen, Meng Han, Chi Zhang, and Jun Wang. Sim4rec: Data-free model extraction attack on sequential recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 12766–12774, 2025.

[33] Zechun Liu, Kwang-Ting Cheng, Dong Huang, Eric P. Xing, and Zhiqiang Shen. Nonuniform-to-uniform quantization: Towards accurate quantization via generalized straight-through estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4942–4952, June 2022.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction in this paper accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See the Section 8.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

https://doi.org/10.52202/085713-4680

Justification: The paper focuses on empirical studies.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: In Section 5 and Appendix B, we listed all the necessary information for the reproduction of the main experiment results. Moreover, we have listed the url of the anonymized code repository in the abstract.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our data and code are available in `https://github.com/Vencent-Won/DBGRME.git`. We also provide sufficient instructions, which are listed in the readme file of the repository, Section 5 and Appendix B to reproduce the main experiment results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: In Section 5 and Appendix B, we specified all the experiment details.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Improvement percentages (>10%) over PTQ mode are discussed in Section 5.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

https://doi.org/10.52202/085713-4680

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Discussions are in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have discussed it in Section 1.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: In Section 1, Section 7 and Section 8, we discuss the importance and potential ways to safeguard recommender models against our attack.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All data sources and baseline models are open-sourced. They have been properly credited and mentioned, as outlined in Section 5 and Appendix B.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

https://doi.org/10.52202/085713-4680

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: No new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

https://doi.org/10.52202/085713-4680

A Algorithm of DBGRME

Algorithm 1 Alogrithm of DBGRME

Input: Query budget Q, iteration times of generator $\text{iter\_}G$, iteration times of surrogate model $\text{iter\_}C$, the synthesized user number n', victim model f_θ

Output: surrogate model f_δ

Initialize surrogate model f_δ, generator G_ζ, budget count $q \leftarrow 0$
while $q < Q$ **do**
  for $\text{iter} = 1, \cdots, \text{iter\_}G$ **do**
    $D_{\text{query}} = G_\zeta(X_{U'}) : X_{U'} \sim N(0,1), |U'| = n'$
    Acquire the $T_\theta = \{T_{\theta,u}\}_{u \in U'}$ by querying f_θ with D_{query}
    Acquire the $S_\delta = \{S_{\delta,u}\}_{u \in U'}$ by querying f_δ with D_{query}
    Calculate $\mathcal{L}_{\text{generator}}$ by Equation (4)
    Update the generator parameter by Equation (12)
  end for
  for $\text{iter} = 1, \cdots, \text{iter\_}C$ **do**
    $D_{\text{query}} = G_\zeta(X_{U'}) : X_{U'} \sim N(0,1), |U'| = n'$
    Acquire the $T_\theta = \{T_{\theta,u}\}_{u \in U'}$ by querying f_θ with D_{query}
    Acquire the $S_\delta = \{S_{\delta,u}\}_{u \in U'}$ by querying f_δ with D_{query}
    Calculate $\mathcal{L}_{\text{surrogate}}$ by Equation (15)
    Update the surrogate model parameter by Equation (14)
  end for
  Update budget count q with Equation (16)
end while

A.1 Optimization

As is illustrated in Algorithm 1, the attack algorithm is trained based on the alternating optimization framework, which involves two phases: generator training and surrogate model training. In the generator training phase, we perform $\text{iter\_}G$ rounds of the gradient descent for G, which is trained to produce query data D_{query} that maximize the BPR loss. The generation loss will be used to update the parameters of generator G_ζ:

$$\zeta \leftarrow \zeta - lr_{\text{generator}} \nabla_\zeta \mathcal{L}_{\text{generator}}, \tag{12}$$

where $lr_{\text{generator}}$ is the learning rate of the generator, and the $\nabla_\zeta \mathcal{L}_{\text{generator}}$ can be decomposed into several components:

$$\nabla_\zeta \mathcal{L}_{\text{generator}} = \frac{\partial \mathcal{L}_{\text{generator}}}{\partial \zeta} = \frac{\partial \mathcal{L}_{\text{generator}}}{\partial D_{\text{query}}} \times \frac{\partial D_{\text{query}}}{\partial \zeta}. \tag{13}$$

