Graph-based Regularization on Embedding Layers for Recommendation

YUAN ZHANG, Peking University, China
FEI SUN, XIAOYONG YANG, CHEN XU, and WENWU OU, Alibaba Group, China
YAN ZHANG, Peking University, China

Neural networks have been extensively used in recommender systems. Embedding layers are not only necessary but also crucial for neural models in recommendation as a typical discrete task. In this article, we argue that the widely used ℓ_2 regularization for normal neural layers (e.g., fully connected layers) is not ideal for embedding layers from the perspective of regularization theory in Reproducing Kernel Hilbert Space. More specifically, the ℓ_2 regularization corresponds to the inner product and the distance in the Euclidean space where correlations between discrete objects (e.g., items) are not well captured. Inspired by this observation, we propose a graph-based regularization approach to serve as a counterpart of the ℓ_2 regularization for embedding layers. The proposed regularization incurs almost no extra computational overhead especially when being trained with mini-batches. We also discuss its relationships to other approaches (namely, data augmentation, graph convolution, and joint learning) theoretically. We conducted extensive experiments on five publicly available datasets from various domains with two state-of-the-art recommendation models. Results show that given a kNN (k-nearest neighbor) graph constructed directly from training data without external information, the proposed approach significantly outperforms the ℓ_2 regularization on all the datasets and achieves more notable improvements for long-tail users and items.

CCS Concepts: • **Information systems** → **Recommender systems**; • **Computing methodologies** → **Regularization**; *Learning latent representations*;

Additional Key Words and Phrases: Embedding, graph-based regularization, neural recommender system

ACM Reference format:
Yuan Zhang, Fei Sun, Xiaoyong Yang, Chen Xu, Wenwu Ou, and Yan Zhang. 2020. Graph-based Regularization on Embedding Layers for Recommendation. *ACM Trans. Inf. Syst.* 39, 1, Article 2 (September 2020), 27 pages.
https://doi.org/10.1145/3414067

1 INTRODUCTION

Followed by the great success in computer vision, neural/deep approaches have also been prevalent to deal with discrete objects (e.g., words and sentences in NLP, product items and user sequences

This work is supported by National Key Research and Development Program of China under Grant No. 2018AAA0101902, NSFC under Grant No. 61532001, and MOE-ChinaMobile Program under Grant No. MCM20170503.
Authors' addresses: Y. Zhang and Y. Zhang (corresponding author), Peking University, 5 Yiheyuan Road Haidian District, Beijing, China, 100871; emails: yuan.z@pku.edu.cn, zhy@cis.pku.edu.cn; F. Sun (corresponding author), X. Yang, C. Xu, and W. Ou, Alibaba Group, Jinhui Mansion B3F Qiyang Rd, Chaoyang District, Beijing 100102, China; emails: ofey.sunfei@gmail.com, {xiaoyong.yxy, chaos.xc, santong.oww}@alibaba-inc.com.

1046-8188/2020/09-ART2 $15.00
https://doi.org/10.1145/3414067

in recommendation) in both academia and industry. They not only have superior performances but also can be deployed easily in real-world production settings. Nonetheless, a crucial issue is that neural networks normally operate in Euclidean spaces instead of discrete domains (e.g., vocabulary of words and collection of items). The common treatment is to learn a discrete mapping (essentially, a lookup table) to map discrete objects into the Euclidean space and another one to map the output from neural networks back to the discrete space. These mappings are often dubbed as *embedding layers*, by which we can tell these "layers" are generally treated the same as other neural layers especially when it comes to regularization. However, are those regularization techniques invented for mappings between Euclidean spaces also suitable for embedding layers? Our answer is no.

In this article, we focus on a typical discrete domain problem, *item recommendation*, to study this research question. We first point out this issue with Reproducing Kernel Hilbert Space (RKHS) theory by showing that ℓ_2 penalties commonly used in regularizing neural networks are not ideal for regularizing embedding layers (Section 3.1). Specifically, training embedding layers with ℓ_2 regularization corresponds to using a kernel function that merely measures similarities between user behaviors by counting their co-clicked items (not being aware of *higher-order proximity*, widely leveraged in the context of collaborative filtering [7, 52, 63]). Also, the *stability* with regard to perturbations is bounded in terms of Euclidean distances, not being aware whether it is a relevant perturbation or not. These problems might, in turn, hurt generalization capacities and become more critical when we come across sparse datasets or the long tail phenomenon.

From our theoretical analysis, the main reason for this issue is that the correlations between items are not well captured in the ℓ_2 regularization. However, these correlations are crucial, since the main point of having embedding layers is to encode semantic correlations among discrete objects in latent spaces. Therefore, we propose to use a simple yet sound regularization approach based on graph Laplacians to overcome the above-mentioned limitations, motivated by the fact that graphs are generic tools to model relations between discrete objects (Section 3.2). With the proposed regularization, we encourage embedding layers to preserve the underlying structure of discrete spaces to get better stability properties, reduce overfitting and thus obtain better model performances. Moreover, as graphs are compact representation of complex relations, it suffices to have only a small number ($O(m)$, m is the set of items) of edges. As a result, the proposed regularization approach incurs no substantial computational overhead, especially when being trained with mini-batches.

Furthermore, we analyze the proposed regularization approach from different perspectives (Section 3.5). We show that training with the proposed graph-based regularization corresponds to *data augmentation* with data noising and smoothing by random-walk-like diffusion on graphs. We also illustrate its relationship to *graph convolutional networks (GCN)*, which have recently attracted extensive research efforts [59] and point out that the proposed regularization is essentially a more reasonable low-pass filter than graph convolutions. Finally, we discuss its connections to such approaches that jointly learn from collaborative signals (*user-item* interactions) and co-occurrence signals (*item-item* interactions).

We evaluated the proposed regularization with two state-of-the-art recommendation models (Section 4). Through extensive experiments on five datasets, we demonstrate that the effectiveness of the proposed regularization. Results also show that the proposed approach brings about more significant improvements regarding long-tail items and users that are inevitable and challenging in real-world recommender systems [18]. Besides, experiments suggest that the proposed regularization can help to obtain better generalization and produce higher quality embeddings. Finally, we investigated the effects of different forms of graphs and different settings of hyper-parameters.

To sum up, the main contribution of this article is threefold:

- We point out that the widely used ℓ_2 regularization is not ideal for embedding layers from the RKHS perspective. We provide specific implications of training recommendation models with ℓ_2 regularization.
- Motivated by our analysis, we propose a generic and efficient graph-based regularization approach and discuss its relationships to other methods, viz. data augmentation, graph convolution and joint learning.
- We conducted extensive experiments with two state-of-the-art neural recommendation models on five publicly available datasets. Results demonstrate the effectiveness of the proposed regularization and support our theoretical analysis as well.

2 BACKGROUND AND PRELIMINARIES

2.1 Problem Formulation and Notations

In this article, we focus on the problem of item recommendation with implicit feedback. That is, only implicit user feedback, such as clicks, watch history, and purchases, is available in contrast to explicit rating information. The goal is to learn a function $f(\cdot)$ that takes user historical behaviors as input and outputs relevance scores for candidate items to each user. The domain of f ($dom(f)$) is a discrete space of all possible combinations or permutations of items depending on whether the ordering of users' sequential behaviors is considered. The range of f ($range(f)$) is usually a discrete probability space (though, sometimes un-normalized) over items. Generally speaking, the learning process is essentially data fitting. In the former case, the training data can be represented as a user behavior matrix $\mathbf{X} \in \mathbb{M}^{N \times m}$ and a ground truth matrix of users' interested items $\mathbf{Y} \in \mathbb{M}^{N \times m}$, where m and N are the number of items and users, respectively, and each row of the matrices (namely, $\mathbf{x}_i$ or $\mathbf{y}_i$) is a multi-hot encoding. In the latter, X becomes an $N \times L \times m$ tensor (L is the maximum length of user behavior sequences), where each sequence i is represented by an $L \times m$ matrix $\mathbf{X}_i$, the lth row of which is a one-hot encoding for the lth behavior.

As mentioned in Section 1, if we want to make use of neural networks to build a function f, then there must exist an input embedding layer that maps $dom(f)$ to Euclidean space and another output embedding layer mapping neural outputs in the Euclidean space back to $range(f)$. Thus, the learnable parameters in f can be grouped into three categories $\{E, W, O\}$, where E and O are parameters within the input and output embedding layers, respectively, and W are other parameters within the neural networks.

The thesis of this article is that it is not ideal to treat these three categories of parameters equally as people usually do: Even ℓ_2 regularization, one of the most commonly used regularization techniques for neural nets W, is shown to be not a very suitable choice for embedding layers E and O; instead, the proposed graph Laplacian regularization is a more proper counterpart.

2.2 Recommendation Models

We review two state-of-the-art recommendation models that are studied in this work (Section 4).

2.2.1 Mult-DAE. Using denoising auto-encoders (DAE) for recommendation has been discussed in several prior works (e.g., References [19, 27, 58]). We choose its latest variant Mult-DAE proposed by Liang et al. [27] due to its conciseness and effectiveness. Mult-DAE consists of a neural encoding function $\mathbf{z_i} = g_\phi(\mathbf{x}_i)$ that encodes the multi-hot encoding of a user's historical behaviors $\mathbf{x}_i$ into a dense vector $\mathbf{z_i}$ and a neural decoding function $\hat{\mathbf{x}}_\mathbf{i} = h_\theta(\mathbf{z_i})$ that outputs predicted item relevance. Note that the first layer of g_ϕ is the input embedding layer parameterized by E and the last layer of h_θ is the output embedding layer parameterized by O. Then, they use softmax

to obtain a normalized relevance distribution over items, i.e., $\tilde{\mathbf{x}}_\mathbf{i} = \text{softmax}(\hat{\mathbf{x}}_\mathbf{i})$. In Mult-DAE, the ground-truth matrix $\mathbf{Y}$ is the same as the input matrix X. To avoid trivial solutions, they randomly dropout input user behaviors during training.

2.2.2 SASRec. Compared with Mult-DAE, SASRec [20] exploits sequential information of user behaviors. It uses Transformer [45], a seq2seq model recently proposed in NLP, to model user sequences. In SASRec, the input and the output embedding layers (as an analogy of word embedding layers) share the same set of parameters, that is, $E = O$. Since SASRec always predicts the next user behavior, each $\mathbf{X}_\mathbf{i}$ represents the user behaviors before certain time step t (cropping or padding to length L) and the corresponding $\mathbf{y}_i$ is the one-hot encoding of the user behavior at time step t.

