INMO: A Model-Agnostic and Scalable Module for Inductive Collaborative Filtering

Yunfan Wu
Data Intelligence System Research Center, Institute of Computing Technology, CAS
University of Chinese Academy of Sciences, Beijing, China
wuyunfan19b@ict.ac.cn

Qi Cao*
Data Intelligence System Research Center, Institute of Computing Technology, CAS
caoqi@ict.ac.cn

Huawei Shen*
Data Intelligence System Research Center, Institute of Computing Technology, CAS
University of Chinese Academy of Sciences, Beijing, China
shenhuawei@ict.ac.cn

Shuchang Tao
Data Intelligence System Research Center, Institute of Computing Technology, CAS
University of Chinese Academy of Sciences, Beijing, China
taoshuchang18z@ict.ac.cn

Xueqi Cheng
CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, CAS
University of Chinese Academy of Sciences, Beijing, China
cxq@ict.ac.cn

ABSTRACT

Collaborative filtering is one of the most common scenarios and popular research topics in recommender systems. Among existing methods, latent factor models, i.e., learning a specific embedding for each user/item by reconstructing the observed interaction matrix, have shown excellent performances. However, such user-specific and item-specific embeddings are intrinsically transductive, making it difficult for them to deal with new users and new items unseen during training. Besides, the number of model parameters heavily depends on the number of all users and items, restricting their scalability to real-world applications. To solve the above challenges, in this paper, we propose a novel model-agnostic and scalable **In**ductive Embedding **Mo**dule for collaborative filtering, namely INMO. INMO generates the inductive embeddings for users (items) by characterizing their interactions with some template items (template users), instead of employing an embedding lookup table. Under the theoretical analysis, we further propose an effective indicator for the selection of template users and template items. Our proposed INMO can be attached to existing latent factor models as a pre-module, inheriting the expressiveness of backbone models, while bringing the inductive ability and reducing model parameters. We validate the generality of INMO by attaching it to Matrix Factorization (MF) and LightGCN, which are two representative latent factor models for collaborative filtering. Extensive experiments on three public benchmarks demonstrate the effectiveness and efficiency of INMO in both transductive and inductive recommendation scenarios.

*Corresponding authors.

SIGIR '22, July 11–15, 2022, Madrid, Spain.

ACM ISBN 978-1-4503-8732-3/22/07.
https://doi.org/10.1145/3477495.3532000

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Recommender System, Collaborative Filtering, Inductive Embedding, Latent Factor Model

ACM Reference Format:
Yunfan Wu, Qi Cao, Huawei Shen, Shuchang Tao, and Xueqi Cheng. 2022. INMO: A Model-Agnostic and Scalable Module for Inductive Collaborative Filtering. In *Proceedings of the 45th Int'l ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3477495.3532000

1 INTRODUCTION

Recommender systems are prevalently deployed in real-world applications to provide personalized recommendation services, helping people out of the dilemma of information overload [4, 6, 42]. In various recommendation tasks, collaborative filtering (CF) is one of the most simple and widely adopted scenarios, which has attracted extensive research attention for more than two decades [34].

Among existing methods, latent factor models have been the state-of-the-art in CF for over a decade [13, 18]. Since high-quality side information is not always available [40], the most general paradigm of latent factor models is to project the ID of a user (or item) to a specific learnable embedding, and then predict user-item interactions based on these embeddings [19]. Recently, with the success of deep learning, researchers further improve latent factor models from two aspects, i.e., representation generation [5, 31, 35, 36] and interaction modeling [10, 12]. The former line generates more informative representations based on initial user/item embeddings, e.g., utilizing graph neural networks to capture the high-order proximity [31]. The latter line devotes to enhancing the interaction modeling between users and items with powerful neural networks

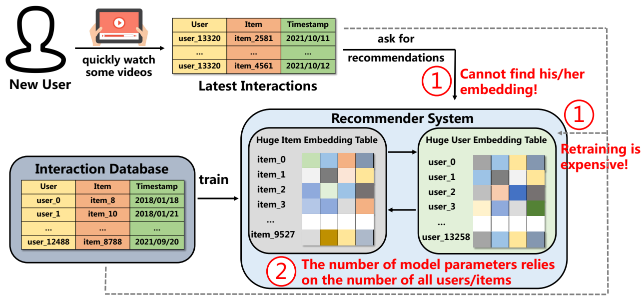

Figure 1: Limitations of existing latent factor models for collaborative filtering.

instead of a simple inner production, e.g., employing multi-layer perceptrons to learn the complex interaction function [12].

However, most of existing latent factor models rely on the user-specific (or item-specific) embedding learning, which have two critical limitations when dealing with real-world recommendation scenarios (shown in Figure 1). First, such methods are intrinsically transductive, making them hard to deal with continuous new users and new items. For example, there may be newly registered Youtube users who quickly watch some preferred videos [44] and it is a practical need to make personalized recommendations for them according to their latest behaviors. Unfortunately, with the embedding lookup paradigm, these methods cannot find the embeddings of new users or new items which are unseen during training, while the cost of retraining or incremental learning for new users/items is generally expensive [30, 43]. Moreover, with the embedding lookup table, the number of model parameters heavily depends on the number of all users and items, restricting their scalability to real-world applications with hundreds of millions of users and items.

Recently, a few works have also noticed the limitations of the transductive nature of existing latent factor models and attempted to propose inductive collaborative filtering methods without side information [7, 27, 33, 40, 44]. However, they either need an expensive computational cost [7, 40, 44] or have a limited recommendation accuracy [33]. For example, Wu et al. [40] present a two-stage framework (IDCF) to learn the latent relational graph among existing users and new users, which requires a quadratic complexity. Shen et al. [33] use a global average embedding for new users, losing the personalization of recommendations for achieving the inductiveness. To sum up, there still lacks a both efficient and effective inductive collaborative filtering method.

In this paper, we formally define the inductive recommendation scenarios and address the aforementioned problems by proposing a novel **In**ductive Embedding **Mo**dule (INMO) for collaborative filtering. Specifically, INMO generates the inductive embedding for a user by considering its past interactions with a set of template items (vice versa), instead of learning a specific embedding for each user and item. As long as a new user (new item) has interacted with the preselected template items (template users), INMO could generate an informative embedding for the new user (new item). Besides, the number of parameters in INMO only depends on the number of template users and template items, which is adjustable according to available computing resources, contributing to its better scalability to real-world applications. Under the theoretical analysis, we further propose an effective indicator for the selection of template users and template items, making it possible for INMO to achieve competitive recommendation performances with much fewer model parameters.

Remarkably, our proposed INMO is model-agnostic, which can be easily attached to most of existing latent factor CF methods as a pre-module, inheriting the expressiveness of backbone models, while bringing the inductive ability and reducing model parameters. We experiment INMO with the classic Matrix Factorization (MF) [19] and the state-of-the-art LightGCN [9] to show its generality. Extensive experiments conducted on three public benchmark datasets, across both transductive and inductive recommendation scenarios, demonstrate the effectiveness of our proposed INMO.

To summarize, our contributions are as follows:

- We formally define two inductive recommendation scenarios to progressively evaluate the inductive ability of CF methods.
- We propose a novel **In**ductive Embedding **Mo**dule (INMO), which is applicable to existing latent factor models, bringing the inductive ability and reducing model parameters.
- Extensive experiments conducted on three real-world datasets across both transductive and inductive recommendation scenarios demonstrate the effectiveness and generality of INMO.

2 RELATED WORK

This section briefly reviews existing works of latent factor collaborative filtering methods and discusses several recent inductive recommenders, which are the most relevant to this work.

2.1 Latent Factor CF Methods

Latent factor models have been the state-of-the-art in collaborative filtering for over a decade [18]. These models learn vectorized embeddings for users and items by reconstructing the original user-item interaction data [13, 18]. From a systematic view, most existing latent factor models have two key components, i.e., representation generation and interaction modeling.

To improve the representation generation, many methods have been proposed to incorporate the external side information, like item attributes [2, 35], social networks [5, 23], knowledge graphs [26, 36], etc. However, high-quality side information is not always available in real-world applications. Recently, with the success of graph neural networks (GNNs) in various fields [1, 41], they have also been introduced into the collaborative filtering task [9, 31, 37, 42, 46]. These GNN-based methods could generate more comprehensive representations for users and items, capturing their high-order relationships by iteratively aggregating neighbor information in the user-item interaction graph. Among them, LightGCN [9] is a light but effective CF model, achieving the state-of-the-art performance.

As for interaction modeling, while the inner product is a commonly adopted and efficient choice [19, 29], the linearity makes it insufficient to reveal the complex and nonlinear interactions between users and items [12]. Some variants of MF [18, 19] add several bias terms to the inner product for better preference modeling. Hsieh et al. [14] employ the Euclidean distance instead of the inner product to estimate the similarities between users and items. He et al. [12] propose NeuMF, introducing a multi-layer perceptron (MLP) to learn the highly non-linear user-item interaction function.

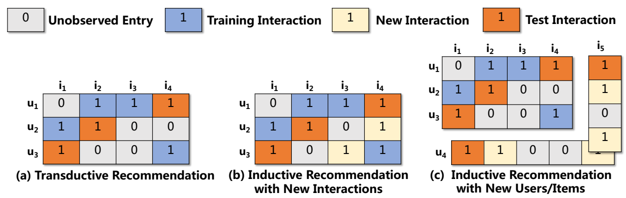

Figure 2: Traditional transductive recommendation scenario and two inductive scenarios proposed in this paper.

Following NeuMF, He et al. [10] use a convolutional neural network, modeling high-order correlations between representations.