Among them, considering that the discretized sampling operation in the generator may lead to gradient loss, we employ the reparameterization technique [33] to enable effective gradient propagation and achieve end-to-end optimization. This is followed by the surrogate model training phase, where we perform $\text{iter\_}C$ rounds of gradient descent for f_δ. The parameters of the surrogate model f_δ are updated using the BPR loss and the constraint loss:

$$\delta \leftarrow \delta - lr_{\text{surrogate}} \nabla_\delta \mathcal{L}_{\text{surrogate}}, \tag{14}$$

where $lr_{\text{surrogate}}$ is the learning rate of the surrogate model. And the $\mathcal{L}_{\text{surrogate}}$ can be denoted as:

$$\mathcal{L}_{\text{surrogate}} = \mathcal{L}_{\text{BPR}} + \alpha \mathcal{L}_{\text{constraint}}, \tag{15}$$

where α is the constraint constant. At each end of iteration, the query budget can be updated as:

$$q \leftarrow q + n' * (\text{iter\_}C + \text{iter\_}G). \tag{16}$$

A.2 Time Complexity

Clearly, for each main loop, the generator training phase incurs a time complexity of $\mathcal{O}(\text{iter\_}G \times n')$, while the surrogate model training phase has a complexity of $\mathcal{O}(\text{iter\_}C \times n')$, where n' is the number of synthesized users, and $\text{iter\_}G$ and $\text{iter\_}C$ denote the iteration numbers for the generator and surrogate model, respectively. Therefore, the overall time complexity per cycle is $\mathcal{O}(n' \times (\text{iter\_}G + \text{iter\_}C))$, consistent with the query budget update rule described in Equation (16).

Table 5: Statistics of datasets. The #User represents the number of users, #Item represents the number of items, #Interaction represents the number of interactions, Density quantifies the density of interactions, and Q represents the query budget.

Dataset	#User	#Item	#Interaction	Density	Q
ML-100K	944	1,683	100,000	0.0630	200k
ML-1M	6,040	3,416	1,000,209	0.0447	400k
Yelp	31,668	38,048	1,561,406	0.0013	9M
Gowalla	29,858	40,981	1,027,370	0.0008	4M

Table 6: The comparison of different model extraction algorithms.

Methods	Partial Target Data	Query	Surrogate	Generator
PTD	✓		BPR, NCF, GCMC, NGCF, LightGCN	
PTQ	✓	✓	BPR, NCF, GCMC, NGCF, LightGCN	
Random-Attack		✓	BPR, NCF, GCMC, NGCF, LightGCN	Random
Random-DBGRME		✓	Surrogate model of DBGRME	Random
Generator-Attack		✓	BPR, NCF, GCMC, NGCF, LightGCN	Generator of DBGRME
DFME		✓	BPR, NCF, GCMC, NGCF, LightGCN	Autoregressive
DBGRME		✓	Surrogate model of DBGRME	Generator of DBGRME

B Experiment Settings

All experiments are implemented in PyTorch[2] on an Ubuntu Server equipped with Intel(R) Core(TM)594 i7-8700K CPU, and 2 NVIDIA GeForce GTX 1080 Ti (with 11GB memory each).

B.1 Datasets

In Table 5, we summarize the four widely used recommendation datasets considered in Section 5, where Q represents the query budget used in the extraction attack.

B.2 Baselines

To verify the effectiveness of DBGRME, we compared it with five baselines. According to the assumption difference, we can divide them into the partial data setting and the data-free setting. The two partial data setting methods are as follows:

- Partial Target Data (PTD): a recommender model extraction attack method with partial target data accessibility assumption. PTQ directly trains a surrogate model with the accessible partial target data.
- Partial Target Data and Query (PTQ): Similar to PTD, PTQ also assumes partial access to target data but introduces an additional query mechanism to refine the surrogate model.

Regarding the implementation of PTD and PTQ, we set the partial data rate as 10%. The others are data-free setting methods:

- Random-Attack: a recommender model extraction attack method with the random data generation and a traditional surrogate model selecting from victim model pools.
- Generator-Attack: a variation of DBGRME which replaces our surrogate model with a traditional surrogate model selected from victim model pools.
- Random-DBGRME: a variation of DBGRME which replaces our generator with random data generation.
- DFME [31]: a data-free black-box model extraction method with autoregressive data generation.