2.3 RKHS and Tikhonov Regularization

We briefly review vector-valued RKHS theory and refer the reader to a comprehensive yet approachable survey by Alvarez et al. [1]. Suppose we are interested in learning functional relationships between an input space $\mathcal{X}$ and an output vector space $\mathbb{R}^D$.[1] Given a symmetric kernel function $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}^{D\times D}$, we can induce a Hilbert space of vector-valued functions,

$$\mathcal{H} = \left\{ f \,\middle|\, f(\mathbf{x}) = \sum_{i=1}^{p} K(\overline{\mathbf{x}}_\mathbf{i}, \mathbf{x})\mathbf{c}_\mathbf{i},\ p \in \mathbb{N}^+,\ \overline{\mathbf{x}}_\mathbf{i} \in \mathcal{X},\ \mathbf{c}_\mathbf{i} \in \mathbb{R}^D \right\},$$

with the inner product and the norm

$$\langle f, g\rangle_\mathcal{H} = \sum_{i=1}^{p}\sum_{j=1}^{p'} \mathbf{c}_\mathbf{i}{}^\top K(\overline{\mathbf{x}}_\mathbf{i}, \overline{\mathbf{x}}'_\mathbf{j})\mathbf{c}'_\mathbf{j}, \tag{1}$$

$$\|f\|_\mathcal{H} = \sqrt{\langle f, f\rangle_\mathcal{H}} = \sqrt{\sum_{i=1}^{p}\sum_{j=1}^{p'} \mathbf{c}_\mathbf{i}{}^\top K(\overline{\mathbf{x}}_\mathbf{i}, \overline{\mathbf{x}}_\mathbf{j})\mathbf{c}_j}, \tag{2}$$

where $f(\mathbf{x}) = \sum_{i=1}^{p} K(\overline{\mathbf{x}}_\mathbf{i}, \mathbf{x})\mathbf{c}_\mathbf{i}$ and $g(\mathbf{x}) = \sum_{i=1}^{p'} K(\overline{\mathbf{x}}'_\mathbf{i}, \mathbf{x})\mathbf{c}'_\mathbf{i}$.[2] The induced space is called a reproducing kernel Hilbert space (RKHS) induced by K, because it has the following reproducing properties:

$$\langle f, K(\cdot, \mathbf{x})\mathbf{c}\rangle_\mathcal{H} = \langle f(\mathbf{x}), \mathbf{c}\rangle. \tag{3}$$

The idea of Tikhonov regularization is that we want to select a function from the RKHS that not only minimizes an empirical loss but also has a lower complexity measured by its norm. Formallly,

$$\arg\min_{f\in\mathcal{H}} \frac{1}{N}\sum_{i}^{N} V(f(\mathbf{x}_i), \mathbf{y}_i) + \lambda\|f\|^2_\mathcal{H}, \tag{4}$$

where $\{(\mathbf{x}_i, \mathbf{y}_i)\}$ are N training samples and $V(f(\mathbf{x}), \mathbf{y})$ is the loss function to measure the deviation of prediction $f(\mathbf{x})$ from the ground-truth $\mathbf{y}$, for example, ℓ_2 loss $\|f(\mathbf{x}) - \mathbf{y}\|_2^2$.

The solution to Equation (4) can be represented by the expression

$$f(x) = \sum_{i=1}^{N} K(\mathbf{x}_i, \mathbf{x})\mathbf{c}_\mathbf{i}, \tag{5}$$

[1]The reader can imagine $\mathcal{X}$ and $\mathbb{R}^D$ as the spaces of user behavior sequences and relevance prediction over D candidate items, respectively. Later, we will assume them both to be m-dimensional vector spaces for the sake of simplicity.
[2]Note that $\{\overline{\mathbf{x}}_\mathbf{i}\}$ can be arbitrary points in $\mathcal{X}$ so that we deliberately use a different notation from that for data points $\{\mathbf{x}_i\}$ to avoid confusion.

which is the well-known *representer theorem* proved by Kimeldorf and Wahba [21]. Specifically, if V is the ℓ_2 loss, the coefficient satisfies the following linear system:

$$\overline{\mathbf{c}} = (K(\mathbf{X}, \mathbf{X}) + \lambda N \mathbf{I})^{-1} \overline{\mathbf{y}}, \tag{6}$$

where $\overline{\mathbf{c}}$ and $\overline{\mathbf{y}}$ are ND vectors obtained by concatenating the coefficients and ground-truth labels, respectively, and $K(\mathbf{X}, \mathbf{X})$ is an $ND \times ND$ block matrix with entries $[K(\mathbf{x}_i, \mathbf{x}_j)]_{d,d'}$.

3 METHODOLOGY

3.1 Motivation: Issues with ℓ_2 Regularization

To motivate the proposed graph Laplacian regularization, we first recover the commonly used ℓ_2 regularization under the Tikhonov regularization framework with a simplified linear model.

Proposition 3.1. *Suppose the input and the output space are both m-dimensional vector spaces, i.e., $\mathbb{R}^m$, given kernel function $K(\overline{\mathbf{x}}_\mathbf{i}, \overline{\mathbf{x}}_\mathbf{j}) = k_E(\overline{\mathbf{x}}_\mathbf{i}, \overline{\mathbf{x}}_\mathbf{j}) \cdot \mathbf{K_O}$, where $k_E(\overline{\mathbf{x}}_\mathbf{i}, \overline{\mathbf{x}}_\mathbf{j}) = \overline{\mathbf{x}}_\mathbf{i}{}^\top \mathbf{K_E} \overline{\mathbf{x}}_\mathbf{j}$ and $\mathbf{K_E}, \mathbf{K_O}$ are nonsingular matrices,*

(1) the RKHS induced by $K(\cdot, \cdot)$ is

$$\mathcal{H} = \left\{ f \,\middle|\, f(\mathbf{x}) = \sum_{i=1}^{p} K(\overline{\mathbf{x}}_\mathbf{i}, \mathbf{x}) \mathbf{c_i} = \Theta \mathbf{x}, \Theta \in \mathbb{R}^{m \times m} \right\}, \tag{7}$$

where $\Theta = \mathbf{K_O} \mathbf{C} \overline{\mathbf{X}} \mathbf{K_E}$ ($\mathbf{C} = [\mathbf{c_1}, \ldots, \mathbf{c_p}]$ and $\overline{\mathbf{X}} = [\overline{\mathbf{x}}_\mathbf{1}, \ldots, \overline{\mathbf{x}}_\mathbf{p}]^\top$);

(2) the induced norm is $\|f\|_\mathcal{H} = \sqrt{\mathrm{tr}(\mathbf{K}_E^{-1} \Theta^\top \mathbf{K}_O^{-1} \Theta)}$.

The recommendation problem with m items can be simplified as selecting a function from the space of linear transformations, namely, $\mathcal{H}$, with the loss function given in Equation (4). Thanks to the representer theorem, Θ can be represented by $\mathbf{K_O} \mathbf{C} \mathbf{X} \mathbf{K_E}$ (as a remainder, $\mathbf{X}$ is the input data/user behavior matrix). Because user behaviors usually have strong clustering properties, $\mathbf{X}$ is by far a full-rank matrix and hence all possible Θ are also low-rank matrices as well (i.e., $\mathrm{rank}(\Theta) \leq \mathrm{rank}(\mathbf{X}) \triangleq d \ll m$). Thus, we can decompose the linear model in Equation (7) by two low-rank matrices Θ_1 and $\Theta_2 \in \mathbb{R}^{m \times d}$ to obtain the most concise recommendation model for illustration, i.e.,

$$f(\mathbf{x}) = \Theta \mathbf{x} = \Theta_2 \Theta_1^\top \mathbf{x}, \tag{8}$$

where $\mathbf{x}$ is the multi-hot encoding of clicked items as mentioned in Section 2.1. In this model, Θ_1 and Θ_2 can be regarded as input and output embedding layers, of which each row corresponds to an item, respectively[3]; the neural layers degenerate to an identity transformation with no learnable parameters. In short, $E = \{\Theta_1\}$, $O = \{\Theta_2\}$ and $W = \emptyset$.

Then, we can obtain the Tikhonov regularization for this model,

$$\arg\min_{\Theta_1, \Theta_2} \frac{1}{N} \sum_{i=1}^{N} V(f(\mathbf{x}_i), \mathbf{y}_i) + \lambda \cdot \mathrm{tr}\Big(\Big(\Theta_1^\top \mathbf{K}_E^{-1} \Theta_1 \Big) \cdot \Big(\Theta_2^\top \mathbf{K}_O^{-1} \Theta_2 \Big) \Big). \tag{9}$$

However, Equation (9) is not convex w.r.t. (Θ_1, Θ_2), because the trace of a product of two matrices (i.e., $\mathrm{tr}(\mathbf{AB})$) is not convex. Thus, we can resort to optimize the convex envelope (the tightest convex lower bound) of this non-convex function with the following proposition.

[3]To see this, the ith row $\Theta_1(i, :)$ and $\Theta_2(i, :)$ denote input and output embeddings for item i, respectively. This simplified model first retrieves input embeddings for all clicked items (i.e., $\{\Theta_1(i, :) | \forall i, x_i = 1\}$), aggregates them by sum pooling and then projects to the output embedding $\Theta_2(i, :)$ to get the relevance score for each item i.

Proposition 3.2. *The convex envelope of* $f((\mathbf{A},\mathbf{B})) = \mathrm{tr}(\mathbf{AB})$ *over the set* $\mathcal{M} = \{(\mathbf{A},\mathbf{B}) \in \mathbb{S}_n \times \mathbb{S}_n | \mathbf{O} \preceq \mathbf{A},\mathbf{B} \preceq \mathbf{I}, \sigma_n(\mathbf{A}) + \sigma_n(\mathbf{B}) \geq 1\}$ *($\sigma_n(\mathbf{A})$ and $\sigma_n(\mathbf{B})$ are the smallest eigenvalues of* $\mathbf{A}$ *and* $\mathbf{B}$*, respectively) is* $\mathrm{tr}(\mathbf{A}) + \mathrm{tr}(\mathbf{B}) - n$.

This proposition is proved with the fact that the bi-conjugate function f^{**} is the (closed) convex envelope of f (see Appendix A.2). In addition, the trace of a product of two matrices are bounded by the sum of traces of these two matrices, i.e., $\mathrm{tr}(\mathbf{A}) + \mathrm{tr}(\mathbf{B}) - n \leq \mathrm{tr}(\mathbf{AB}) \leq \mathrm{tr}(\mathbf{A}) + \mathrm{tr}(\mathbf{B})$ when $\mathbf{O} \preceq \mathbf{A},\mathbf{B} \preceq \mathbf{I}$. Thus, Equation (9) now can be reasonably approximated by

$$\arg\min_{\Theta_1,\Theta_2} \frac{1}{N}\sum_{i=1}^{N} V(f(\mathbf{x}_i),\mathbf{y}_i) + \lambda\cdot\Big(\mathrm{tr}\Big(\Theta_1^\top \mathbf{K}_E^{-1}\Theta_1\Big) + \mathrm{tr}\Big(\Theta_2^\top \mathbf{K}_O^{-1}\Theta_2\Big)\Big). \tag{10}$$

If $\mathbf{K_E} = \mathbf{K_O} = \mathbf{I}$, then we recover the ℓ_2 regularization,

$$\begin{aligned}&\arg\min_{\Theta_1,\Theta_2} \frac{1}{N}\sum_{i=1}^{N} V(f(\mathbf{x}_i),\mathbf{y}_i) + \lambda\cdot\Big(\mathrm{tr}\Big(\Theta_1^\top\Theta_1\Big) + \mathrm{tr}\Big(\Theta_2^\top\Theta_2\Big)\Big)\\ &= \arg\min_{\Theta_1,\Theta_2} \frac{1}{N}\sum_{i=1}^{N} V(f(\mathbf{x}_i),\mathbf{y}_i) + \lambda\cdot\Big(\|\Theta_1\|_F^2 + \|\Theta_2\|_F^2\Big).\end{aligned} \tag{11}$$

After formulating ℓ_2 regularization as an approximation for Tikholov regularization in RKHS, we point out issues in using this type of regularization to motivate our work.