However, despite the great success of latent factor models in CF, most existing methods rely on the embedding lookup table with user-specific and item-specific embeddings, making these models intrinsically transductive with the scalability difficulty.

2.2 Inductive CF Methods

In real-world applications, there are always new interactions, as well as newly registered users and items. Practical recommender systems need to be periodically retrained to refresh the models with the new data [45]. Though some works have discussed incremental and online recommenders to incorporate new users and new items [30, 43], they still bring an additional training cost or incremental error. As a result, it is a valuable property for the recommenders to possess the inductive ability, i.e., directly modeling additional new interactions and new users/items without retraining.

It is worth noting that, the inductive scenario is different from the *cold-start problem*. The cold-start methods focus on improving recommendations for users with few or no interactions [20, 22]. In contrast, the inductive scenario requires the recommenders to incorporate the new data and update their predictions without the need of retraining. An inductive recommender could predict the changeful preference of an existing user or a completely new user, according to its latest behaviors. In a word, the inductiveness discussed in this paper concentrates on the dynamic modeling capacity, instead of improving the experiences for long-tailed users.

Recently, a few works have discussed recommendations in the inductive scenarios [7, 27, 33, 40, 42, 44]. Ying et al. [42] propose PinSage, adopting an inductive variant of GNNs to make recommendations in an online content discovery service Pinterest. It leverages the visual and textual features of pins as inputs to achieve the inductiveness. However, high-quality side information is inadequate in many situations [44]. Hartford et al. [7] study the matrix completion in an inductive scenario, proposing an exchangeable matrix layer to do the message passing between interactions and use the numerical ratings as input features. But it is neither time-efficient nor can be applied to the implicit feedback data without real-valued ratings. Zhang and Chen [44] (IGMC) consider the rating prediction as a graph-level regression task. They define some heuristic node features and predict the rating of a user-item pair by learning the local graph pattern. In spite of its inductive ability, IGMC has to do subgraph extraction and graph regression for every user-item pair independently, resulting in an unaffordable time for the top-k recommendation task. Besides, Wu et al. [40] present a two-stage framework to estimate the relations from key users to query users, which takes a quadratic complexity. A very recent work IMC-GAE [33] generates the same embedding for all new users, leading to a poor performance.

To conclude, existing inductive CF methods are either time-consuming or have limited recommendation accuracy. There still lacks a both efficient and effective inductive method for the collaborative filtering task.

3 PRELIMINARIES

In this section, with a brief review of the commonly studied transductive recommendation task, we propose and formalize two inductive recommendation scenarios. Afterward, we introduce two representative latent factor methods for CF, i.e., MF [19] and LightGCN [9], which are the backbone models to apply INMO for experiments.

3.1 Transductive and Inductive CF Scenarios

Existing researches generally evaluate the performance of recommender systems in a transductive scenario. Specifically, they select a part of observed interactions from each user to serve as the training data, and treat the remaining interactions as the test data (Figure 2(a)). All users and items are assumed to be seen during training. However, in a real-world recommendation service, there are always newly registered users and newly created items, as well as new interactions between existing users and items. Such new interactions and new users/items emerge continuously and have not been seen during training. As a result, it is necessary to evaluate recommender systems from an inductive view, i.e., evaluating their ability to make recommendations for new users/items with new interactions that are unseen during the training phase.

In this work, we propose two specific scenario settings of inductive recommendations, which could better evaluate the dynamic modeling capability of recommender systems. The first is the **inductive recommendation with new interactions**, where some additional new interactions between existing users and items are observed after training, see Figure 2(b). Formally, $\mathcal{N}_\mathrm{u}$ denotes the set of interacted items of user u, and $\mathcal{N}_\mathrm{i}$ denotes the set of interacted users of item i. In the test phase, the extended interaction sets $\mathcal{N}_\mathrm{u}^{test-ob} = \mathcal{N}_\mathrm{u}^{train} \cup \mathcal{N}_\mathrm{u}^{new}, \mathcal{N}_\mathrm{i}^{test-ob} = \mathcal{N}_\mathrm{i}^{train} \cup \mathcal{N}_\mathrm{i}^{new}$ are observed for each user and item. In this scenario, it requires the recommenders to flexibly incorporate new interactions and predict the updated user preference without retraining.

The second scenario is the **inductive recommendation with new users/items**, where some new users U^{new} and new items I^{new} are created after the training phase. This scenario is different from the extreme cold start problem [44], since the new users and new items should have at least some observed interactions at the test phase, according to which the models could make accurate recommendations (Figure 2(c)). Formally,

$$U^{test} = U^{train} \cup U^{new},\ I^{test} = I^{train} \cup I^{new};$$
$$\forall \mathrm{u} \in U^{train},\ \mathcal{N}_\mathrm{u}^{test-ob} = \mathcal{N}_\mathrm{u}^{train} \cup \mathcal{N}_\mathrm{u}^{new};\ \forall \mathrm{u} \in U^{new},\ \mathcal{N}_\mathrm{u}^{test-ob} = \mathcal{N}_\mathrm{u}^{new};$$
$$\forall \mathrm{i} \in I^{train},\ \mathcal{N}_\mathrm{i}^{test-ob} = \mathcal{N}_\mathrm{i}^{train} \cup \mathcal{N}_\mathrm{i}^{new};\ \forall \mathrm{i} \in I^{new},\ \mathcal{N}_\mathrm{i}^{test-ob} = \mathcal{N}_\mathrm{i}^{new}.$$

The inductive evaluation in this scenario expects the recommenders to accurately recommend items for new users and recommend new items to all users, which is a prevalent need in practice.

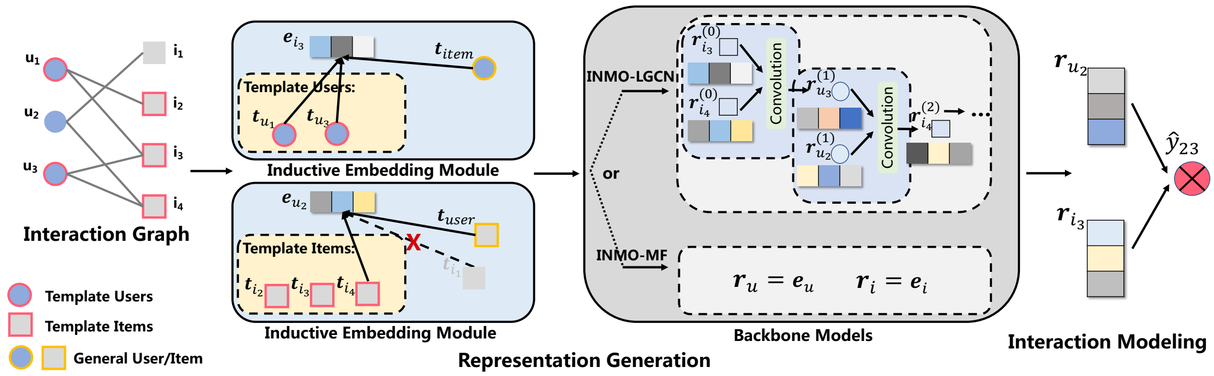

Figure 3: The overall architecture of an inductive CF recommender, which consists of two key components, i.e., representation generation and interaction modeling. Here, we use INMO implemented with MF (INMO-MF) and LightGCN (INMO-LGCN) as two specific instances. The template users (template items) are some carefully selected users (items) to serve as the bases to characterize each item (user) in the embedding space.

3.2 MF and LightGCN

Let $U = \{\mathrm{u}_1, \mathrm{u}_2, \cdots, \mathrm{u}_n\}$ and $I = \{\mathrm{i}_1, \mathrm{i}_2, \cdots, \mathrm{i}_m\}$ denote the set of users and items in a recommender system. Matrix Factorization (MF) [19] is the most basic latent factor model, which directly obtains the final representations of users and items from an embedding lookup table, i.e.,

$$\boldsymbol{r}_{\mathrm{u}} = \boldsymbol{e}_{\mathrm{u}},\ \boldsymbol{r}_{\mathrm{i}} = \boldsymbol{e}_{\mathrm{i}}. \tag{1}$$

Here, $\boldsymbol{e}_{\mathrm{u}}, \boldsymbol{e}_{\mathrm{i}} \in \mathbb{R}^d$ are the embeddings of user u and item i in an embedding lookup table, and $\boldsymbol{r}_{\mathrm{u}}, \boldsymbol{r}_{\mathrm{i}}$ are their final representations.

LightGCN [9] is a state-of-the-art CF recommender. After obtaining the initial representations $\boldsymbol{r}_{\mathrm{u}}^{(0)} = \boldsymbol{e}_{\mathrm{u}}$, $\boldsymbol{r}_{\mathrm{i}}^{(0)} = \boldsymbol{e}_{\mathrm{i}}$, it leverages a linear GNN to refine representations by iteratively aggregating the neighbor information in the user-item interaction graph, i.e.,

$$\boldsymbol{r}_{\mathrm{u}}^{(l+1)} = AGG(\{\boldsymbol{r}_{\mathrm{i}}^{(l)} : \mathrm{i} \in \mathcal{N}_{\mathrm{u}}\})\,, \tag{2}$$

where $\boldsymbol{r}_{\mathrm{u}}^{(l)}$ is the representation of user u at the l-th layer. The final representations are obtained by an average of all layers, i.e, $\boldsymbol{r}_{\mathrm{u}} = \frac{1}{K+1}\sum_{l=0}^{K}\boldsymbol{r}_{\mathrm{u}}^{(l)}$, $\boldsymbol{r}_{\mathrm{i}} = \frac{1}{K+1}\sum_{l=0}^{K}\boldsymbol{r}_{\mathrm{i}}^{(l)}$.