[2] https://pytorch.org/

https://doi.org/10.52202/085713-4680

Among them, PTD, PTQ, Random-Attack, Generator-Attack, and DFME do not have a definite surrogate model architecture and can select a model from the victim model pools (BPR, NCF, GCMC, NGCF, LightGCN). As for the settings of DFME, we refer to [31]. Particularly, we reported the optimal surrogate model (LightGCN) results for these methods, which are presented in Table 1. Also, to clearly compare the baselines, we provide a detailed comparison in Table 6.

Table 7: Hyperparameter of victim models.

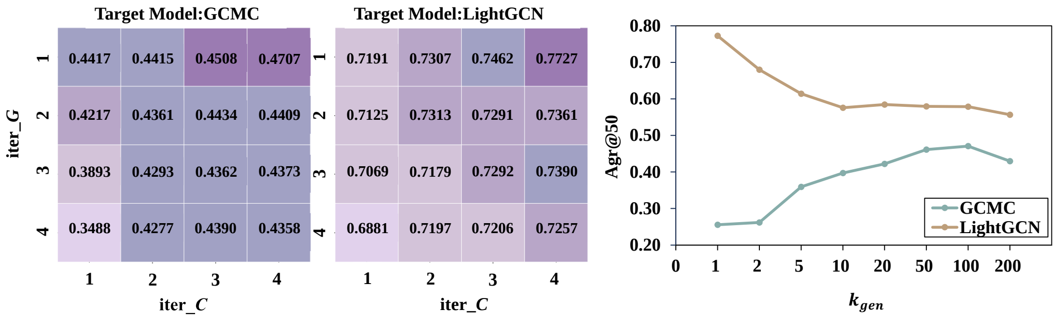
Model	BPR	NCF	GCMC	NGCF	LightGCN
hidden dim/factors	64	64	64	64	64
num layer	/	/	3	3	3
optimizer	Adam				
learning rate	$\{0.0001, 0.0005, 0.001, 0.005\}$				
batch size	$\{1024, 4096\}$				
dropout	$\{0, 0.1, 0.2\}$				
epoch	$\{600, 1000\}$				

Table 8: Hyperparameters space considered for the DBGRME selection.

Hyperparameter	Values
Generator	
· Hidden dim	$\{128, 256\}$
· Number of hidden layers L_G	$\{2, 4\}$
Surrogate	
· Hidden dim	$\{64, 128, 256\}$
· Number of hidden layers L_S	$\{2, 3\}$
· Dropout rate	$\{0, 0.1\}$
Training	
· Learning rate of surrogate $lr_{\text{surrogate}}$	$\{1e-4, 5e-4, 1e-3, 5e-3, 1e-2\}$
· Optimizer of surrogate	Adam
· Coefficient of constraint loss α	$\{0.5, 1\}$
· Iteration of surrogate iter_C	4
· Learning rate of generator $lr_{\text{generator}}$	$\{1e-4, 5e-4, 1e-3\}$
· Optimizer of generator	Adam
· Iteration of generator iter_G	1
· The interactions of each generated user k_{gen}	$\{1, 20, 50, 100\}$
· The number of generated user in one epoch n'	$\{300, 900, 2000, 6000\}$
· Batch size	$\{4096, 8192\}$

B.3 Patameter Setting

The parameters of recommender model extraction involve four parts: victim model, generator, surrogate model, and training. We report the detailed parameter space in Table 6. 1) As described in Table 7, regarding the victim model, we selected five popular recommender models (BPR, NCF, GCMC, NGCF, LightGCN), and pre-trained them with the configuration that their paper recommended. We set black-box API returns to be top-100 recommended items. 2) The generator consists of the initialization of items, users, and the encoder layer. The initialization of items and users utilize the Embedding layer and random initialization, respectively. Both the encoder layers of items and users are the fully-connected neural network, consisting of L_G layers. 3) For the surrogate model in PTD, PTQ, Random-Attack and Generator-Attack, the surrogate models are selected from the model pools (BPR, NCF, GCMC, NGCF, LightGCN) with the same configuration as victim models. And our proposed surrogate model consists of a generalization-aware module with L_S layers graph convolution backbone. Among them, the generalization-aware module includes the Embedding layer of items, users, and the Item/User Projection layers. Both the Item/User Projection layers are the fully-connected Neural Network, consisting of two layers with unit d. 4) The parameters of training mainly involve the learning rate, optimizer, batch size, dropout, query budget, etc. One of the most important parameters is the query budget, which limits the attacker's operation. And we have conducted query budget analysis experiments in Section 5.3.