According to Equations (5) and (6), the solution to Equation (9) (assuming that V is the ℓ_2 loss for simplicity and N is relatively large) is

$$f(\mathbf{x}) \approx \sum_{i=1}^{N} \frac{1}{\lambda N} k_E(\mathbf{x}_i,\mathbf{x})\mathbf{K}_O\mathbf{y}_i. \tag{12}$$

The prediction $f(\mathbf{x})$ at a test point $\mathbf{x}$ is (approximately) a weighted average of transformed ground-truth labels $\mathbf{K}_O\mathbf{y}_i$ and the weights are similarities $k_E(\mathbf{x}_i,\mathbf{x})$ between $\mathbf{x}$ and user behaviors $\mathbf{x}_i$ in the training set. However, one issue is that, in the case of ℓ_2 regularization where $\mathbf{K_E} = \mathbf{K_O} = \mathbf{I}$, the similarity (i.e., $k_E(\mathbf{x}_i,\mathbf{x}) = \mathbf{x}_i^\top\mathbf{x}$) is only measured by the count of their shared items. Therefore, *higher-order proximity* cannot be aware of in this model, for example, if we add to $\mathbf{x}$ some items that are neither in $\mathbf{x}$ nor $\mathbf{x}_i$ but often co-clicked with (hence, relevant to) those items in $\mathbf{x}_i$ and let $\mathbf{x}'$ denote the result, $\mathbf{x}'$ should have been considered as more relevant to $\mathbf{x}_i$ than $\mathbf{x}$ while the similarity measure remains unchanged, i.e., $k_E(\mathbf{x}_i,\mathbf{x}) = k_E(\mathbf{x}_i,\mathbf{x}')$. Also, the multi-hot encoded ground-truth labels $\{\mathbf{y}_i\}$ that are being averaged are sparse and "non-smooth" in the sense that two similar items can be regarded as relevant (1) and irrelevant (0) at the same time, respectively. This is more of an issue if the data point $\mathbf{x}$ we are evaluating at only has a small number of items, meanwhile, most of which are long-tail items. In that case, only a *small number* of *sparse* ground-truth labels are averaged out in Equation (12) leading to a high prediction variance (see empirical evidence in Section 4).

Furthermore, we also investigate the stability bound of the model.

Proposition 3.3. *For any function f from the RKHS in Proposition 3.1 and any vector $\mathbf{x}, \Delta\mathbf{x} \in \mathbb{R}^m$, the stability of f at the point $\mathbf{x}$ with regard to perturbation $\Delta\mathbf{x}$ is bounded as follows* [4]*:*

$$\|f(\mathbf{x}+\Delta\mathbf{x}) - f(\mathbf{x})\|_{\mathbf{K}_O^{-1}} \leq \|f\|_{\mathcal{H}} \cdot \|\Delta\mathbf{x}\|_{\mathbf{K_E}}. \tag{13}$$

[4]We denote $\|\mathbf{v}\|_{\mathbf{M}} = \langle\mathbf{v},\mathbf{Mv}\rangle$ for any vector $\mathbf{v}$ and matrix $\mathbf{M}$.

We can see that, if $\mathbf{K_O} = \mathbf{K_E} = \mathbf{I}$ as in the ℓ_2 regularization, the perturbation and the deviation of $f(\mathbf{x})$ against it in the stability bound are both measured by Euclidean distances, i.e., $\|f(\mathbf{x}+\Delta\mathbf{x}) - f(\mathbf{x})\|_2$ and $\|\Delta\mathbf{x}\|_2$. However, the Euclidean distance cannot properly capture semantic differences within the discrete item spaces. For instance, suppose we replace an item in $\mathbf{x}$ with a relevant item and an irrelevant one, and let $\Delta_{\mathbf{rel}}$ and $\Delta_{\mathbf{irrel}}$ denote the changes, respectively. The magnitude of these two perturbations are deemed as the same and $f(\mathbf{x}+\Delta_{\mathbf{rel}})$ is not bounded to be closer to $f(\mathbf{x})$ than $f(\mathbf{x}+\Delta_{\mathbf{irrel}})$, simply because $\|\Delta_{\mathbf{rel}}\|_2 = \|\Delta_{\mathbf{irrel}}\|_2 = \sqrt{1^2+1^2} = \sqrt{2}$.

To sum up, ℓ_2 regularization implicitly assumes that the regularized transformations (e.g., Θ_1, Θ_2) should lie in the Euclidean space. However, the correlation among dimensions are not captured in the Euclidean space, while the items are highly interacted with each other, which is the underlying mechanism that enables recommender systems to work. This can be made more clear from the Bayesian perspective [55]: If V is the ℓ_2 loss, then the solution to the regularization problem is exactly the posterior mean of the Gaussian Process (GP), which assumes that

$$f(\mathbf{X}) \sim \mathcal{N}(\mathbf{0}, K(\mathbf{X},\mathbf{X})), \tag{14}$$

where $K(\mathbf{X},\mathbf{X}) = (\mathbf{X}\mathbf{K_E}\mathbf{X}^\top) \otimes \mathbf{K_O}$ ($\otimes$ is the Kronecker product) in our case, where $\mathbf{X}\mathbf{K_E}\mathbf{X}^\top$ and $\mathbf{K_O}$ denote correlations between rows (i.e., users) and columns (i.e., items), respectively.

3.2 Graph-based Regularization

Motivated by the analysis above, we propose to use graphs as a generic tool to capture correlations between the discrete objects. More specifically, given a graph $\mathcal{G}$, which encodes the relevance relationships among items, we choose the inverse of the regularized graph Laplacian [40] as input and output kernel matrices, that is,

$$\mathbf{K_E} = (\mathbf{I} + \gamma_E \cdot \mathbf{L}_\mathcal{G})^{-1}, \tag{15}$$

$$\mathbf{K_O} = (\mathbf{I} + \gamma_O \cdot \mathbf{L}_\mathcal{G})^{-1}, \tag{16}$$

where $\gamma_E \geq 0$ and $\gamma_O \geq 0$ are tunable parameters and $\mathbf{L}_\mathcal{G}$ is the graph Laplacian given by $\mathbf{L}_\mathcal{G} = \mathbf{D} - \mathbf{A}$ (used in the rest of this article) or its normalized form $\mathbf{L}_\mathcal{G} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ with diagonal degree matrix $\mathbf{D}$ and adjacency matrix $\mathbf{A}$ of graph $\mathcal{G}$. Compared with the graph Laplacian matrix, which encodes local interactions between only a few pairs of items, its inverse captures global correlations between each pair, analogous to precision matrices versus covariance matrices.

By plugging Equations (15) and (16) into Equation (10), we obtain the proposed regularization term for embedding layers,

$$\begin{aligned}
\mathcal{L}_{reg} &= \mathrm{tr}\left(\Theta_1^\top(\mathbf{I} + \gamma_E \cdot \mathbf{L}_\mathcal{G})\Theta_1\right) + \mathrm{tr}\left(\Theta_2^\top(\mathbf{I} + \gamma_O \cdot \mathbf{L}_\mathcal{G})\Theta_2\right) \\
&= \mathrm{tr}\left(\Theta_1^\top\Theta_1\right) + \mathrm{tr}\left(\Theta_2^\top\Theta_2\right) + \gamma_E \cdot \mathrm{tr}\left(\Theta_1^\top\mathbf{L}_\mathcal{G}\Theta_1\right) + \gamma_O \cdot \mathrm{tr}\left(\Theta_2^\top\mathbf{L}_\mathcal{G}\Theta_2\right) \\
&= \|\Theta_1\|_F^2 + \|\Theta_2\|_F^2 + \gamma_E \cdot \sum_{(i,j)\in\mathcal{G}} \|\Theta_1(i,:) - \Theta_1(j,:)\|_2^2 + \gamma_O \cdot \sum_{(i,j)\in\mathcal{G}} \|\Theta_2(i,:) - \Theta_2(j,:)\|_2^2,
\end{aligned} \tag{17}$$

where $\Theta_1(i,:)$ denotes the ith row of input embedding matrix Θ_1, i.e., the embedding vector of the ith item as an input, and so forth. If $\gamma_E = \gamma_O = 0$ or $\mathcal{G}$ is an empty graph, then $\mathbf{K_E} = \mathbf{K_O} = \mathbf{I}$ and $\mathcal{L}_{reg}$ degenerates to ℓ_2 regularization. Otherwise, we can check that the issues mentioned in Section 3.1 get resolved with the proposed regularization. Notably, the LHS of the stability bound in Equation (13) can be decomposed into two parts now,

$$\begin{aligned}
\left\|f(\mathbf{x}+\Delta\mathbf{x}) - f(\mathbf{x})\right\|_{\mathbf{K_O}^{-1}} &= f(\mathbf{x}+\Delta\mathbf{x}) - f(\mathbf{x})\|_2 \\
&+ \gamma_O \cdot \sum_{(i,j)\in\mathcal{G}} \left([f(\mathbf{x}+\Delta\mathbf{x}) - f(\mathbf{x})]_i - [f(\mathbf{x}+\Delta\mathbf{x}) - f(\mathbf{x})]_j\right)^2,
\end{aligned} \tag{18}$$

where the first term measures *deviations* of $f(\mathbf{x} + \Delta\mathbf{x})$ from $f(\mathbf{x})$, and the second term quantifies the *consistency* of deviations among correlated items. Furthermore, the consistency among predictions for items made by f is also bounded by the norm of f and user histories, $\|f(\mathbf{x})\|_{\mathbf{K_O}^{-1}} = \sum_{(i,j)\in\mathcal{G}}(f(\mathbf{x})_i - f(\mathbf{x})_j)^2 \leq \|f\|_{\mathcal{H}} \cdot \|\mathbf{x}\|_{\mathbf{K_E}}$.

Note that the regularization term given in Equation (17) is applicable to embedding layers for *arbitrary* neural recommendation models although it is derived from a simplified model for more insight. In Section 4, we will evaluate the proposed regularization technique with two state-of-the-art recommendation models.

3.3 Constructing Graphs from Data

There are various ways to obtain graph $\mathcal{G}$, e.g., References [2, 10]. Since this line of research is mostly orthogonal to our contribution, here we only discuss two guidelines for graph construction.

- **Accurate Edges.** The point of the graph here is to capture correlations between items. Thus, the more faithfully the edges can reflect correlated item pairs, the better performance the graph-based regularization can achieve. That being said, it is not necessary nor practical to pursue a perfect graph to obtain a significant improvement over the ℓ_2 regularization. After all, we are replacing an empty graph as assumed in the ℓ_2 regularization with a graph that is capable of providing some signals about item correlations.
- **Balanced Distribution.** It is important to avoid having an overly unbalanced degree distribution. For instance, as might not be easily noticed, it is not a good choice to use the number of co-occurrences in user behavior sequences (i.e., $|\mathcal{U}_i \cap \mathcal{U}_j|$, where $\mathcal{U}_i$ denotes the set of users who have clicked item i and so forth) as the metric to select edges connecting relevant items. It would lead to a highly biased degree distribution towards head/popular items.

Following the above guidelines, we use the k-nearest neighbor (kNN) graph based on the Jaccard similarities between the sets of users who have clicked the items, i.e., $|\mathcal{U}_i \cap \mathcal{U}_j|/|\mathcal{U}_i \cup \mathcal{U}_j|$. This is just a simple heuristic-based graph construction procedure, so it is not necessarily optimal and the results in our experiments only provide a lower bound for the performance of our proposed approach. The reader can find more empirical discussions in Section 4.3.4.

3.4 Training and Time Complexity

The proposed approach differs from ℓ_2 regularization in the last two additional terms as in Equation (17). The form of these two terms has computational advantages. First, the summation is taken over the set of edges, while graphs have the ability to represent complex relations with very sparse structures. Empirical studies suggest that a constant (say, three to five) times of the number of nodes/items (i.e., $O(m)$, m is the number of items) suffices as the number of edges to bring about substantially improvements. Thus, these two added terms are no more time-consuming to calculate than the first two terms as also appear in ℓ_2 regularization. Formally, given that the number of edges is $|\mathcal{G}| = O(m)$ in the graph used for regularization, the time complexity is $O((m + |\mathcal{G}|)d) = O(md)$ where d is the embedding dimension.

More importantly, the computation of the two added terms is highly parallelizable. During each training iteration, we only need to sample a batch of edges to approximate the summations. This makes the proposed regularization more efficient so that it even takes almost no extra wall-clock time (not only asymptotically) than ℓ_2.