Both of MF and LightGCN employ a simple inner product to predict the preference score $\hat{y}_{ij}$ between user u_i and item i_j, based on their final representations, i.e., $\hat{y}_{ij} = \boldsymbol{r}_{\mathrm{u}_i}^T \boldsymbol{r}_{\mathrm{i}_j}$.

To attach our proposed INMO to MF or LightGCN, we just replace the transductive embedding lookup table with INMO, which can generate the embeddings $\boldsymbol{e}_{\mathrm{u}}$ and $\boldsymbol{e}_{\mathrm{i}}$ in an inductive manner.

4 METHODOLOGY

This section first introduces our intuitions behind the design of INMO, which provides both theoretical and empirical analyses. Then, we propose a novel **In**ductive Embedding **Mo**dule, to make recommendations in the inductive scenarios with an adjustable number of parameters. Lastly, an additional self-enhanced loss and two training techniques are presented to facilitate model optimization. The overall architecture of an inductive CF recommender with INMO is illustrated in Figure 3.

4.1 Theoretical Analysis

In the classic collaborative filtering setting without any side information, most existing latent factor models leverage an embedding lookup table, mapping the one-hot index of a user/item to an embedding vector. However, such an embedding lookup table is intrinsically transductive and brings the scalability difficulty. In this work, we aim to propose a scalable inductive embedding module, which could inductively generate the embeddings for new users and new items.

Before diving into our proposed method, let us first review the fundamental assumption of CF, i.e., if two users have similar past interactions with some items, they will act on other items similarly in the future [11, 24, 32, 34]. Based on such assumption, in this paper, we propose to design the inductive embedding module for users (items) by considering their past interactions with some carefully selected template items (template users). Such template items (template users) serve as a set of bases, the combination of which could represent different user preferences (item characteristics).

Let U_{tem}, I_{tem} denote the sets of template users and template items, and $\boldsymbol{T}_{\mathrm{u}} \in \mathbb{R}^{n_t \times d}, \boldsymbol{T}_{\mathrm{i}} \in \mathbb{R}^{m_t \times d}$ denote the template user vectors and template item vectors, where $n_t = |U_{tem}|$, $m_t = |I_{tem}|$. We expect to design two inductive functions f_{u} and f_{i}, to generate the embeddings for users and items according to their interactions with template items or template users, i.e., $\boldsymbol{e}_{\mathrm{u}} = f_{\mathrm{u}}(\mathcal{N}_{\mathrm{u}} \cap I_{tem}, \boldsymbol{T}_{\mathrm{i}})$ and $\boldsymbol{e}_{\mathrm{i}} = f_{\mathrm{i}}(\mathcal{N}_{\mathrm{i}} \cap U_{tem}, \boldsymbol{T}_{\mathrm{u}})$. Considering the recent finding that the nonlinear transformation adopted by neural networks is burdensome for the CF task [9, 38, 39], in this paper, we present a really simple yet theoretically effective design of f_{u} and f_{i}, which brings little optimization difficulty and has sufficient expressiveness from $\boldsymbol{T}_{\mathrm{u}}$ and $\boldsymbol{T}_{\mathrm{i}}$. Formally, we design that,

$$\boldsymbol{e}_{\mathrm{u}} = \sum_{\mathrm{i} \in \mathcal{N}_{\mathrm{u}} \cap I_{tem}} \boldsymbol{t}_{\mathrm{i}}, \quad \boldsymbol{e}_{\mathrm{i}} = \sum_{\mathrm{u} \in \mathcal{N}_{\mathrm{i}} \cap U_{tem}} \boldsymbol{t}_{\mathrm{u}}, \tag{3}$$

where $\boldsymbol{t}_{\mathrm{i}}$ is the template item vector of item i. Through such inductive functions, we can generate the embeddings for new users and new items which are unseen during the training phase.

We first theoretically prove the expressiveness of INMO in Eq. (3) based on a representative latent factor model and then present a both theoretically and empirically effective indicator to determine the sets of template users and template items.

Theorem 4.1. *Assuming the original MF can achieve a matrix factorization error ϵ on the interaction matrix Y, then there exists a solution for INMO-MF such that its error is less than or equal to ϵ, when INMO takes all users/items as the template users/items.*

Proof. Next, we theoretically illustrate that, when all users/items are selected as the template users/items, with the embeddings $\boldsymbol{e}_{\mathrm{u}}$ and $\boldsymbol{e}_{\mathrm{i}}$ generated from Eq. (3), the performance of INMO-MF would not be worse than the original MF.

Essentially, matrix factorization aims to do a low rank approximation on the interaction matrix, i.e., minimizing the difference between $\boldsymbol{Y}$ and $\boldsymbol{E}_{\mathrm{u}}\boldsymbol{E}_{\mathrm{i}}^T$, where $\boldsymbol{Y} \in \{0,1\}^{n\times m}$ is the observed interaction matrix between users and items. According to the Eckart-Young theorem, we can get $\boldsymbol{Y} = \boldsymbol{U}_{n\times p}\boldsymbol{S}_{p\times p}\boldsymbol{V}_{p\times m}^T = \boldsymbol{U}_{n\times d}^d\boldsymbol{S}_{d\times d}^d(\boldsymbol{V}^d)_{d\times m}^T + \boldsymbol{U}_{n\times(p-d)}^{\epsilon}\boldsymbol{S}_{(p-d)\times(p-d)}^{\epsilon}(\boldsymbol{V}^{\epsilon})_{(p-d)\times m}^T$, where $p = min(n,m)$. $\boldsymbol{S}$ is a diagonal matrix whose elements are singular values of $\boldsymbol{Y}$, and $\boldsymbol{U}, \boldsymbol{V}$ are column orthogonal matrices. $\boldsymbol{U}^d\boldsymbol{S}^d(\boldsymbol{V}^d)^T$ with d largest singular values in $\boldsymbol{S}^d$, is the closest rank-d matrix to $\boldsymbol{Y}$ in both Frobenius norm and spectral norm, denoted as $|Y - U^dS^d(V^d)^T|_F = \epsilon_{min}$, where $\epsilon_{min} = min_{rank(\hat{Y})=d}|Y - \hat{Y}|_F$ and $\hat{Y} = \boldsymbol{E}_{\mathrm{u}}\boldsymbol{E}_{\mathrm{i}}^T$ is the low rank approximation of MF.

Here, we show that with $\boldsymbol{E}_{\mathrm{u}} = \boldsymbol{Y}\boldsymbol{T}_{\mathrm{i}}, \boldsymbol{E}_{\mathrm{i}} = \boldsymbol{Y}^T\boldsymbol{T}_{\mathrm{u}}$ (the matrix form of Eq. (3)), INMO could learn the closest rank-d solution of $\boldsymbol{Y}$. In other words, with the same embedding dimension, INMO-MF can achieve the error ϵ_{min} which is the minimum possible error obtained by any solutions of MF. Specifically, there exists a solution for INMO-MF as $\boldsymbol{T}_{\mathrm{u}} = \boldsymbol{U}^d(\boldsymbol{S}^d)^{-1}, \boldsymbol{T}_{\mathrm{i}} = \boldsymbol{V}^d$, that has

$$
\begin{aligned}
|\boldsymbol{Y} - \boldsymbol{E}_{\mathrm{u}}\boldsymbol{E}_{\mathrm{i}}^T|_F &= |\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{V}^d(\boldsymbol{S}^d)^{-1}(\boldsymbol{U}^d)^T\boldsymbol{Y}|_F \\
&= |\boldsymbol{Y} - \boldsymbol{U}^d(\boldsymbol{U}^d)^T\boldsymbol{Y}|_F = |\boldsymbol{Y} - \boldsymbol{U}^d\boldsymbol{S}^d(\boldsymbol{V}^d)^T|_F = \epsilon_{min}.
\end{aligned} \tag{4}
$$

□

The above proof validates that, our **INMO-MF has at least the same expressiveness as the original MF** while being capable to make inductive recommendations. A similar proof could be deduced for INMO-LGCN, which is a linear model as well.

The next important question is how to carefully select the template users and template items in order to reduce model parameters while avoiding the additional error as much as possible. When we only select a part of users and items as the template ones, the INMO in Eq. (3) can be written as $\boldsymbol{E}_{\mathrm{u}} = \boldsymbol{Y}_{n\times m}(\boldsymbol{C}_{\mathrm{i}})_{m\times m_t}(\boldsymbol{T}_{\mathrm{i}})_{m_t\times d}$. Each column of $\boldsymbol{C}_{\mathrm{i}}$ is an one-hot vector, and each row of $\boldsymbol{C}_{\mathrm{i}}$ has at most one non-zero entry. $(\boldsymbol{C}_{\mathrm{i}})_{i,j} = 1$ means that the i-th item i_i is selected as the j-th template item $(\mathrm{i}_{tem})_j$. Similarly, $\boldsymbol{E}_{\mathrm{i}} = \boldsymbol{Y}_{m\times n}^T(\boldsymbol{C}_{\mathrm{u}})_{n\times n_t}(\boldsymbol{T}_{\mathrm{u}})_{n_t\times d}$. The number of model parameters in INMO is now $(n_t + m_t)d$, which is much smaller than the original embedding table with $(n + m)d$ parameters.