(a)The results across different iter_*C* and iter_*G* on ML-100K dataset.

(b)The results of k_{gen} on ML-100K dataset.

Figure 7: Effects of the hyper-parameters.

C Hyper-parameters Experiments

To systematically explore the optimal configuration of the hyperparameters $\text{iter\_}C$, $\text{iter\_}G$, and k_{gen}, this study selects two representative models, GCMC and LightGCN, as the research subjects. Multiple comparative experiments are conducted based on the public MovieLens-100K dataset. During the hyperparameter tuning process, all possible combinations of $\text{iter\_}C$ and $\text{iter\_}G$ within the set $\{1, 2, 3, 4\}$ are evaluated to assess their impact on model performance, as shown in Figure 7(a). Figure 7(b) further presents a quantitative analysis of the dynamic influence of k_{gen} on the consistency evaluation metric under the values $\{1, 2, 5, 10, 20, 50, 100, 200\}$. Experimental results reveal that model performance is highly sensitive to these hyperparameters, underscoring the critical role of these hyperparameters in model extraction attack. Based on empirical observations and performance comparisons, $\text{iter\_}C$ = 4 and $\text{iter\_}G$ = 1 are selected as the default configuration for subsequent experiments. As for the gen parameter, the optimal value is selected individually for each model from the candidate set $\{1, 20, 50, 100\}$.

Table 9: Ablation Study

Datasets	Victim Models	Black-box	w/o GA		w/o constraint		w/o GA&constraint		DBGRME	
		R@50	R@50	Agr@50	R@50	Agr@50	R@50	Agr@50	R@50	Agr@50
Yelp	BPR	0.0833	0.0234	0.0656	0.0242	0.0676	0.0231	0.0653	**0.0235**	**0.0703**
	NCF	0.0777	0.0228	0.1757	0.0200	0.1952	0.0217	0.1637	**0.0232**	**0.2004**
	GCMC	0.1059	0.0389	0.1201	0.0401	0.1422	0.0361	0.1084	**0.0618**	**0.1942**
	NGCF	0.1065	0.0408	0.1233	0.0424	0.1552	0.0394	0.1168	**0.0652**	**0.2872**
	LightGCN	0.1266	0.0814	0.3426	0.1046	0.5855	0.0731	0.3216	**0.1037**	**0.6062**
Gowalla	BPR	0.1857	0.0616	0.0587	0.0646	0.0673	0.0632	0.0580	**0.0654**	**0.0704**
	NCF	0.1853	0.0415	0.0518	0.0636	0.0744	0.0414	0.0510	**0.0659**	**0.0763**
	GCMC	0.2079	0.0894	0.1319	0.1329	0.2162	0.0799	0.1165	**0.1350**	**0.2252**
	NGCF	0.2408	0.1504	0.2412	0.1609	0.2694	0.1413	0.2201	**0.1662**	**0.2828**
	LightGCN	0.2758	0.1770	0.4529	0.1874	0.5389	0.1716	0.4395	**0.2177**	**0.6154**

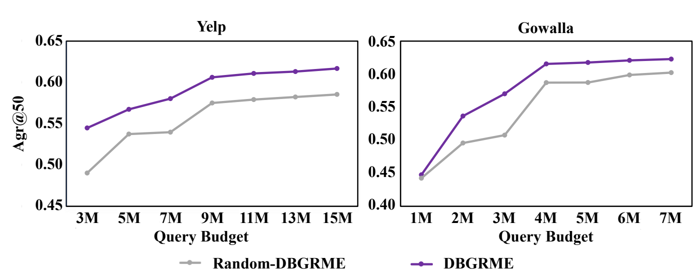


Figure 8: Agr@50 on Yelp and Gowalla across different query budgets.

https://doi.org/10.52202/085713-4680

D Supplementary Experiment

Due to space limitations, here we present additional results on Yelp and Gowalla to further verify the effectiveness of DBGRME. Specifically, Table 9 shows the results of ablation experiments, which clearly exhibit similar trends to those observed on ML-100K and ML-1M in Section 5. These findings further confirm that each module in DBGRME contributes positively.