3.5 Relationship to Other Approaches

3.5.1 Graph-based Regularization as Data Noising and Smoothing. Bishop [5] proves the correspondence between Tikholov regularization and training with noises if a sum-of-squares error

function is used. We now reach to an analogous result and investigate the specific form of noises that the proposed graph-based regularization is connected to. We start by plugging Equations (15) and (16) into Equation (5),

$$\begin{aligned} f(\mathbf{x}) &\approx \frac{1}{\lambda N}\sum_{i=1}^{N}\langle(\mathbf{I}+\gamma_E\cdot\mathbf{L}_{\mathcal{G}})^{-1}\mathbf{x}_i,\mathbf{x}\rangle\cdot(\mathbf{I}+\gamma_O\cdot\mathbf{L}_{\mathcal{G}})^{-1}\mathbf{y}_i \\ &= \frac{1}{\lambda N}\sum_{i=1}^{N}\mathbb{E}_{\tilde{\mathbf{x}}_i\sim P(\cdot|\mathbf{x}_i)}\mathbb{E}_{\tilde{\mathbf{y}}_i\sim P'(\cdot|\mathbf{y}_i)}\langle\tilde{\mathbf{x}}_\mathbf{i},\mathbf{x}\rangle\cdot\tilde{\mathbf{y}}_\mathbf{i}, \end{aligned} \tag{19}$$

where $P(\cdot|\mathbf{x}_i)$ (the same with $P'(\cdot|\mathbf{y}_i)$) is the probability distribution of random vector $\tilde{\mathbf{x}}_i$ obtained by independently replacing each jth item of $\mathbf{x}_i$, if $x_{ij}=1$, randomly with another item according to the jth column of $(\mathbf{I}+\gamma_E\cdot\mathbf{L}_{\mathcal{G}})^{-1}$, which is a probability mass distribution thanks to the following proposition.

Proposition 3.4. *If $\mathbf{L}_{\mathcal{G}}$ is the unnormalized graph Laplacian, then $(\mathbf{I}+\gamma_E\cdot\mathbf{L}_{\mathcal{G}})^{-1}$ and $(\mathbf{I}+\gamma_I\cdot\mathbf{L}_{\mathcal{G}})^{-1}$ are stochastic matrices (i.e., component-wise non-negative and each row/column sums to one).*

Therefore, suppose that we use ℓ_2 loss, training with the proposed regularization term in Equation (17),

$$\arg\min_{\Theta_1,\Theta_2}\frac{1}{N}\sum_{i=1}^{N}\|f(\mathbf{x}_i)-\mathbf{y}_i\|_2^2+\lambda\cdot\mathcal{L}_{reg}, \tag{20}$$

leads to the same solution as optimizing the ℓ_2-regularized loss with noises/perturbations,

$$\arg\min_{\Theta_1,\Theta_2}\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{\substack{\tilde{\mathbf{x}}_i\sim P(\cdot|\mathbf{x}_i)\\ \tilde{\mathbf{y}}_i\sim P'(\cdot|\mathbf{y}_i)}}\|f(\tilde{\mathbf{x}}_i)-\tilde{\mathbf{y}}_i\|_2^2+\lambda\cdot\left(\|\Theta_1\|_F^2+\|\Theta_2\|_F^2\right). \tag{21}$$

We dive deeper into the noising mechanism. Suppose graph $\mathcal{G}$ is approximately k-regular (i.e., the degree of each node is k) for mathematical simplicity,

$$\begin{aligned} (\mathbf{I}+\gamma\cdot\mathbf{L}_{\mathcal{G}})^{-1} &\approx \frac{1}{1+\gamma k}\left(\mathbf{I}-\frac{\gamma}{1+\gamma k}\mathbf{A}\right)^{-1} \\ &= \frac{1}{1+\gamma k}\mathbf{I}+\sum_{t=1}^{\infty}\frac{(\gamma k)^t}{(1+\gamma k)^{t+1}}\left(\frac{1}{k}\mathbf{A}\right)^t, \end{aligned} \tag{22}$$

where the second line is due to the fact $-\mathbf{I}\prec\gamma/(1+\gamma k)\mathbf{A}\prec\mathbf{I}$ guaranteed by Gershgorin's circle theorem, because the off-diagonal entries of each row in $\gamma/(1+\gamma k)\mathbf{A}$ sums to strictly less than one. Equation (22) tells that one item stays unchanged with probability $\gamma/(1+\gamma k)$ or, otherwise, is replaced by another item (though, not necessarily a different item) sampled from a random walk process where the walker stops at step t with probability $(\gamma k)^t/(1+\gamma k)^{t+1}$.

Interestingly, our work can also be viewed from a similar perspective to Xie et al. [60] and Kong et al. [23] in NLP where the connections between data noising in neural language models and smoothing in n-gram models are derived. In other words, training with the proposed regularization is equivalent to propagating multi-hot encodings (i.e., $\{\mathbf{x_i}\}$ and $\{\mathbf{y_i}\}$) on graphs to obtain smoother inputs and labels (i.e., $(\{\mathbf{I}+\gamma_E\cdot\mathbf{L}_{\mathcal{G}})^{-1}\mathbf{x}_i\}$ and $\{(\mathbf{I}+\gamma_O\cdot\mathbf{L}_{\mathcal{G}})^{-1}\mathbf{y}_i\}$) used for training. Different from n-gram-based noising, which corresponds to replacing a word with another one sampled from a *unified* n-gram distribution *for any word*, our approach smooths data by replacing an item with its *relevant* substitutes via random walks *started at that item* (analogous to the *synonym replacement* technique [54, 69] in NLP).

3.5.2 Graph-based Regularization as Low-pass Filter. Another tempting approach for our problem is to stack GCN (Graph Convolution Network) layers on embeddings layers to propagate embeddings over graphs. Formally,

$$f(\mathbf{x}) = \phi_2(\Phi_2) \cdot \phi_1(\Phi_1)^\top \cdot \mathbf{x}, \tag{23}$$

where ϕ_1 and ϕ_2 are, say, K-layer GCNs on input and output embeddings layers, respectively. To compare this approach with the proposed, we follow the idea of Wu et al. [56] to consider a simplified version of GCN by removing nonlinearity in it (Wu et al. also show that this simplied version can obtain comparable, occasionally even better, performance), which results in

$$f(\mathbf{x}) = \mathbf{S}^K\Phi_2 \cdot (\mathbf{S}^K\Phi_1)^\top \cdot \mathbf{x} = \tilde{\mathbf{S}}\Phi_2 \cdot (\tilde{\mathbf{S}}\Phi_1)^\top \cdot \mathbf{x}. \tag{24}$$

From this form, as argued by Wu et al., GCNs are merely *low-pass filters* associated with $\tilde{\mathbf{S}}$, while different designs of GCNs correspond to different choices of propagation matrices $\mathbf{S}$ and in turn different low-pass filters. For instance, Kipf and Welling [22] derive a propagation matrix $\mathbf{S}_1 = \mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ based on the first-order Chebyshev approximation, which implies the low-pass filter $g_1(\lambda_i) = (2 - \lambda_i)^K$ associated with $\tilde{\mathbf{S}}_1 = (2\mathbf{I} - \mathbf{L}_\mathcal{G})^K$, where λ_i denotes the ith eigenvalue of normalized graph Laplacian $\mathbf{L}_\mathcal{G}$[5]; to avoid numerical instabilities, they also propose to use a different GCN design by adding self-loops to each node so that $\mathbf{S}_2 = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ ($\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}$) and low-pass filter $g_2(\tilde{\lambda}_i) = (1 - \tilde{\lambda}_i)^K$ associated with $\tilde{\mathbf{S}}_2 = (\mathbf{I} - \tilde{\mathbf{L}}_\mathcal{G})^K$.

This simplified form provides us with a good starting point to compare our work with GCNs. We first recover a similar form as in Equation (24) with the RKHS defined in Proposition 3.1.

Proposition 3.5. *With the RKHS in Proposition 3.1, Tikholov regularization is equivalent to learning a* $f(\mathbf{x}) = (\mathbf{K}_\mathrm{O}^{1/2}\Phi_2) \cdot (\mathbf{K}_\mathrm{E}^{1/2}\Phi_1)^\top \cdot \mathbf{x}$ *with the following loss function,*

$$\arg\min_{\Phi} \frac{1}{N}\sum_i^N V(f(\mathbf{x}_i), \mathbf{y}_i) + \lambda \cdot \left(\|\Phi_1\|_F^2 + \|\Phi_2\|_F^2\right). \tag{25}$$

This proposition tells that the proposed graph-based regularization is implicitly a low-pass filtering process as well. However, it utilizes a different low-pass filter from those in GCNs: $g_3(\lambda_i) = 1/\sqrt{1 + \gamma\lambda_i}$ associated with $\tilde{\mathbf{S}}_3 = \mathbf{K}_{\mathrm{E/O}}{}^{1/2} = (\mathbf{I} + \gamma\mathbf{L}_\mathcal{G})^{-1/2}$. The benefits of the low-pass filter g_3 are

(1) it is positive and monotonically decreasing across the whole spectrum of graph Laplacian compared with polynomial filters (especially, g_2),
(2) it does not blow up near zero as some other graph filters (e.g., g_1), which could lead to a larger generalization gap according to Verma and Zhang [46],
(3) it applies to both normalized and unnormalized graph Laplacian (i.e., the eigenvalues $\{\lambda_i\}$ are not necessarily bounded in $[0, 2]$ as required by g_1 and g_2).

Moreover, the receptive field of convolution filter $\tilde{\mathbf{S}}_3 = (\mathbf{I} + \gamma\mathbf{L}_\mathcal{G})^{-1/2}$ is not limited within certain K-hop neighborhood, while it is still "localized" in the sense that the contribution of neighbors decays with their distances (see Equation (22)). However, directly applying such graph filter by matrix multiplication as opposed to its equivalent regularization approach is computationally inefficient, because it involves calculating the inverse of the graph Laplacian matrix.

[5]Larger eigenvalue λ_i of graph Laplacian $\mathbf{L}_\mathcal{G}$ means higher frequency of the corresponding eigenvector as a (basis) graph signal and therefore $g(\lambda_i)$ should be smaller (but still positive) for low-pass filtering.

3.5.3 Graph-based Regularization as Joint Learning. Besides user-item relations, modeling item-item relations is another popular approach for recommendation (e.g., a memory-based version [29] and a model-based version [53]). Learning with the proposed regularization can also be considered as a joint learning process of these two approaches: the last two terms in Equation (17) are modeling item-item similarities encoded in the given graph $\mathcal{G}$. Similar joint learning techniques have been successfully used in the context of matrix factorization (MF) [7, 8, 26, 61]. For example, CoFactor [26] jointly decomposes the user-item interaction matrix and the item-item co-occurrence matrix with shared item latent factors. Our article extends this line of research to general recommendation models and provides a theoretical justification for the proposed joint learning approach.

4 EMPIRICAL STUDY

4.1 Research Questions

We are interested in whether and, if so, how the proposed regularization approach can improve recommendation performances. This principal research question is broken down into the following specific ones to guide our empirical study.

RQ1. How does the proposed regularization approach affect the performances of recommendation models? Is the regularization on both the input and the output embedding layers necessary?
RQ2. In what circumstances does the proposed regularization perform better?
RQ3. Why does the proposed graph-based regularization work?
RQ4. How do different forms of the graphs affect the effectiveness of the proposed regularization? Are the proposed guidelines for constructing the graphs valid?
RQ5. How do different settings of hyper-parameters (i.e., λ and γ) affect the effectiveness of the proposed regularization?

4.2 Experimental Setups

In our experiments, we apply the proposed graph-based regularization approach to two state-of-the-art neural recommendation models introduced in Section 2.2, *Mult-DAE* [27] and *SASRec* [20] (both with properly tuned ℓ_2 regularization on all the parameters) on five publicly available datasets. Mult-DAE is tested on the following datasets where the ordering of user behaviors is ignored.