Theorem 4.2. *When selecting those users u_j with the largest $|s_j^{\mathrm{u}}|_2^2\sum_{\mathrm{i}\in\mathcal{N}_{\mathrm{u}_j}}|\mathcal{N}_{\mathrm{i}}|$ as the template users, INMO minimizes an upper bound of the additional error caused by ignoring non-template users.*

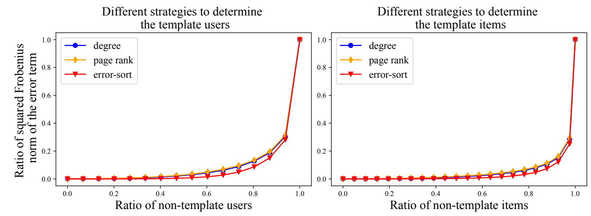

Figure 4: The ratio of squared Frobenius norm of the error term under different ratios of non-template users/items.

Proof. Similar to the proof in Theorem 4.1, a solution of INMO-MF can be written as $\boldsymbol{T}_{\mathrm{u}} = \boldsymbol{C}_{\mathrm{u}}^T\boldsymbol{U}^d(\boldsymbol{S}^d)^{-1}, \boldsymbol{T}_{\mathrm{i}} = \boldsymbol{C}_{\mathrm{i}}^T\boldsymbol{V}^d$, where $\boldsymbol{C}_{\mathrm{u}}, \boldsymbol{C}_{\mathrm{i}}$ indicate the selected template users and template items. Then,

$$
\begin{aligned}
|\boldsymbol{Y} - \boldsymbol{E}_{\mathrm{u}}\boldsymbol{E}_{\mathrm{i}}^T|_F &= |\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{C}_{\mathrm{i}}\boldsymbol{C}_{\mathrm{i}}^T\boldsymbol{V}^d(\boldsymbol{S}^d)^{-1}(\boldsymbol{U}^d)^T\boldsymbol{C}_{\mathrm{u}}\boldsymbol{C}_{\mathrm{u}}^T\boldsymbol{Y}|_F \\
&= |\boldsymbol{Y} - \boldsymbol{Y}(\boldsymbol{E} - \boldsymbol{L}_{\mathrm{i}})\boldsymbol{V}^d(\boldsymbol{S}^d)^{-1}(\boldsymbol{U}^d)^T(\boldsymbol{E} - \boldsymbol{L}_{\mathrm{u}})\boldsymbol{Y}|_F \\
&= |\boldsymbol{Y} - \boldsymbol{U}^d\boldsymbol{S}^d(\boldsymbol{V}^d)^T - \boldsymbol{Y}\boldsymbol{L}_{\mathrm{i}}\boldsymbol{V}^d(\boldsymbol{S}^d)^{-1}(\boldsymbol{U}^d)^T\boldsymbol{L}_{\mathrm{u}}\boldsymbol{Y} \\
&\quad + \boldsymbol{U}^d(\boldsymbol{U}^d)^T\boldsymbol{L}_{\mathrm{u}}\boldsymbol{Y} + \boldsymbol{Y}\boldsymbol{L}_{\mathrm{i}}\boldsymbol{V}^d(\boldsymbol{V}^d)^T|_F,
\end{aligned} \tag{5}
$$

where $\boldsymbol{L}_{\mathrm{u}} \in \{0,1\}^{n\times n}$ is a diagonal matrix and $(\boldsymbol{L}_{\mathrm{u}})_{i,i} = 1$ means that u_i is a non-template user. For simplicity, we only consider the non-template users and assume $\boldsymbol{L}_{\mathrm{i}} = \boldsymbol{0}$. Then,

$$
\begin{aligned}
|\boldsymbol{Y} - \boldsymbol{E}_{\mathrm{u}}\boldsymbol{E}_{\mathrm{i}}^T|_F &= |\boldsymbol{Y} - \boldsymbol{U}^d\boldsymbol{S}^d(\boldsymbol{V}^d)^T + \boldsymbol{U}^d(\boldsymbol{U}^d)^T\boldsymbol{L}_{\mathrm{u}}\boldsymbol{Y}|_F \\
&\leq |\boldsymbol{Y} - \boldsymbol{U}^d\boldsymbol{S}^d(\boldsymbol{V}^d)^T|_F + |\boldsymbol{U}^d(\boldsymbol{U}^d)^T\boldsymbol{L}_{\mathrm{u}}\boldsymbol{Y}|_F \\
&= \epsilon_{min} + |\boldsymbol{U}^d(\boldsymbol{U}^d)^T\boldsymbol{L}_{\mathrm{u}}\boldsymbol{Y}|_F.
\end{aligned} \tag{6}
$$

To minimize the norm of the additional error matrix $\boldsymbol{U}^d(\boldsymbol{U}^d)^T\boldsymbol{L}_{\mathrm{u}}\boldsymbol{Y}$, let $\boldsymbol{U}^d(\boldsymbol{U}^d)^T = (s_1^{\mathrm{u}}, s_2^{\mathrm{u}}, \cdots, s_n^{\mathrm{u}})$ and $U_{non-tem}$ denotes the set of non-template users. Then,

$$
\boldsymbol{U}^d(\boldsymbol{U}^d)^T\boldsymbol{L}_{\mathrm{u}}\boldsymbol{Y} = (\sum_{\mathrm{u}_j\in\mathcal{N}_{\mathrm{i}_1}\cap U_{non-tem}} s_j^{\mathrm{u}}, \cdots, \sum_{\mathrm{u}_j\in\mathcal{N}_{\mathrm{i}_m}\cap U_{non-tem}} s_j^{\mathrm{u}}) \tag{7}
$$

$$
\begin{aligned}
|\boldsymbol{U}^d(\boldsymbol{U}^d)^T\boldsymbol{L}_{\mathrm{u}}\boldsymbol{Y}|_F^2 &= \sum_{\mathrm{i}\in I}|\sum_{\mathrm{u}_j\in\mathcal{N}_{\mathrm{i}}\cap U_{non-tem}} s_j^{\mathrm{u}}|_2^2 \\
&\leq \sum_{\mathrm{i}\in I}|\mathcal{N}_{\mathrm{i}}\cap U_{non-tem}|\sum_{\mathrm{u}_j\in\mathcal{N}_{\mathrm{i}}\cap U_{non-tem}}|s_j^{\mathrm{u}}|_2^2 \\
&\leq \sum_{\mathrm{u}_j\in U_{non-tem}}|s_j^{\mathrm{u}}|_2^2\sum_{\mathrm{i}\in\mathcal{N}_{\mathrm{u}_j}}|\mathcal{N}_{\mathrm{i}}|
\end{aligned} \tag{8}
$$

Apparently, the above error upper bound can be minimized by selecting the template users with the largest $|s_j^{\mathrm{u}}|_2^2\sum_{\mathrm{i}\in\mathcal{N}_{\mathrm{u}_j}}|\mathcal{N}_{\mathrm{i}}|$. □

Next, we conduct an empirical experiment on a real-world dataset Gowalla, to validate the effectiveness of the theoretical indicator $|s_j^{\mathrm{u}}|_2^2\sum_{\mathrm{i}\in\mathcal{N}_{\mathrm{u}_j}}|\mathcal{N}_{\mathrm{i}}|$ to select the template users ($|s_j^{\mathrm{i}}|_2^2\sum_{\mathrm{u}\in\mathcal{N}_{\mathrm{i}_j}}|\mathcal{N}_{\mathrm{u}}|$ to select the template items), namely the **error-sort** indicator. We compare our proposed error-sort with the other two heuristic indicators, i.e., node degree and page rank [25].

Figure 4 demonstrates how the additional error $|\boldsymbol{U}^d(\boldsymbol{U}^d)^T\boldsymbol{L}_{\mathrm{u}}\boldsymbol{Y}|_F^2$ ($|\boldsymbol{Y}\boldsymbol{L}_{\mathrm{i}}\boldsymbol{V}^d(\boldsymbol{V}^d)^T|_F^2$) changes with the number of non-template users (non-template items). There are two important findings. First, we can ignore 70% of users/items and set them as the non-template ones, which only leads to an additional error smaller than 10%. Besides, we notice that the error-sort indicator based on Theorem 4.2 produces a smaller error ratio, which validates the superiority of our proposed error-sort compared with the heuristic indicators.

4.2 Inductive Embedding Generation

Based on our above analyses, we now introduce the specific design of INMO. According to Figure 4, most of the users/items could be ignored as the non-template ones. When obtaining the embeddings of users and items, INMO only utilizes their interactions with the template items I_{tem} or template users U_{tem}. To stabilize the training procedure, we add a denominator to Eq. (3), adjusting the norms of embeddings. Formally,

$$\begin{aligned} \boldsymbol{e}_{\mathrm{u}} &= \frac{1}{(|\mathcal{N}_{\mathrm{u}} \cap I_{tem}|+1)^{\alpha}} \Big(\sum_{\mathrm{i} \in \mathcal{N}_{\mathrm{u}} \cap I_{tem}} \boldsymbol{t}_{\mathrm{i}} + \boldsymbol{t}_{user} \Big) \quad , \\ \boldsymbol{e}_{\mathrm{i}} &= \frac{1}{(|\mathcal{N}_{\mathrm{i}} \cap U_{tem}|+1)^{\alpha}} \Big(\sum_{\mathrm{u} \in \mathcal{N}_{\mathrm{i}} \cap U_{tem}} \boldsymbol{t}_{\mathrm{u}} + \boldsymbol{t}_{item} \Big) \quad , \end{aligned} \tag{9}$$

where $\boldsymbol{e}_{\mathrm{u}}, \boldsymbol{e}_{\mathrm{i}}$ denote the inductive embeddings of user u and item i. Here, we use $\boldsymbol{t}_{user}$ and $\boldsymbol{t}_{item}$ to model the global characteristics of users and items, which may help make recommendations for new users and new items with few observed interactions. α is an exponent controlling the degree of normalization, which we will discuss in Section 4.3.2.