Furthermore, Figure 8 reports the query budget experiments on Yelp and Gowalla using LightGCN as the victim model. We observe that DBGRME achieves consistent performance improvements as the query budget increases. However, the gains gradually diminish once the budget reaches a certain scale, which is consistent with the phenomena observed on ML-100K and ML-1M in Section 5.3.

Table 10: Agr@10 results of different methods across ML-100K and ML-1M datasets.

Dataset	Method	BPR	NCF	GCMC	NGCF	LightGCN
ML-100K	random-DBGRME	0.1546	0.1128	0.2177	0.1395	0.5965
	DFME	0.1370	0.1189	0.1372	0.2159	0.3353
	DBGRME	**0.1895**	**0.1672**	**0.2584**	**0.2854**	**0.6285**
ML-1M	random-DBGRME	0.0874	0.2090	0.1690	0.1477	0.4378
	DFME	0.0410	0.1989	0.1210	0.1506	0.2109
	DBGRME	**0.1424**	**0.2176**	**0.1896**	**0.1675**	**0.4725**

E High-rank Performance Analysis

Considering that in real-world recommendation scenarios, users are typically more concerned with items ranked at the top of the recommendation list, we further evaluate the high-rank agreement (Agr@10) between the surrogate model trained by DBGRME and the victim model. This evaluation provides a finer-grained perspective on whether the surrogate model can accurately capture the most critical recommendation positions. Table 10 presents the Agr@10 results of DBGRME, DFME, and random-DBGRME across five different victim models (BPR, NCF, GCMC, NGCF, and LightGCN) on ML-100K and ML-1M. These results further confirm that DBGRME is not only effective in overall agreement but also excels in top-ranked recommendations most relevant to user experience.

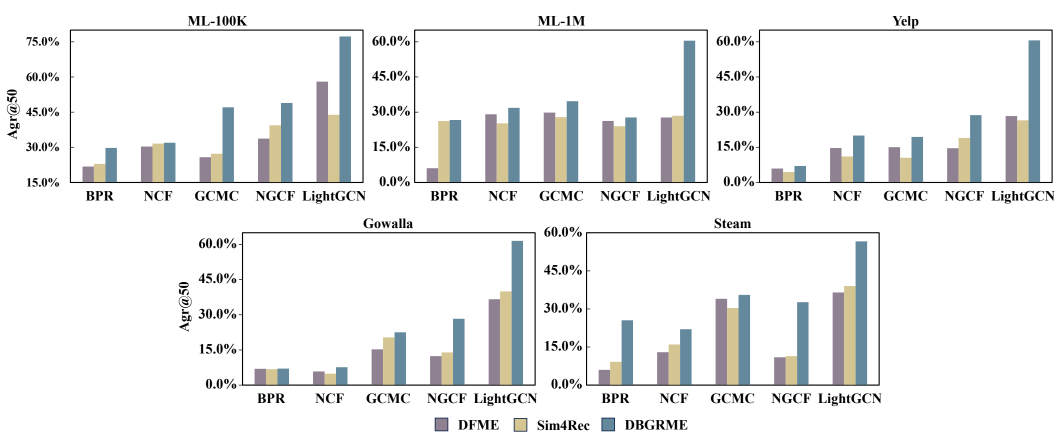


Figure 9: Comparison results with Sim4Rec on five datasets.

F Contemporary Baseline and Large-scale Dataset

To further validate the effectiveness and robustness of our method, we incorporate the contemporary state-of-the-art baseline, Sim4Rec [32], into comparisons. In addition, to examine the scalability of our approach in more practical scenarios, we extend the evaluation to a large-scale dataset, Steam, which contains 334,730 users and 13,047 items. Figure 9 presents the comparison results of DBGRME with DFME and Sim4Rec across five victim models on five datasets. We observe that DBGRME consistently outperforms both DFME and Sim4Rec in all settings, demonstrating its superiority over the state-of-the-art baselines. More importantly, the results on Steam highlight the strong scalability of DBGRME, showing that our approach remains effective even when facing extremely sparse and large-scale datasets. These findings further confirm the broad applicability of DBGRME in realistic recommendation environments.

https://doi.org/10.52202/085713-4680