- **ML-20M** [12]. This is a widely used benchmark for collaborative filtering methods. We keep ratings of four or higher and interpret them as implicit feedback as in Liang et al. [27]. We filter out users and items with fewer than five interactions.
- **LastFM** [6]. This dataset contains users' playback counts of artists. A user is considered to "like" an artist if s/he has played songs by that artist for more than 60 times (it accounts for over 60% of user-artist records in this dataset). We filter out users with fewer than five liked artists and artists fewer than 20 "likes."

Meanwhile, SASRec as a sequential recommendation model is tested on the following datasets where the ordering of user behaviors is available.

- Amazon **Beauty** and **Games**. These two datasets are part of a series of Amazon review datasets from Reference [33], which are widely used benchmarks for e-commerce recommender systems. In our experiments, we use the proprocessed version from Kang and McAuley [20].

Table 1. Statistics of Datasets (After Preprocessing)

(a)

	#users	#items	#actions	sparsity	# held-out users
ML-20M	136,674	13,681	10.0M	0.53%	20,000 (val)/20,000 (test)
LastFM	228,986	62,978	9.9M	0.07%	25,000 (val)/25,000 (test)

(b)

	#users	#items	#actions	avg. #actions /user
Beauty	52,024	57,289	0.4M	7.6
Games	31,013	23,715	0.3M	9.3
Taobao-100K	100,844	212,659	7.0M	69.0

- **Taobao-100K.** This dataset contains user behaviors during November 25 to December 03, 2017 on a popular Chinese online shopping website, *Taobao*. A subset of 100k users are sampled from the original dataset [74] for experiments due to limited computational resources. We regard *clicks, adding to shopping carts, purchases* as implicit feedback and filter out product items with fewer than 10 behaviors.

Table 1 summarizes statistics of the datasets after data preprocessing and data partitioning.

We adopt the default model architectures and training for *Mult-DAE* and *SASRec* as in their original papers. For Mult-DAE, the DAE architecture is $[m \rightarrow 200 \rightarrow m]$. For SASRec, the embedding size is set to 50 and two self-attention blocks with the learned positional embedding are used. The maximum sequence length is set to 200 for Taobao-100K and 50 for the other two datasets. If not otherwise specified, then we use a kNN graph based on the Jaccard similarity and k is set to five. The hyper-parameters in the proposed regularization term is reported in Table 5.

For fair comparison, we also follow the same evaluation protocols as used in *Mult-DAE* [27] and *SASRec* [20]. For the former, a certain number of users are held out for validation and test; for the latter, all users are used at training, while the last two behaviors of each user are held out. These two evaluation protocols correspond to strong and weak generalization experiments in Marlin [32], respectively. As for evaluation metrics, Reference [27] uses ranking metrics *Recall@20*, *Recall@50*, and *nDCG@100* to measure the performance of ranking all unclicked items. In contrast, Reference [20] randomly samples 100 negative items for each user, and ranks these items with the ground-truth item (101 items in total), based on which *HR*@10 and *nDCG*@10 are evaluated. What is slightly different from common practices is that the metric *Recall@K* used in Reference [27] is computed as the following for each user u:

$$Recall@K = \frac{\sum_{k=1}^{K} \mathbb{I}(\omega_u(k) \in \mathcal{I}_u)}{\min(K, |\mathcal{I}_u|)},$$

where $\omega_u(k)$ is the kth item recommended to user u and the denominator is the minimum of K and the number of relevant items $|\mathcal{I}_u|$ (actually making itself a maximum of *Precision@K* and *Recall@K*). The reader can find more details on the evaluation protocols in their original papers [20, 27]. We use nDCG@K evaluated on the validation set to tune hyper-parameters for baselines and the proposed, and then report the results achieved by the best configurations.

4.3 Results and Analysis

*4.3.1 Main Results (**RQ1**).* Table 2 summarizes recommendation performances when we apply different regularization approaches to Mult-DAE. We can see that the proposed graph-based

Table 2. Recommendation Performance with *Mult-DAE* as Baseline

Model	ML-20M			LastFM		
	Recall@20	Recall@50	nDCG@100	Recall@20	Recall@50	nDCG@100
Baseline	0.3906	0.5224	0.4219	0.2863	0.4321	0.3629
GCN	0.3921	0.5251	0.4234	0.2931	0.4398	0.3685
GraReg_{E}	$0.3921^{\dagger}$	$0.5258^{\dagger}$	$0.4236^{\dagger}$	$0.2915^{\dagger}$	$0.4379^{\dagger}$	$0.3677^{\dagger}$
GraReg_{O}	$0.3930^{\dagger}$	$\mathbf{0.5285}^{\dagger\ddagger}$	$0.4235^{\dagger}$	$0.2917^{\dagger}$	$0.4374^{\dagger}$	$0.3676^{\dagger}$
$\text{GraReg}_{\text{EO}}$	$\mathbf{0.3941}^{\dagger\ddagger}$	$\mathbf{0.5284}^{\dagger\ddagger}$	$\mathbf{0.4249}^{\dagger\ddagger}$	$\mathbf{0.2952}^{\dagger\ddagger}$	$\mathbf{0.4414}^{\dagger\ddagger}$	$\mathbf{0.3715}^{\dagger\ddagger}$
Improv.	0.90%	1.14%	0.70%	3.11%	2.15%	2.36%

GCN denotes using graph convolution layers for both input and output. $\textit{GraReg}_{E}$ and $\textit{GraReg}_{O}$ represent the cases where $\gamma_O = 0$ and $\gamma_E = 0$, respectively, while $\textit{GraReg}_{EO}$ applies the proposed regularization on both input and output embedding layers of *Mult-DAE*. *Improv.* shows improvement over *Baseline*. † and ‡ denote statistically significant ($p < 0.05$ by the Student's t-test) improvements over *Baseline* and *GCN*, respectively.

Table 3. Recommendation Performance with SASRec as Baseline

Model	Beauty		Games		Taobao-100K	
	HR@10	nDCG@10	HR@10	nDCG@10	HR@10	nDCG@10
TransRec [14]	0.4608	0.3020	0.6838	0.4557	-	-
GRU4Rec^{+} [16]	0.3949	0.2556	0.6599	0.4759	-	-
Caser [43]	0.4264	0.2547	0.5282	0.3214	-	-
MF	0.3893	0.2424	0.6033	0.4003	-	-
$\text{GraReg}_{\text{MF}}$	**0.4056**	**0.2533**	**0.6497**	**0.4212**	-	-
SASRec	0.4750	0.3206	0.7402	0.5397	0.7984	0.6106
$\text{GraReg}_{\text{SASRec}}$	$\mathbf{0.4986}^{\dagger}$	$\mathbf{0.3330}^{\dagger}$	$\mathbf{0.7607}^{\dagger}$	$\mathbf{0.5528}^{\dagger}$	$\mathbf{0.8138}^{\dagger}$	$\mathbf{0.6310}^{\dagger}$
Improv.	4.96%	3.87%	2.76%	2.43%	1.92%	3.35%

$\textit{GraReg}_{SASRec}$ applies graph regularization on the embedding layer of SASRec. *Improv.* denotes the performance gains over *SASRec*. † denotes statistically significant ($p < 0.01$ by the Student's t-test) improvements over *SASRec*. Results of TransRec, GRU4Rec^{+} and Caser are taken from Kang and McAuley [20]. For reference, we also report the performance of MF (low-rank matrix factorization) models without and with GraReg (denoted as MF and $\text{GraReg}_{\text{MF}}$, respectively).

regularization approach leads to improvements on both the ML-20M and LastFM datasets. It is worth mentioning that MovieLens datasets are much denser than most other datasets in both academia and industry. Thus, the around one percent improvement on ML-20M, despite seemingly small, is notable, since the effect of regularization techniques is often believed to diminish as more training data is used. We also observe that, while it is beneficial to apply the proposed regularization to input or output embedding layers alone (i.e., GraReg_{E} and GraReg_{O}), they also complement each other. Moreover, our approach outperforms the one that uses GCN layers (with the same graph structure) for both input and output. We performed the one-sided Student's t-test. Results suggest that the improvements over both the ℓ_2 regularization and the GCN approach are both statistically significant at the 0.05 level.

As shown in Table 3, we obtain consistent results when experimenting with SASRec. The proposed graph regularization approach achieves up to 4.96% and 3.87% of improvements in nDCG@10 and HR@10, respectively. The results of several state-of-the-art baselines are quoted from Kang and McAuley [20]. This demonstrates that SASRec is a competitive baseline model and the proposed regularization is able to boost its performance even further.

Finally, one can observe that the performance gains on the MovieLens-20M dataset over Mult-DAE are smaller than the others. We hypothesize the main reason is that GraphReg is especially

Table 4. Recommendation Performance in *nDCG@100* on ML-20M and LastFM After Removing the Top-k Popular Items in the Test Stage (Including when Calculating the Normalization Factors for DCG)

k	ML-20M			LastFM		
	Baseline	GraReg	Improv.	Baseline	GraReg	Improv.
0	0.4219	**0.4249**	0.70%	0.3629	**0.3715**	2.36%
100	0.3458	**0.3506**	1.39%	0.3213	**0.3313**	3.12%
500	0.2832	**0.2892**	2.09%	0.2800	**0.2919**	4.25%
1,000	0.2531	**0.2622**	3.62%	0.2561	**0.2692**	5.13%
2,000	0.2205	**0.2328**	**5.57%**	0.2346	**0.2489**	**6.07%**

Table 5. The Hyper-parameters Used to Report the Results for the Proposed Regularization

(a)

	λ	γ_E	γ_O
ML-20M	2×10^{-5}	2	0.2
LastFM	1×10^{-5}	1	1

(b)

	λ	$\gamma = \gamma_E = \gamma_O$
Beauty	1×10^{-7}	2
Games	1×10^{-6}	4
Taobao-100K	1×10^{-8}	8

Table 6. Recommendation Performance in Terms of *nDCG@100* for Tailed Users

User Group	ML-20M			LastFM		
	Baseline	$\text{GraReg}_{\text{EO}}$	Improv.	Baseline	$\text{GraReg}_{\text{EO}}$	Improv.
bottom 1/32	0.1778	**0.1877**	**5.55%**	0.3754	**0.3928**	**4.61%**
bottom 1/16	0.2072	**0.2120**	2.31%	0.3515	**0.3651**	3.88%
bottom 1/8	0.2511	**0.2545**	1.36%	0.3520	**0.3617**	2.76%
overall	0.4219	**0.4248**	0.69%	0.3629	**0.3715**	2.36%

Users are ranked by the total popularity of their clicked items.

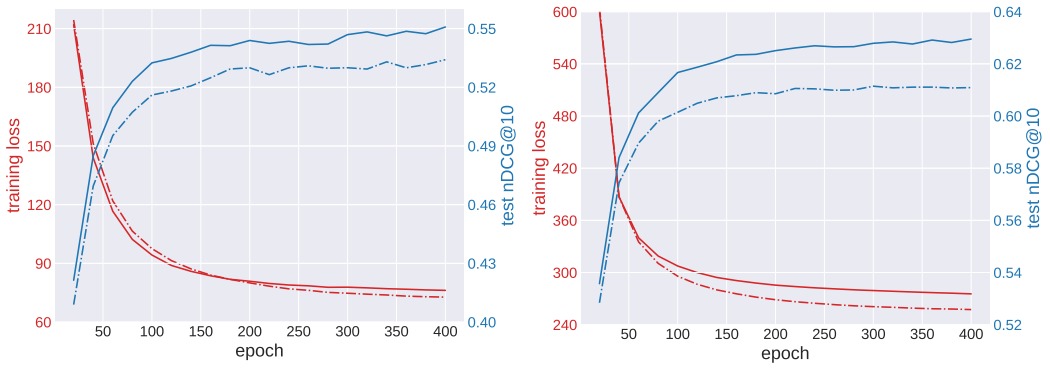
beneficial for sparser datasets (see Table 1 for some statistics), on which models' generalization capacity becomes more important.