With the additional denominator, the indicator error-sort should be fine-tuned as $|\boldsymbol{s}_j'^{\mathrm{u}}|_2^2 \sum_{\mathrm{i} \in \mathcal{N}_{\mathrm{u}_j}} 1/|\mathcal{N}_{\mathrm{i}}|$, where $\boldsymbol{s}_j'^{\mathrm{u}}$ is the j_{th} column of $\boldsymbol{D}_{\mathrm{u}}^{-1}\boldsymbol{U}^d(\boldsymbol{U}^d)^T$ and $\boldsymbol{D}_{\mathrm{u}}$ is the diagonal degree matrix of users. In this case, $|\boldsymbol{s}_j'^{\mathrm{u}}|$ will highlight the importance of low-degree users, which may introduce some noises in practice. Moreover, the calculation of $\boldsymbol{U}^d(\boldsymbol{U}^d)^T$ is expensive. Therefore, we only implement a simplified version of **error-sort**, i.e., sorting users by $\sum_{\mathrm{i} \in \mathcal{N}_{\mathrm{u}}} 1/|\mathcal{N}_{\mathrm{i}}|$.

Figure 3 shows an example user-item interaction graph, where red circles denote template users and red squares denote template items. For user u_2, its embedding is aggregated by both the template item vector $\boldsymbol{t}_{\mathrm{i}_4}$ and the global user template $\boldsymbol{t}_{user}$. Note that, although item i_1 also interacts with user u_2, according to our error-sort indicator, we neither learn the template vector of i_1 nor use it to represent users.

The proposed INMO has two major advantages: 1) **Inductive ability**. When facing new users and new items after training, INMO can obtain their embeddings without the need of retraining. 2) **Adjustable scalability**. The number of parameters in INMO is only dependent on the number of template users and template items, which is adjustable according to available computing resources.

Complexity Analysis. The time complexity of INMO to generate the inductive embedding for user u is $O(|\mathcal{N}_{\mathrm{u}} \cap I_{tem}|)$, which is affordable in most cases. In contrast, IDCF [40] takes the $O(n)$ complexity to pass messages from key users and IGMC [44] needs to do the graph regression task $O(m)$ times to recommend for a single user. In general cases, $|\mathcal{N}_{\mathrm{u}} \cap I_{tem}|$ is much smaller than n and m, because a user always interacts with a limited number of items. The space complexity of INMO is $O((n_t + m_t)d)$, as we only need to save the template user vectors and template item vectors. The embeddings of other users and items can be generated on the fly, which is memory-efficient.

4.3 Model Optimization

4.3.1 Loss Function. A commonly adopted training loss for top-k CF methods is the pairwise Bayesian Personalized Ranking (BPR) loss [28]. It encourages the model to predict a higher score for an observed entry than an unobserved entry. The loss function formulates as follows,

$$\mathcal{L}_{BPR} = -\sum_{\mathrm{u}_i \in U} \sum_{\mathrm{i}_j \in \mathcal{N}_{\mathrm{u}_i}} \sum_{\mathrm{i}_k \notin \mathcal{N}_{\mathrm{u}_i}} ln\, \sigma(\boldsymbol{r}_{\mathrm{u}_i}^T \boldsymbol{r}_{\mathrm{i}_j} - \boldsymbol{r}_{\mathrm{u}_i}^T \boldsymbol{r}_{\mathrm{i}_k}) + \lambda ||\Theta||_2^2 \quad , \tag{10}$$

where λ controls the L_2 regularization strength and Θ denotes all trainable model parameters.

To facilitate the optimization of template user vectors and template item vectors, we propose an additional self-enhanced loss $\mathcal{L}_{SE}$. Intuitively, the supervised BPR loss on the final representations $\boldsymbol{r}_{\mathrm{u}}, \boldsymbol{r}_{\mathrm{i}}$ may be not enough, since the template users/items are aggregated together and lose their identities when generating the inductive embeddings $\boldsymbol{e}_{\mathrm{i}}/\boldsymbol{e}_{\mathrm{u}}$. So we design a new supervised signal to directly guide the learning process of $\boldsymbol{t}_{\mathrm{u}}, \boldsymbol{t}_{\mathrm{i}}$ for template users and template items, enhancing their identities,

$$\mathcal{L}_{SE} = -\sum_{\mathrm{u}_i \in U_{tem}} \sum_{\mathrm{i}_j \in \mathcal{N}_{\mathrm{u}_i} \cap I_{tem}} \sum_{\mathrm{i}_k \in \overline{\mathcal{N}_{\mathrm{u}_i}} \cap I_{tem}} ln\, \sigma(\boldsymbol{t}_{\mathrm{u}_i}^T \boldsymbol{W}_s \boldsymbol{t}_{\mathrm{i}_j} - \boldsymbol{t}_{\mathrm{u}_i}^T \boldsymbol{W}_s \boldsymbol{t}_{\mathrm{i}_k}) \ . \tag{11}$$

The final loss is $\mathcal{L} = \mathcal{L}_{BPR} + \beta \mathcal{L}_{SE}$, where β is a hyper-parameter to balance the BPR loss and the self-enhanced loss.

From another perspective, $\mathcal{L}_{SE}$ is actually the BPR loss, but on the template user vectors and template item vectors $\boldsymbol{t}$, instead of the final representations $\boldsymbol{r}$. Consistent with the analysis in Section 4.1, the optimal solution is $\boldsymbol{E}_{\mathrm{u}} = \boldsymbol{U}^d \boldsymbol{S}^d$, $\boldsymbol{E}_{\mathrm{i}} = \boldsymbol{V}^d$ for MF and $\boldsymbol{T}_{\mathrm{u}} = \boldsymbol{U}^d(\boldsymbol{S}^d)^{-1}, \boldsymbol{T}_{\mathrm{i}} = \boldsymbol{V}^d$ for INMO-MF. Thus, we add a learnable diagonal matrix $\boldsymbol{W}_s$ to model this difference.

4.3.2 Normalization Annealing. Considering the exponent of normalization α, we find that constantly setting it to 1 may overly punish the weights of active users with long interaction histories. Especially in the early stage of training, hard normalization ($\alpha = 1$) may lead to a slow convergence, which is consistent with the findings in [11]. Consequently, we adopt an annealing strategy to dynamically control the degree of normalization. Specifically, in the training phase, α is first set to an initial value of 0.5 and will be gradually increased to 1. In the test phase, α is fixed to 1. At the beginning of training, the embeddings of active users would be trained better than non-active users as they have more training data. Therefore, increasing the norms of active users, that is, making the early normalization exponent α smaller than 1, could temporarily emphasize active users and accelerate the training procedure.

4.3.3 Drop Interaction. Since we expect the learned model to be inductive, it is supposed to make recommendations for new users with unseen combinations of past interactions. For this reason, we propose to randomly drop the interaction sets $\mathcal{N}_u, \mathcal{N}_{\mathrm{i}}$ with a certain probability during training as an approach of data augmentation, which also prevents the model from over-fitting. In this way, INMO could see varying combinations of past interactions from a single user in different training steps, improving its inductive ability.

5 EXPERIMENTS

To evaluate the effectiveness of our proposed inductive embedding module, we conduct extensive experiments on three public real-world datasets. Specifically, the experiments are intended to answer the following three questions:

Table 1: Statistics of the datasets.

Dataset	#Users	#Items	#Interactions	Density
Gowalla	29,858	40,988	1,027,464	0.00084
Yelp	75,173	42,706	1,931,173	0.00060
Amazon-Book	109,730	96,421	3,181,759	0.00030

- **Q1**: How does INMO perform in the transductive scenario as compared with the embedding lookup table?
- **Q2**: How strong is the inductive ability of INMO in inductive scenarios with new interactions and new users/items?
- **Q3**: How do different hyper-parameters (i.e., the strength of the self-enhanced loss and the number of template users/items) and training techniques (i.e., normalization annealing and drop interaction) affect the performance of INMO?

Next, we introduce the specific experimental settings and present detailed experimental analyses to each question.

5.1 Experimental Settings

5.1.1 Dataset. We conduct experiments on three public benchmark datasets: **Gowalla**[1] [3] , **Yelp**[2], and **Amazon-Book**[3] [8], where the items are locations, local businesses, and books respectively. For Yelp and Amazon-book, we regard the ratings greater than 3 as observed interactions and filter out users and items with less than 10 interactions, similar to the pre-processing procedure in [21]. For each user, we randomly split its interactions into 70%, 10%, and 20% as the train, validation, and test sets. The experiments are repeated five times with different dataset splits, and the average results with standard deviations are reported. The characteristics of the three datasets are summarized in Table 1.

5.1.2 Baseline Methods. We implement our INMO with the classic MF (INMO-MF) and the state-of-the-art LightGCN (INMO-LGCN) to explore how INMO improves the recommendation accuracy and brings the inductive ability. We compare our methods with the following baselines:

- **MF-BPR** [19]: It is the most basic latent factor model optimized by BPR loss, yielding competitive performances in many cases.
- **NeuMF** [12]: It explores the nonlinear interactions between the representations of users and items by a fusion of Hadamard product and MLP.
- **Mult-VAE** [21]: It employs the variational autoencoder architecture [16] to encode and decode users' interaction behaviors.
- **NGCF** [37]: It is a GNN-based recommender model containing the feature transformation and the nonlinear activation like the standard graph convolutional network (GCN) [17].
- **LightGCN** [9]: It is a simplified version of GCN for recommendation, which linearly propagates the representations and achieves the state-of-the-art transductive performance.
- **IMC-GAE** [33]: It employs a graph autoencoder with a postprocessing strategy for the inductive rating prediction. We adapt it to do the implicit top-k recommendation task.
- **IDCF-LGCN** [40]: IDCF is a two-stage inductive recommendation framework, estimating the underlying relations from key users to query users. We implement it with LightGCN to recommend for both new users and new items.

5.1.3 Evaluation Metrics. We evaluate the effectiveness of CF methods on predicting users' preferences as a ranking problem. Specifically, three widely-used evaluation metrics for top-k recommender systems are adopted, i.e., $recall@k$, $precision@k$, and $NDCG@k$. We set $k = 20$ following [37] and report the average values over all users in the test set. For clarity, we show all the metrics after multiplied by 100 in the tables of this paper, similar to [11].

5.1.4 Parameter Settings. We implement our INMO-MF, INMO-LGCN, and all other baseline methods based on Pytorch. The codes including dataset processing, hyper-parameter tuning, and model implementations are accessible here [4]. All models are learned via optimizing the BPR loss, except that NeuMF uses the binary cross-entropy loss and Mult-VAE maximizes the multinomial likelihood as proposed in their original papers. We use the Adam optimizer [15] to train all the models for at most 1000 epochs. The embedding size of different models is fixed to 64 for a fair comparison, and the batch size is fixed to 2048. We apply a grid search on the validation set, tuning hyper-parameters for INMO and other baseline methods, the learning rate is tuned over $\{10^{-4}, 10^{-3}, 10^{-2}\}$, the L_2 regularization coefficient over $\{0, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$, and the dropout rate over $\{0, 0.1, 0.3, 0.5, 0.7, 0.9\}$. We set the number of graph convolution layers to three for all graph-based methods. The sampling size of IDCF-LGCN is set to 50 for an affordable time consumption. Moreover, the early stopping strategy is performed, i.e., stopping training if $NDCG@20$ on the validation data does not increase for 50 successive epochs. To comprehensively demonstrate the effectiveness of our method, we implement two specific versions of INMO, denoted as INMO-MF, INMO-LGCN and INMO-MF*, INMO-LGCN*. Specifically, INMO-MF and INMO-LGCN take all users/items as the template users/items, while INMO-MF* and INMO-LGCN* take only 30% of users as the template users and 30% of items as the template items. By default, the template users and template items are selected by the error-sort indicator. A detailed analysis about the number of template users and template items is provided in Section 5.4.1.

5.2 Transductive Recommendation (Q1)

We first conduct traditional transductive experiments to demonstrate the general effectiveness of our proposed INMO. Table 2 shows the recommendation results of all methods in the transductive scenario [5]. The results include seven baseline methods, and four models using our INMO, including INMO-MF, INMO-MF*, INMO-LGCN, and INMO-LGCN*. We have the following observations:

- Our implemented INMO-LGCN model outperforms all other baseline methods in the transductive recommendation scenario, beating state-of-the-art recommenders Mult-VAE and LightGCN.
- Both of our implemented INMO-MF and INMO-LGCN significantly outperform their basic versions MF and LightGCN, which

[1] https://snap.stanford.edu/data/loc-Gowalla.html
[2] https://www.yelp.com/dataset
[3] http://jmcauley.ucsd.edu/data/amazon/

[4] https://github.com/WuYunfan/igcn_cf
[5] As we run 5 times on different dataset splits, our results are a little different from those reported in [9].

Table 2: Overall performances in the transductive recommendation scenario.

	Gowalla			Yelp			Amazon-book		
	Recall@20	Precision@20	NDCG@20	Recall@20	Precision@20	NDCG@20	Recall@20	Precision@20	NDCG@20
NeuMF	15.08(±0.07)	3.88(±0.02)	11.20(±0.06)	7.94(±0.05)	1.75(±0.01)	4.87(±0.04)	\	\	\
Mult-VAE	18.36(±0.08)	4.80(±0.03)	13.90(±0.06)	10.13(±0.04)	2.24(±0.00)	6.49(±0.02)	13.63(±0.05)	2.81(±0.01)	9.44(±0.05)
NGCF	17.36(±0.08)	4.48(±0.04)	12.77(±0.13)	8.72(±0.22)	1.82(±0.28)	5.31(±0.37)	12.04(±0.13)	2.42(±0.03)	7.89(±0.08)
IMC-GAE	15.42(±0.09)	3.98(±0.03)	11.63(±0.05)	6.43(±0.05)	1.41(±0.01)	4.00(±0.04)	8.24(±0.04)	1.76(±0.01)	5.43(±0.03)
IDCF-LGCN	12.80(±0.23)	3.43(±0.05)	9.60(±0.15)	5.49(±0.14)	1.32(±0.02)	3.55(±0.08)	\	\	\
MF-BPR	16.11(±0.08)	4.14(±0.02)	12.03(±0.07)	8.61(±0.15)	1.90(±0.03)	5.36(±0.09)	11.41(±0.13)	2.28(±0.03)	7.37(±0.11)
INMO-MF	18.41(±0.10)	4.92(±0.02)	14.15(±0.10)	9.54(±0.03)	2.10(±0.01)	6.07(±0.03)	13.40(±0.06)	2.81(±0.01)	9.23(±0.04)
INMO-MF*	16.20(±0.09)	4.41(±0.03)	12.40(±0.08)	9.09(±0.09)	2.01(±0.01)	5.81(±0.05)	11.88(±0.05)	2.53(±0.01)	8.13(±0.04)
LightGCN	18.88(±0.13)	4.95(±0.04)	14.18(±0.10)	9.68(±0.08)	2.14(±0.01)	6.16(±0.06)	12.32(±0.10)	2.56(±0.02)	8.19(±0.06)
INMO-LGCN	**20.17**(±0.11)	**5.36**(±0.04)	**15.41**(±0.10)	**10.26**(±0.03)	**2.25**(±0.01)	**6.51**(±0.02)	**14.28**(±0.05)	**3.01**(±0.01)	**9.86**(±0.03)
INMO-LGCN*	19.58(±0.09)	5.21(±0.03)	14.96(±0.06)	10.21(±0.04)	2.23(±0.01)	6.47(±0.03)	13.77(±0.04)	2.92(±0.01)	9.53(±0.04)

'\': These methods cannot deal with the large dataset Amazon-book.

Table 3: Performances in the inductive recommendation scenario with new interactions.

	Gowalla			Yelp			Amazon-book		
	Recall@20	Precision@20	NDCG@20	Recall@20	Precision@20	NDCG@20	Recall@20	Precision@20	NDCG@20
INMO-LGCN-retrain	20.17(±0.11)	5.36(±0.04)	15.41(±0.10)	10.26(±0.03)	2.25(±0.01)	6.51(±0.02)	14.28(±0.05)	3.01(±0.01)	9.86(±0.03)
Mult-VAE	16.68(±0.08)	4.40(±0.01)	12.60(±0.05)	9.16(±0.06)	2.06(±0.01)	5.86(±0.03)	11.79(±0.06)	2.46(±0.01)	8.11(±0.04)
Mult-VAE-new	17.03(±0.06)	4.47(±0.02)	12.89(±0.03)	9.49(±0.04)	**2.11**(±0.01)	**6.07**(±0.02)	12.30(±0.05)	2.55(±0.01)	8.46(±0.02)
IMC-GAE	14.02(±0.05)	3.66(±0.02)	10.61(±0.07)	5.68(±0.04)	1.24(±0.01)	5.57(±0.03)	6.80(±0.03)	1.50(±0.01)	4.52(±0.03)
IMC-GAE-new	14.39(±0.09)	3.74(±0.03)	10.91(±0.09)	5.85(±0.06)	1.28(±0.02)	3.68(±0.04)	7.22(±0.02)	1.58(±0.01)	4.80(±0.03)
LightGCN	17.66(±0.24)	4.67(±0.06)	13.41(±0.15)	8.58(±0.07)	1.93(±0.02)	5.48(±0.06)	10.67(±0.07)	2.27(±0.02)	7.14(±0.06)
LightGCN-new	17.95(±0.24)	4.73(±0.06)	13.53(±0.16)	8.96(±0.08)	1.98(±0.02)	5.69(±0.06)	11.65(±0.08)	2.46(±0.02)	7.80(±0.07)
IDCF-LGCN	12.54(±0.21)	3.39(±0.07)	9.45(±0.17)	5.16(±0.06)	1.26(±0.01)	3.34(±0.03)	\	\	\
IDCF-LGCN-new	12.93(±0.29)	3.46(±0.08)	9.69(±0.20)	5.43(±0.08)	1.31(±0.02)	3.52(±0.05)	\	\	\
INMO-MF	16.66(±0.08)	4.49(±0.03)	12.81(±0.12)	8.45(±0.05)	1.89(±0.01)	5.37(±0.03)	11.30(±0.04)	2.42(±0.00)	7.76(±0.01)
INMO-MF-new	17.45(±0.08)	4.65(±0.04)	13.37(±0.13)	8.98(±0.06)	1.97(±0.01)	5.70(±0.04)	12.21(±0.06)	2.57(±0.01)	8.32(±0.07)
INMO-LGCN	18.25(±0.05)	4.92(±0.02)	14.03(±0.06)	9.23(±0.06)	2.05(±0.01)	5.87(±0.04)	12.16(±0.05)	2.64(±0.01)	8.39(±0.01)
INMO-LGCN-new	**19.21**(±0.04)	**5.05**(±0.02)	**14.51**(±0.03)	9.58(±0.09)	2.07(±0.01)	6.03(±0.05)	**13.44**(±0.04)	**2.79**(±0.00)	**9.02**(±0.03)
INMO-LGCN*-new	18.61(±0.19)	4.92(±0.05)	14.13(±0.15)	**9.63**(±0.07)	2.07(±0.01)	6.06(±0.05)	13.01(±0.07)	2.71(±0.02)	8.76(±0.06)

employ a traditional embedding lookup table. It suggests the superiority of INMO in the transductive recommendation scenario, which can generate more accurate recommendation results.