*4.3.2 Performance on Long-tail Users and Items (**RQ2**).* As shown in Table 4, the proposed regularization approach achieves more substantial improvements when recommending less popular items. This might be partially because popular items pretty much dominate the top of recommendation lists, which allows less room for personalization. More importantly, it is also because less popular items have less related training samples so that it becomes more significant to be able to transfer knowledge from other correlated items with the graph regularization.

Meanwhile, motivated by the analysis on Equation (12) in Section 3.1, we sort users by the total popularity counts of their clicked items and report the performance results for users on the tail. Results in Table 6 show that the proposed regularization is more beneficial to users who have less common behaviors with the majority. Although it only gets a 0.69% relative improvement in NDCG@100 on the ML-20M dataset overall, we can observe a 5.55% relative gain for the bottom 1/32 users.

Therefore, our proposed graph-based regularization approach works better for *long-tail items and users* for which less training samples are available.

*4.3.3 Further Analysis (**RQ3**).* We plot the learning curves of SASRec on Games and Taobao-100K in Figure 1. On both datasets, although the training loss at convergence increases, we can



(a) Games (b) Taobao-100K

Fig. 1. Learning curves of SASRec with the ℓ_2 regularization (dash-dotted line) and with the proposed graph regularization (solid line) on Games and Taobao-100K.

Table 7. The Performance of Predicting Item Categories in Taobao-100K with Their Embeddings Obtained from Different Models

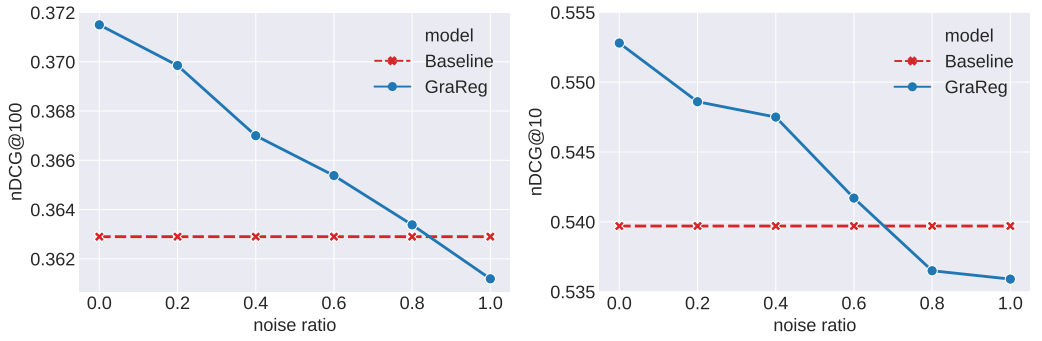
	Skipgram	Deepwalk	Baseline	GraReg
NMI	0.534	0.561	0.581	**0.607**
Accuracy	0.485	0.485	0.572	**0.609**

achieve higher nDCG@10 on the test sets. This suggests that the proposed regularization can indeed *reduce overfitting and lead to better generalization*. Besides, we can observe that the training loss of the proposed approach decreases more quickly at the beginning. This is because the graph-based regularization can also provide useful training signals (which amounts to item-item losses) and help with optimization by encouraging more well-structured parameters as opposed to such "uninformative" regularization methods as the ℓ_2 penalties and dropout.

We also investigate the quality of the embeddings, which is crucial for neural recommendation models. We leverage the category information available in Taobao-100K where each item belongs to one of the 3,929 categories. We first run K-means to cluster the embeddings into 3,929 classes and compute NMI (normalized mutual information) between the clustering and ground-truth categories. Also, we split items into training (50%) and test (50%), and train a softmax classifier on the training set to predict items' categories with their embeddings as features. Since the proposed method can be considered as joint training with item-based models as discussed in Section 3.5, we also compare with two item embedding models, Skipgram [34] and Deepwalk [36]. As in Table 7, embeddings learned from SASRec as a state-of-the-art sequence recommendation model outperform those from Skipgram and Deepwalk; the proposed regularization complements sequence modeling in terms of obtaining *high-quality embeddings*.

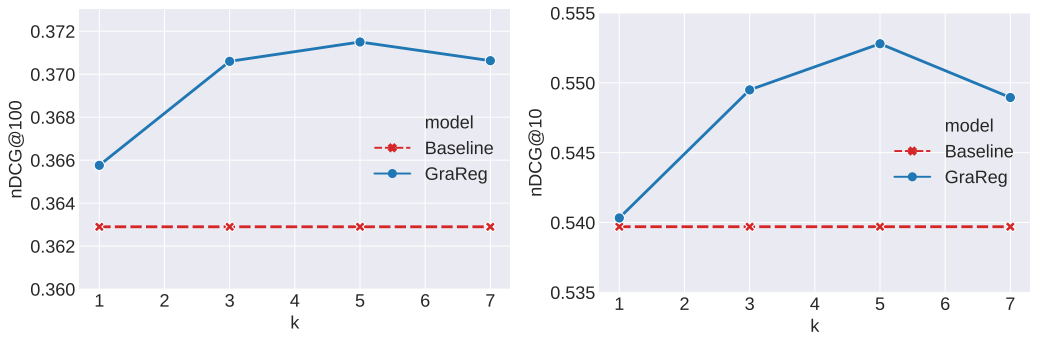
*4.3.4 Effect of Different Graphs (**RQ4**).* In Section 3.3, we discuss two guidelines for graph construction: "accurate edges" and "balanced distribution." We intended to empirically validate the two guideline in our experiments.

Accurate edges. We gradually replace edges with pairs of uniformly random items and compare the performance when using this noisy version of the graph. Figure 2 shows that the performance indeed declines as the proportion of noisy edges increases. This might not be surprising, but the



(a) LastFM (b) Games

Fig. 2. Performance of the corrupted graphs at different noise ratios. Solid lines and dashed lines denote the proposed approach, GraReg, and the corresponding baseline models (Mult-DAE and SASRec), respectively.



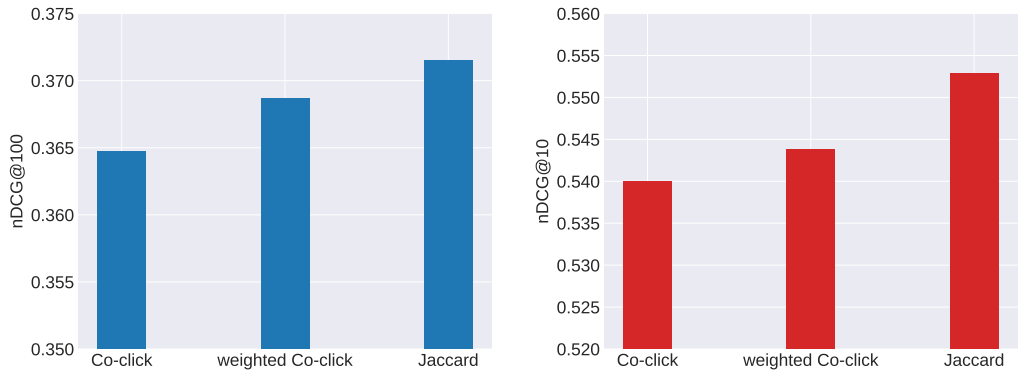
(a) LastFM (b) Games

Fig. 3. Performance of the kNN graphs across different choices of k. Solid lines and dashed lines denote the proposed approach, GraReg, and the corresponding baseline models (Mult-DAE and SASRec), respectively.

curves also suggest that the proposed graph-based regularization is robust to the quality of graphs: it still outperforms the baseline until around 60% to 80% of edges are corrupted.

In addition, Figure 3 shows the results when using different choices of cutoff k. The performance in nDCG first goes up and then gradually decreases as k increases. This can be explained as the classic bias-variance tradeoff: the model becomes more regularized as the number of edges increases; there might be more inaccurate edges at the same time.

Balanced Distribution. We compare the performances of the kNN graphs constructed with the number of co-occurrences in users' behavior sequences (*Co-click*) and the Jaccard similarity (*Jaccard*), respectively. As discussed in Section 3.3, the former might be highly biased towards popular items, since they have more chances to be co-clicked with other items, while the Jaccard similarity is (equivalently) normalized by item popularity. We are also interested in an alternative solution to balance degree distributions, which uses the kNN graph based on co-clicks while weighting edges proportional to the inverse item popularity in the loss function (*weighted Co-click*). As shown in Figure 4, *Jaccard* outperforms the other counterparts. Although *weighted Co-click* avoids popular items from dominating the loss, tailed items might have very little chance of being retained due to the top-k truncation. For instance, on the Games dataset, only 67.1% of items have appeared in the top-k list of more than one item compared with 96.2% in *Jaccard*.



(a) LastFM (b) Games

Fig. 4. The performance of different construction methods. *Co-click* and *Jaccard* denote the kNN graphs based on the number of co-occurrences in user behavior sequences (i.e., $|\mathcal{U}_i \cap \mathcal{U}_j|$, where $\mathcal{U}_i$ denotes the set of users who have clicked item i and so forth) and the Jaccard similarity (i.e., $|\mathcal{U}_i \cap \mathcal{U}_j|/|\mathcal{U}_i \cup \mathcal{U}_j|$), respectively. *Weighted Co-click* represents weighting the edges of *Co-click* with inverse item popularity in the loss function.

*4.3.5 Effect of Hyper-parameters (**RQ5**).* We plot the performance of *GraReg* on SASRec with regard to the hyper-parameters (i.e., λ and $\gamma = \gamma_E = \gamma_O$) in Figure 5. We can see that, with a properly tuned weight of the overall regularization term λ, the proposed regularization method is not very sensitive to the added hyper-parameter γ that controls the contribution of graphs. In other words, compared with the ℓ_2 regularization, our proposed approach not only incurs no extra computational burden but also does not require much additional hyper-parameter tuning.

4.4 Discussion

In this section, we mainly discuss when the proposed GraReg is beneficial. Broadly speaking, similar to other regularization techniques, there is generally a tradeoff between biases (e.g., the graph might be flawed as discussed previously) and variances (to improve robustness and avoid overfitting) when imposing GraReg to recommendation models. However, extensive theoretical and empirical studies show that GraReg is especially effective in the following cases:

- **Sparse data.** If data is too sparse, then neural models are prone to memorize all the labels and have very poor generalization capacities. However, generalization is of crucial importance in recommender systems, since they are generally expected to discover new potential user interests. Comparison between results on the ML-20M dataset and the LastFM dataset clearly demonstrate this point.
- **Unbalanced data (long-tailed items/users).** In addition to the scale of data, the distribution is important as well. If the data were highly skewed, then items or users with relatively less training samples would often be "ignored" by data-driven training processes. This would lead to non-optimal performances of recommender systems and even some fairness issues especially on the user side. Results in Section 4.3.3 suggest that GraReg can dramatically improve model performances on long-tailed items/users and help to mitigate this problem.
- **No relational constraints considered in the model.** As suggested by some recent studies (see Section 5.2), explicitly imposing relational constraints is a very effective way to improving recommendation performances. The proposed GraReg is one approach of this type. Nonetheless, there are other important methods such as GNNs (Graph Neural Networks) [3]