- INMO-MF* and INMO-LGCN* show better performances than MF and LightGCN, while with only 30% of parameters, indicating the potential of our INMO in resource limited applications.

In INMO-MF and INMO-LGCN, two users with the same historical behaviors will obtain the same embedding, i.e., the embedding mapping function is injective. While in their original versions (MF and LightGCN), these two users may have different recommended items, owing to their randomly initialized individual embeddings. Such injective property may further help the recommenders to make the most of the training data and reduce noises, leading to better performances.

5.3 Inductive Recommendation (Q2)

A great advantage of our INMO lies in its capability to model new interactions and new users/items in the test phase without the need of retraining. Thus, we conduct experiments in two inductive recommendation scenarios with *new interactions* and *new users/items*, as described in Section 3.1.

5.3.1 New Interactions. In this scenario, we randomly remove 20% of training interactions from each user, then train the recommender models on the remaining data. During the test phase, previously removed interactions arrive as the new interactions, which can be further utilized to improve the recommendation performances.

Note that not all of the baseline methods can handle this inductive scenario. MF-based methods (MF and NeuMF) cannot make recommendations with new interactions or new users/items. We adapt Mult-VAE to this scenario by adding the new interactions to its updated inputs of the encoder. GNN-based CF methods, i.e., NGCF, LightGCN, and IMC-GAE, take the new interactions into consideration via adding new links in the interaction graph. As for our INMO, it updates the inductive embeddings of users and items, enhancing the utilization of new interactions.

We present the experimental results in Table 3. The suffix **-new** indicates the updated results considering additional new interactions in the test phase, otherwise not. **INMO-LGCN-retrain** refers to the performances of INMO-LGCN through retraining to incorporate the new interactions, served as the performance upper bound. As shown in Table 3, new interactions help to improve the performances for all methods, verifying the benefits of modeling additional new interactions in the test phase. Both INMO-MF and

Table 4: Performances in the inductive recommendation scenario with new users and new items.

	Gowalla			Yelp			Amazon-book		
	New User	New Item	Over All	New User	New Item	Over All	New User	New Item	Over All
INMO-LGCN-retrain	14.01(±0.42)	16.20(±0.20)	15.41(±0.10)	6.21(±0.07)	13.21(±0.10)	6.51(±0.02)	14.73(±0.12)	14.91(±0.19)	9.86(±0.03)
Popular	1.54(±0.12)	0.91(±0.06)	2.10(±0.03)	1.04(±0.03)	1.91(±0.05)	1.01(±0.02)	0.55(±0.02)	0.82(±0.03)	0.70(±0.01)
Mult-VAE	10.77(±0.31)	\	12.58(±0.08)	4.93(±0.07)	\	5.70(±0.05)	**9.12**(±0.06)	\	7.87(±0.04)
IMC-GAE	8.42(±0.15)	9.25(±0.21)	9.81(±0.08)	2.57(±0.03)	7.54(±0.11)	3.25(±0.03)	4.91(±0.35)	5.12(±0.15)	4.19(±0.05)
IMC-LGCN	10.38(±0.31)	10.90(±0.09)	13.24(±0.12)	4.67(±0.04)	10.74(±0.08)	5.57(±0.03)	7.18(±0.19)	7.07(±0.16)	6.39(±0.07)
IDCF-LGCN	8.29(±0.18)	8.60(±0.13)	9.40(±0.09)	3.28(±0.04)	7.77(±0.10)	3.48(±0.06)	\	\	\
INMO-MF	10.85(±0.24)	10.92(±0.14)	13.10(±0.06)	4.85(±0.24)	10.73(±0.25)	5.50(±0.05)	1.89(±0.66)	0.63(±0.26)	1.56(±0.59)
INMO-LGCN	**12.36**(±0.38)	**13.62**(±0.08)	**14.52**(±0.11)	**5.75**(±0.08)	**12.17**(±0.05)	**6.13**(±0.02)	9.05(±0.05)	**7.99**(±0.27)	**7.94**(±0.07)
INMO-LGCN*	10.95(±0.28)	11.07(±0.10)	13.49(±0.05)	5.56(±0.08)	12.16(±0.07)	6.10(±0.02)	7.29(±0.16)	6.78(±0.07)	7.20(±0.03)

'\': Mult-VAE cannot handle the inductive scenario with new items and IDCF cannot apply to large datasets.

INMO-LGCN significantly outperform their basic versions on all datasets, whether adding new interactions or not. Especially after adding new interactions, INMO-LGCN-new increases *NDCG*@20 by 7.24%, 5.98%, 15.64% on Gowalla, Yelp, and Amazon-book, compared with LightGCN-new. The results empirically validate the inductive capability of our INMO by considering the new interactions in the embedding generation process.

5.3.2 New Users/Items. In this scenario, we randomly remove 20% of users and 20% of items from the training and validation data, but the test set keeps the same as the one used in the transductive scenario. This is a common scenario in real-world services, which means the methods need to recommend both new items and old items for new users who have not been seen during training.

Since NGCF and LightGCN need to learn the user-specific and item-specific embeddings, they can not work on the inductive recommendation scenario with new users and new items. To better demonstrate the effectiveness of INMO, we adapt LightGCN to this scenario by employing the same postprocessing strategy in IMC-GAE [33], denoted as IMC-LGCN. As for Mult-VAE, which is intrinsically unable to recommend new items without retraining, we evaluate its performance when only giving recommendations with old items. In addition, we introduce a non-personalized recommender **Popular**, which recommends the most popular items for all users, as the lower bound of performances.

Table 4 shows the performance comparison in the new users and new items scenario in terms of *NCDG*@20. In addition to the overall performances, we report the average *NCDG*@20 of new users and the retrieval results among new items. INMO-LGCN achieves the best recommendation results in these inductive methods, significantly outperforming all the baseline models, approaching the upper bound of INMO-LGCN-retrain. It indicates that our proposed INMO is quite effective in generalizing to new users and new items which are unseen during training.

Retraining Cost. INMO can avoid the frequent retraining of recommender models, which is of great value in real-world applications. Specifically, the full retraining of the LightGCN model with a Tesla V100 in a small dataset Gowalla still takes more than 2 hours, while our INMO is able to inductively recommend for new users and new items with an inference time of only several seconds. Figure 5 illustrates the time consumptions with and without INMO when

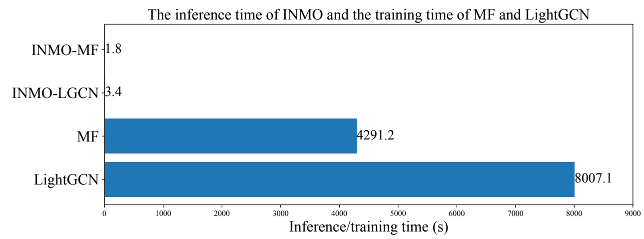

Figure 5: The retraining cost of two representative latent factor models.

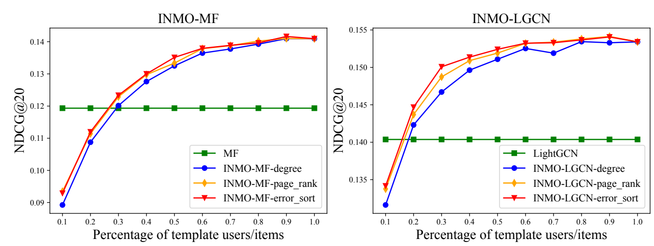

Figure 6: The recommendation performances under different percentages of template users and template items.

facing new users and new items. It is evident that our proposed inductive embedding module can save a lot of computing resources and provide timely and accurate recommendations.

5.4 Hyper-parameter Analysis (Q3)

In this section, we conduct experiments to analyze the impact of some hyper-parameters and training techniques.

5.4.1 The Number of Template Users and Template Items. We explore the influence of template users and template items on the recommendation performances, and compare various indicators to select the template users and template items as mentioned in Section 4.1. Experiments are conducted with both INMO-MF and INMO-LGCN on Gowalla dataset (shown in Figure 6). It is observed that both of INMO-MF and INMO-LGCN can yield better performances than their original versions, while with much fewer parameters. Specifically, INMO-MF outperforms MF with only 30% of model parameters, and INMO-LGCN beats the state-of-the-art LightGCN with only 20% of parameters. Figure 6 also empirically

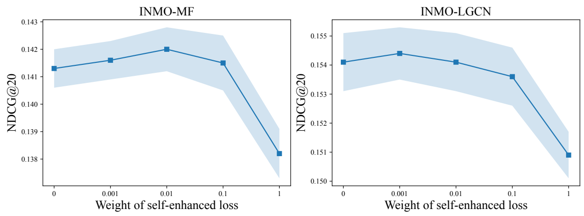

Figure 7: The recommendation performances under different weights β of the self-enhanced loss.

Table 5: The ablation study on two training techniques.