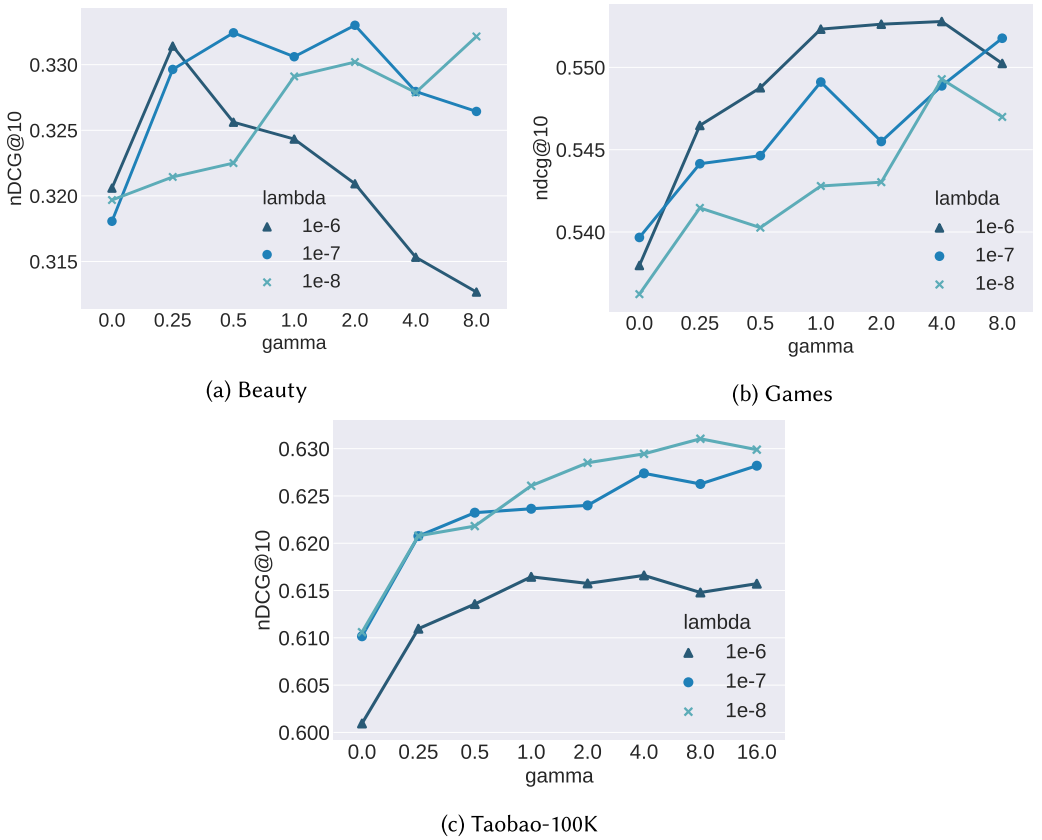


Fig. 5. Sensitivity of the proposed regularization on SASRec w.r.t. hyper-parameters λ and $\gamma = \gamma_E = \gamma_O$.

Table 8. Performance of the Proposed GraReg on a State-of-the-Art GNN-based Sequential Recommendation Model, SR-GNN [57], on the Diginetica Dataset

	All		Non-repeated	
	Recall@20	MRR@20	Recall@20	MRR@20
SR-GNN	0.507	**0.176**	0.441	0.133
SR-GNN w/ GraphReg	**0.511**	**0.176**	**0.449**	**0.136**
Improvement	+0.8%	+0.0%	+1.8%	+2.3%

For more details on the experimental setup, please refer to Wu et al. [57]. *All* means using all the test samples in the original dataset, while *Non-repeated* stands for filtering out those test samples where the target item once appears in the user sequence.

that can also take proper relational inductive biases into consideration, although we have shown that the proposed regularization approach has some better theoretical properties (see Section 3.5.2) and, in practice, is more lightweight to deploy than graph convolutions. If similar inductive biases (especially in the form of *graphs*) are already considered in the model architecture, then we hypothesize that GraReg would be less beneficial. That being said, the results in Table 8 show that GraReg can still aid the performance of GNN-based recommendation models (more significantly for recommending novel items) *as a complement* to GNNs in encoding graph relations into models.

5 RELATED WORK

5.1 General Recommendation

The goal of recommendation algorithms is to infer relevant items to users from their historical behaviors. Matrix factorization (MF) [24] is one of the most popular approaches to recommender systems. MF models aim to learn latent representations for users and items, respectively, and use their inner products to capture users' preference to different items. Recently, deep-learning-based recommendation models have become prevalent. He et al. [15] replace the inner product in MF with a deep neural architecture. Covington et al. [9] propose a deep neural architecture for youtube video recommendation. Autoencoders are applied to collaborative filtering in several works as well, e.g., [19, 27, 58]. Xue et al. [62] propose a deep variant of item-based collaborative filtering (ICF) methods, called DeepICF, which also uses the attention mechanism to attend to discriminative items. The popularity of deep-learning methods is not only attributed to their superior performance but also to their flexibility in incorporating both collaborative signals and content-related signals (e.g., References [13, 49, 68, 70]).

Another line of research aims to leverage deep-learning approaches to capture sequential patterns of user behaviors. The core idea of these works is to model the user behavior sequences analogously to word sequences with various recurrent architectures (e.g., GRU and LSTM) [16, 17, 25, 65]. Other than recurrent neural networks, inspired by their successes in text modeling, CNN and self-attention are also introduced to capture the temporal patterns in user behaviors [20, 41, 43, 67]. For example, Tang and Wang [43] and Yuan et al. [67] employ the convolutional filters to learn the sequential patterns in user behaviors, while SASRec [20] and BERT4Rec [41] adopt the self-attention blocks in Transformer [45] with different direction settings.

The missing data problem is an intrinsic and important issue in recommender systems [28, 30]. We can only observe users' feedbacks (e.g., ratings, clicks, purchases) on a small subset of items, and other unobserved items can be either disliked or unknown. Most recommendation models treat all the unobserved items equally. However, this assumption may sometimes be troublesome, because unobserved items are by no means all irrelevant to users (otherwise, there is no point making recommendations). Pan et al. [35] propose to use handcrafted weighting schemes and negative sampling strategies to overcome this issue. Zhang et al. [71] propose to use a path-based model to propagate users' behaviors over graphs to produce more informative training labels. The proposed approach tackles this issue from a regularization perspective; Section 3.5.1 shows that our approach implicitly propagates sparse multi-hot encodings, which assume all unobserved items are irrelevant (0), into smoother distributions over items.

5.2 Graph-based Recommendation

Path-based recommendation models have been widely studied in the literature. Yu et al. [66] use meta-paths [42] in heterogeneous information networks (HINs) to diffuse user-item preferences and then exploit MF techniques to calculate latent vectors for users and items for implicit recommendation. Shi et al. [39] extend this work to weighted paths for explicit recommendation. Zhao et al. [72] propose to exploit MF to extract latent features from different meta-paths and then use factorization machines with Group lasso to combine these features. Shi et al. [38] propose a heterogeneous network embedding method for recommendation (HERec) based on meta-path guided random walks. Wang et al. [47] propose an end-to-end framework called RippleNet, which leverages knowledge graph embeddings to propagate user preferences through paths in knowledge graphs with attention mechanism.

Recently, there has also been extensive work on using graph convolutions in recommender systems to model high-order proximity among users and items (e.g., References [4, 11, 48, 51,

52, 64, 73]). SpectralCF [73] exploits a spectral convolution operation on the bipartite graph of users and items to model high-order proximity in the spectral domain. Wang et al. [52] propose Neural Graph Collaborative Filtering (NGCF), which exploits the user-item graph structure by propagating embeddings on it. Fan et al. [11] propose a graph neural network architecture to incorporate social networks with user-item interactions. Besides our theoretical comparison with graph convolutions in Section 3.5.2, the proposed regularization approach is a more lightweight "plugin" at training and introduces no overhead afterwards in practice.

5.3 Regularizing Recommendation Models

Ma et al. [31] propose to regularize users' latent vectors with social relationships. CoFactor [26] regularizes MF with item co-occurrence by jointly decomposing the user-item interaction matrix and the item-item co-occurrence matrix with shared item latent factors. Tran et al. [44] and Hop-Rec [63] share similar ideas in exploiting second-order and higher-order proximity among users and items to regularize MF models. More specifically, in Hop-Rec, the similarity between two items/users is defined by the overlaps of their k-hop neighbors and embeddings for similar items/users are regularized to be similar. RCF [61] and MKR [50] both propose to jointly train recommendation models and knowledge embedding models where the latter tasks can effectively regularize the former ones with domain knowledge encoded in the knowledge graphs.

Our work is most related to Rao et al. [37], which studies some theoretical properties when applying pair-wise constraints on user and item latent vectors in MF models. Compared with Rao et al., the contribution of this article lies in the following two aspects: (1) our work aims to provide researchers and practitioners in this community with approachable insights on why to use the graph regularization for embedding layers, and thus the theoretical analysis in this article has more practical implications; (2) more importantly, this work extends MF to a more general case with very different theoretical techniques, and we also conduct extensive empirical studies with two state-of-the-art deep-learning-based recommendation models.

6 CONCLUSION

In this article, we study regularization on embedding layers by focusing on a typical and practical problem, *item recommendation*. We first show, with a simplified model, that the widely used ℓ_2 regularization for regular neural layers has some issues when applied to embedding layers. The main reason is that ℓ_2 regularization assumes the layer being regularized lies in the Euclidean space, while the inner product and distance in the Euclidean space cannot capture correlations between items. Motivated by our analysis, we propose a graph-based regularization term to serve as a counterpart of ℓ_2 penalties for embedding layers, which is applicable to general neural recommendation models, and discuss its relationships to other approaches from different perspectives. Extensive experiments show the effectiveness of the proposed approach and support our theoretical analysis.

For future work, we plan to explore more principled approaches to construct graphs from data. We are also interested in developing further theoretical results for the proposed regularization method with more complicated recommendation models.

APPENDIX

A PROOFS

A.1 Proof for Proposition 3.1

The functions in the induced Hilbert space is of the form

$$\begin{aligned} f(\mathbf{x}) &= \sum_{i=1}^{p} K(\overline{\mathbf{x}}_\mathbf{i}, \mathbf{x})\mathbf{c}_\mathbf{i} = \sum_{i=1}^{p} \mathbf{K}_\mathbf{O}\mathbf{c}_\mathbf{i} \cdot \left(\overline{\mathbf{x}}_\mathbf{i}^\top \mathbf{K}_\mathbf{E}\mathbf{x}\right) \\ &= \mathbf{K}_\mathbf{O} \sum_{i=1}^{p} \left(\mathbf{c}_\mathbf{i} \cdot \overline{\mathbf{x}}_\mathbf{i}^\top\right)\mathbf{K}_\mathbf{E}\mathbf{x} = (\mathbf{K}_\mathbf{O}\mathbf{C}\overline{\mathbf{X}}\mathbf{K}_\mathbf{E})\mathbf{x}. \end{aligned} \tag{26}$$

Since $\mathbf{K}_\mathbf{E}$ and $\mathbf{K}_\mathbf{O}$ are non-singular, matrix $\Theta = \mathbf{K}_\mathbf{O}\mathbf{C}\overline{\mathbf{X}}\mathbf{K}_\mathbf{E}$ can take any value in $\mathbb{R}^{m\times m}$. Plugging kernel function $K(\overline{\mathbf{x}}_\mathbf{i}, \overline{x}_i) = (\overline{\mathbf{x}}_\mathbf{i}^\top \mathbf{K}_\mathbf{E}\overline{\mathbf{x}}_\mathbf{j}) \cdot \mathbf{K}_\mathbf{O}$ into Equation (2), we obtain the induced norm

$$\begin{aligned} \|f\|_\mathcal{H} &= \sqrt{\sum_{i=1}^{p}\sum_{j=1}^{p'} \left(\overline{\mathbf{x}}_\mathbf{i}^\top \mathbf{K}_\mathbf{E}\overline{\mathbf{x}}_\mathbf{j}\right)\cdot\left(\mathbf{c}_\mathbf{i}^\top \mathbf{K}_\mathbf{O}\mathbf{c}_j\right)} \\ &= \sqrt{\sum_{i=1}^{p}\sum_{j=1}^{p'} [\overline{\mathbf{X}}\mathbf{K}_\mathbf{E}\overline{\mathbf{X}}^\top]_{i,j} \cdot [\mathbf{C}^\top\mathbf{K}_\mathbf{O}\mathbf{C}]_{i,j}} \\ &= \sqrt{\mathrm{tr}((\overline{\mathbf{X}}\mathbf{K}_\mathbf{E}\overline{\mathbf{X}}^\top)\cdot(\mathbf{C}^\top\mathbf{K}_\mathbf{O}\mathbf{C}))} \\ &= \sqrt{\mathrm{tr}\left[(\overline{\mathbf{X}}\mathbf{K}_\mathbf{E})\mathbf{K}_\mathbf{E}^{-1}(\overline{\mathbf{X}}\mathbf{K}_\mathbf{E})^\top \cdot (\mathbf{K}_\mathbf{O}\mathbf{C})^\top\mathbf{K}_\mathbf{O}^{-1}(\mathbf{K}_\mathbf{O}\mathbf{C})\right]} \\ &= \sqrt{\mathrm{tr}\left[\mathbf{K}_\mathbf{E}^{-1}(\mathbf{K}_\mathbf{O}\mathbf{C}\overline{\mathbf{X}}\mathbf{K}_\mathbf{E})^\top\mathbf{K}_\mathbf{O}^{-1}(\mathbf{K}_\mathbf{O}\mathbf{C}\overline{\mathbf{X}}\mathbf{K}_\mathbf{E})\right]} \\ &= \sqrt{\mathrm{tr}\left(\mathbf{K}_E^{-1}\Theta^\top\mathbf{K}_O^{-1}\Theta\right)}, \end{aligned} \tag{27}$$

where the second to the last line is due to the cyclic property of trace operator.