Method	NA	DI	New User	New Item	Over All
INMO-MF	✓	✓	**10.85**(±0.24)	**10.92**(±0.14)	**13.10**(±0.06)
	×	✓	3.29(±0.27)	4.06(±0.29)	5.31(±0.21)
	✓	×	10.45(±0.23)	10.70(±0.12)	12.83(±0.06)
	×	×	6.44(±0.41)	7.09(±0.38)	9.29(±0.26)
INMO-LGCN	✓	✓	12.36(±0.38)	**13.62**(±0.08)	**14.52**(±0.11)
	×	✓	12.26(±0.32)	13.42(±0.09)	14.37(±0.09)
	✓	×	**12.38**(±0.39)	13.44(±0.11)	**14.52**(±0.07)
	×	×	12.17(±0.35)	13.05(±0.15)	14.26(±0.11)

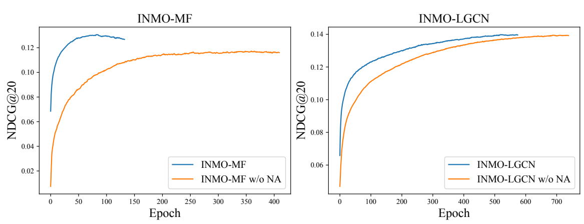

Figure 8: The training procedures with and without the normalization annealing technique.

validates that the error-sort indicator could guide to select a better set of template users and template items, leading to a higher recommendation accuracy. These findings demonstrate the effectiveness and scalability of our proposed INMO.

5.4.2 Self-enhanced loss. To demonstrate the impact of our proposed self-enhanced loss $\mathcal{L}_{SE}$, we evaluate the recommendation performances under different weights of the self-enhanced loss on the Gowalla dataset. As shown in Figure 7, we find that INMO-MF yields the best performance with a β at 0.01, while INMO-LGCN at 0.001, indicating that INMO-MF needs more additional supervised information. These results prove the general effectiveness of $\mathcal{L}_{SE}$ on various backbone models and suggest that the strength of the self-enhanced loss should be carefully tuned in different situations.

5.4.3 Ablation Study. We conduct experiments to evaluate the effectiveness of the training techniques proposed for INMO. We ablate the normalization annealing (NA) and drop interaction (DI) techniques in INMO-MF and INMO-LGCN, and then evaluate them in the inductive scenario with new users and new items on the Gowalla dataset. The comprehensive experimental results are presented in Table 5, which empirically verifies the effectiveness of both the two training techniques, especially for INMO-MF. It illustrates that, the optimization procedure for INMO-MF can be significantly improved with our well designed training techniques. It is necessary for INMO to adopt a dynamic normalization strategy and see varying combinations of interactions during training. Specifically, these techniques help INMO-MF to increase the $NDCG@20$ by 68.48% and 54.02% for new users and new items respectively. We realize that the hyper-parameter settings are significantly important for INMO-MF to achieve a competent inductive recommendation performance. In the case of INMO-MF without NA, it yields a passable transductive accuracy (shown in Figure 8), while performing poorly when facing new users and new items.

To further investigate how normalization annealing accelerates the model training, we delve into the training procedures of INMO-MF and IMNO-LGCN with and without this technique. The $NDCG@20$ on the validation set of Gowalla during training is illustrated in Figure 8. The lines end at different epochs as a result of the early stopping strategy. We can notice that the variants with normalization annealing converge much faster to the plateau and even achieve better performances, demonstrating the effectiveness of normalization annealing for model optimization.

6 CONCLUSION

In this work, we propose a novel **In**ductive Embedding **Mo**dule, namely INMO, to make recommendations in the inductive scenarios with *new interactions* and *new users/items* for collaborative filtering. INMO generates the inductive embeddings for users and items by considering their past interactions with some template users and template items. Remarkably, INMO is model-agnostic and scalable, which is applicable to existing latent factor models and has an adjustable number of parameters. To demonstrate the effectiveness and generality of our proposed INMO, we attach it to MF and LightGCN and obtain the inductive variants INMO-MF and INMO-LGCN. We evaluate INMO on three public real-world benchmarks across both transductive and inductive recommendation scenarios. Experimental results demonstrate that, INMO-MF* and INMO-LGCN* outperform their original versions MF and LightGCN with only 30% of parameters. Furthermore, INMO-LGCN yields the best performances in all the scenarios. We hope this work provides some new ideas for researchers to consider the inductive recommendation task, which is a common scenario in real-world services.

ACKNOWLEDGMENTS

This work is funded by the National Natural Science Foundation of China under Grant Nos. 62102402, U21B2046, and the National Key R&D Program of China (2020AAA0105200). Huawei Shen is also supported by Beijing Academy of Artificial Intelligence (BAAI).

REFERENCES

[1] Qi Cao, Huawei Shen, Jinhua Gao, Bingzheng Wei, and Xueqi Cheng. 2020. Popularity Prediction on Social Platforms with Coupled Graph Neural Networks. In *Proceedings of the 13th International Conference on Web Search and Data Mining (WSDM '20)*. 70–78.

[2] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '17)*. 335–344.

[3] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '11)*. 1082–1090.

[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. 191–198.

[5] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *Proceedings of the World Wide Web Conference (WWW '19)*. 417–426.

[6] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. 1992. Using Collaborative Filtering to Weave an Information Tapestry. *Commun. ACM* 35 (1992), 61–70.

[7] Jason S. Hartford, Devon R. Graham, Kevin Leyton-Brown, and Siamak Ravanbakhsh. 2018. Deep Models of Interactions Across Sets. In *Proceedings of the 35th International Conference on Machine Learning (ICML '18)*. 1914–1923.

[8] Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *Proceedings of the 25th International Conference on World Wide Web (WWW '16)*. 507–517.

[9] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. 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 (SIGIR '20)*. 639–648.

[10] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. 2018. Outer Product-based Neural Collaborative Filtering. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI '18)*. 2227–2233.

[11] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural Attentive Item Similarity Model for Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2018), 2354–2366.

[12] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. 173–182.

[13] Thomas Hofmann and Jan Puzicha. 1999. Latent Class Models for Collaborative Filtering. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI '99)*. 688–693.

[14] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative Metric Learning. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. 193–201.

[15] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

[16] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proceedings of the International Conference on Learning Representations (ICLR '14)*.

[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR '17)*.

[18] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '08)*. 426–434.

[19] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* (2009), 30–37.

[20] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*. ACM, 1073–1082.

[21] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. 689–698.

[22] Xixun Lin, Jia Wu, Chuan Zhou, Shirui Pan, Yanan Cao, and Bin Wang. 2021. Task-adaptive Neural Process for User Cold-Start Recommendation. In *Proceedings of The Web Conference 2021*. 1306–1316.

[23] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. 2008. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management (CIKM '08)*. 931–940.

[24] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *11th IEEE International Conference on Data Mining (ICDM '11)*. 497–506.

[25] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report.

[26] Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. 2017. entity2rec: Learning User-Item Relatedness from Knowledge Graphs for Top-N Item Recommendation. In *Proceedings of the 11th ACM Conference on Recommender Systems (RecSys '17)*. 32–36.

[27] Rahul Ragesh, Sundararajan Sellamanickam, Vijay Lingam, Arun Iyer, and Ramakrishna Bairi. 2021. User Embedding based Neighborhood Aggregation Method for Inductive Recommendation. *arXiv preprint arXiv:2102.07575* (2021).

[28] S. Rendle, C. Freudenthaler, Zeno Gantner, and L. Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI '09)*.

[29] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural Collaborative Filtering vs. Matrix Factorization Revisited. In *Proceedings of the 14th ACM Conference on Recommender Systems (RecSys '20)*. 240–248.

[30] Steffen Rendle and Lars Schmidt-Thieme. 2008. Online-Updating Regularized Kernel Matrix Factorization Models for Large-Scale Recommender Systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems (RecSys '08)*. 251–258.

[31] Rianne Rianne van den Berg, Thomas N Kipf, and Max Welling. 2018. Graph Convolutional Matrix Completion. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*.

[32] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*. 285–295.

[33] Wei Shen, Chuheng Zhang, Yun Tian, Liang Zeng, Xiaonan He, Wanchun Dou, and Xiaolong Xu. 2021. Inductive Matrix Completion Using Graph Autoencoder. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21)*. 1609–1618.

[34] Xiaoyuan Su and Taghi M. Khoshgoftaar. 2009. A Survey of Collaborative Filtering Techniques. *Advances in artificial intelligence* (Jan. 2009).

[35] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '15)*. 1235–1244.

[36] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*. 950–958.

[37] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19)*. 165–174.

[38] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled Graph Collaborative Filtering. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*. 1001–1010.

[39] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research)*. 6861–6871.

[40] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Junchi Yan, and Hongyuan Zha. 2021. Towards Open-World Recommendation: An Inductive Model-based Collaborative Filtering Approach. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research)*. 11329–11339.

[41] Bingbing Xu, Huawei Shen, Qi Cao, Keting Cen, and Xueqi Cheng. 2019. Graph Convolutional Networks Using Heat Kernel for Semi-Supervised Learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19)*. 1928–1934.

[42] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '18)*. 974–983.

[43] Tong Yu, Ole J. Mengshoel, Alvin Jude, Eugen Feller, Julien Forgeat, and Nimish Radia. 2016. Incremental learning for matrix factorization in recommender systems. In *2016 IEEE International Conference on Big Data (Big Data)*. 1056–1063.

[44] Muhan Zhang and Yixin Chen. 2020. Inductive Matrix Completion Based on Graph Neural Networks. In *Proceedings of the International Conference on Learning Representations (ICLR '20)*.

[45] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. 2020. How to Retrain Recommender System?: A Sequential Meta-Learning Method. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*. 1479–1488.

[46] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S Yu. 2018. Spectral Collaborative Filtering. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys '18)*. 311–319.