A.2 Proof for Proposition 3.2

We will show that the convex envelope of $f((\mathbf{A},\mathbf{B})) = \mathrm{tr}(\mathbf{AB})$ over the set $\mathcal{M} = \{(\mathbf{A},\mathbf{B}) \in \mathbb{S}_n \times \mathbb{S}_n | O \preceq \mathbf{A},\mathbf{B} \preceq I, \sigma_n(\mathbf{A}) + \sigma_n(\mathbf{B}) \geq 1\}$ ($\sigma_1(\cdot), \ldots, \sigma_n(\cdot)$ are eigenvalues of matrices in descending order) is $\mathrm{tr}(\mathbf{A}) + \mathrm{tr}(\mathbf{B}) - n$ with the fact that bi-conjugate function f^{**} is the (closed) convex envelope of f.

A.2.1 Conjugate Function of $f((\mathbf{A},\mathbf{B}))$. According to the definition, the conjugate function of $f((\mathbf{A},\mathbf{B}))$ is

$$f^*((\mathbf{C},\mathbf{D})) = \sup_{(\mathbf{A},\mathbf{B})\in\mathcal{M}} \mathrm{tr}(\mathbf{AC}) + \mathrm{tr}(\mathbf{BD}) - \mathrm{tr}(\mathbf{AB}). \tag{28}$$

The domain of f^* is $\mathbb{S}_n \times \mathbb{S}_n$, since its value is bounded everywhere.

We consider a subset of $\mathcal{M}$ denoted by $\mathcal{M}'$, where $\mathbf{A}$ and $\mathbf{B}$ have the same eigendecomposition as $\mathbf{C}$ and $\mathbf{D}$, respectively (i.e., $\mathbf{A} = \mathbf{U}\Sigma_\mathbf{A}\mathbf{U}^\top$ and $\mathbf{C} = \mathbf{U}\Sigma_\mathbf{C}\mathbf{U}^\top$ for some orthogonal matrix $\mathbf{U}$ and the same with $\mathbf{B}$), so that $\mathrm{tr}(\mathbf{AC}) = \sum_{i=1}^n \sigma_i(\mathbf{A})\sigma_i(\mathbf{C})$ and $\mathrm{tr}(\mathbf{BD}) = \sum_{i=1}^n \sigma_i(\mathbf{B})\sigma_i(\mathbf{D})$. Meanwhile, thanks to von Neumann's trace theorem, $\mathrm{tr}(\mathbf{AB}) \leq \sum_{i=1}^n \sigma_i(\mathbf{A})\sigma_i(\mathbf{B})$ with equality when $\mathbf{A}$ and $\mathbf{B}$ have the same eigendecomposition. Thus,

$$
\begin{aligned}
f^*((\mathbf{C},\mathbf{D})) &\geq \sup_{(\mathbf{A},\mathbf{B})\in\mathcal{M}'} \operatorname{tr}(\mathbf{AC}) + \operatorname{tr}(\mathbf{BD}) - \operatorname{tr}(\mathbf{AB}) \\
&= \sup_{(\mathbf{A},\mathbf{B})\in\mathcal{M}'} \sum_{i=1}^{n} \sigma_i(\mathbf{A})\sigma_i(\mathbf{C}) - \sum_{i=1}^{n} \sigma_i(\mathbf{B})\sigma_i(\mathbf{D}) - \operatorname{tr}(\mathbf{AB}) \\
&\geq \sup_{(\mathbf{A},\mathbf{B})\in\mathcal{M}'} \sum_{i=1}^{n} (\sigma_i(\mathbf{A})\sigma_i(\mathbf{C}) + \sigma_i(\mathbf{B})\sigma_i(\mathbf{D}) - \sigma_i(\mathbf{A})\sigma_i(\mathbf{B})) \\
&\geq \sum_{i=1}^{F} (\sigma_i(\mathbf{C}) + \sigma_i(\mathbf{D}) - 1) + \sum_{i=F+1}^{n} \max(\sigma_i(\mathbf{C}), \sigma_i(\mathbf{D})),
\end{aligned}
$$

where F slices indices into two parts such that $\sigma_i(\mathbf{C}) \geq 1$ and $\sigma_i(\mathbf{D}) \geq 1$ if and only if $1 \leq i \leq F$. In the last line of Equation (29), we plug in a particular solution $(\mathbf{A}^*, \mathbf{B}^*)$ in $\mathcal{M}'$: for $1 \leq i \leq F$, $\sigma_i(\mathbf{A}^*) = 1$ and $\sigma_i(\mathbf{B}^*) = 1$; for $i > F$, $\sigma_i(\mathbf{A}^*) = 1$ and $\sigma_i(\mathbf{B}^*) = 0$ if $\sigma_i(\mathbf{C}) \geq \sigma_i(\mathbf{D})$, while, otherwise, $\sigma_i(\mathbf{A}^*) = 0$ and $\sigma_i(\mathbf{B}^*) = 1$.

A.2.2 Conjugate Function of $f^*((\mathbf{C},\mathbf{D}))$. By definition,

$$
f^{**}((\mathbf{A},\mathbf{B})) = \sup_{(\mathbf{C},\mathbf{D})} \operatorname{tr}(\mathbf{AC}) + \operatorname{tr}(\mathbf{BD}) - f^*((\mathbf{C},\mathbf{D})). \tag{30}
$$

We denote $l(\mathbf{C},\mathbf{D}) = \operatorname{tr}(\mathbf{AC}) + \operatorname{tr}(\mathbf{BD}) - f^*((\mathbf{C},\mathbf{D}))$. By plugging Equation (29) into $l(\mathbf{C},\mathbf{D})$ and assuming $\max(\sigma_i(\mathbf{C}), \sigma_i(\mathbf{D})) = \sigma_i(\mathbf{C})$ for $i > F$ without loss of generality, we have

$$
\begin{aligned}
l(\mathbf{C},\mathbf{D}) \leq& \operatorname{tr}(\mathbf{AC}) + \operatorname{tr}(\mathbf{BD}) - \sum_{i=1}^{F} (\sigma_i(\mathbf{C}) + \sigma_i(\mathbf{D}) - 1) - \sum_{i=F+1}^{n} \max(\sigma_i(\mathbf{C}), \sigma_i(\mathbf{D})) \\
\leq& \sum_{i=1}^{n} \sigma_i(\mathbf{A})\sigma_i(\mathbf{C}) + \sigma_i(\mathbf{B})\sigma_i(\mathbf{D}) - \sum_{i=1}^{F} (\sigma_i(\mathbf{C}) + \sigma_i(\mathbf{D}) - 1) - \sum_{i=F+1}^{n} \max(\sigma_i(\mathbf{C}), \sigma_i(\mathbf{D})) \\
=& \sum_{i=1}^{F} [(\sigma_i(\mathbf{A}) - 1)\sigma_i(\mathbf{C}) + (\sigma_i(\mathbf{B}) - 1)\sigma_i(\mathbf{D}) + 1] \\
&+ \sum_{i=F+1}^{n} [\sigma_i(\mathbf{A})\sigma_i(\mathbf{C}) + \sigma_i(\mathbf{B})\sigma_i(\mathbf{D}) - \max(\sigma_i(\mathbf{C}), \sigma_i(\mathbf{D}))] \\
=& \sum_{i=1}^{F} [(\sigma_i(\mathbf{A}) - 1)\sigma_i(\mathbf{C}) + (\sigma_i(\mathbf{B}) - 1)\sigma_i(\mathbf{D}) + 1] \\
&+ \sum_{i=F+1}^{n} \sigma_i(\mathbf{D})(\sigma_i(\mathbf{A}) + \sigma_i(\mathbf{B}) - 1) + \sum_{i=F+1}^{n} (\sigma_i(\mathbf{A}) - 1)(\sigma_i(\mathbf{C}) - \sigma_i(\mathbf{D})) \\
\leq& \sum_{i=1}^{n} (\sigma_i(\mathbf{A}) + \sigma_i(\mathbf{B}) - 1) + 0 = \operatorname{tr}(\mathbf{A}) + \operatorname{tr}(\mathbf{B}) - n,
\end{aligned} \tag{31}
$$

where the equality holds when $\mathbf{C} = \mathbf{D} = \mathbf{I}$. The first inequality is obtained by plugging Equation (29) into Equation (30). The last inequality in Equation (31) is because $\sigma_i(\mathbf{A}) - 1 \leq 0$, $\sigma_i(\mathbf{B}) - 1 \leq 0$ and $\sigma_i(\mathbf{A}) + \sigma_i(\mathbf{B}) - 1 \geq 0$, while $\sigma_i(\mathbf{C}), \sigma_i(\mathbf{D}) \geq 1$ for $1 \leq i \leq F$ and $\min(\sigma_i(\mathbf{C}), \sigma_i(\mathbf{D})) = \sigma_i(\mathbf{D}) < 1$ for $i > F$ (the equality is achieved if $F = n$). When $\mathbf{C} = \mathbf{D} = \mathbf{I}$, $\mathbf{A}^* = \mathbf{B}^* = \mathbf{I}$ and therefore the equality in Equation (29) also holds in this case. Thus, $f^{**}((\mathbf{A},\mathbf{B})) = \sup_{(\mathbf{C},\mathbf{D})} l(\mathbf{C},\mathbf{D}) = \operatorname{tr}(\mathbf{A}) + \operatorname{tr}(\mathbf{B}) - n$ over the set $\mathcal{M}$.

Therefore, the convex envelope of $\operatorname{tr}(\mathbf{AB})$ over the set $\mathcal{M}$ is $\operatorname{tr}(\mathbf{A}) + \operatorname{tr}(\mathbf{B}) - n$.

A.3 Proof for Proposition 3.3

Because of the linearity of functions in $\mathcal{H}$, $f(\mathbf{x}+\Delta x)-f(\mathbf{x})=f(\Delta x)$. Thanks to the reproducing property in Equation (3) and the Cauchy-Schwarz inequality,

$$\begin{aligned}\langle f(\Delta x),\mathbf{c}\rangle &= \langle f, K(\cdot,\Delta x)\mathbf{c}\rangle_{\mathcal{H}}\\ &\le \|f\|_{\mathcal{H}}\cdot\|K(\cdot,\Delta x)\mathbf{c}\|_{\mathcal{H}}\\ &= \|f\|_{\mathcal{H}}\sqrt{\mathbf{c}^\top K(\Delta x,\Delta x)\mathbf{c}}\\ &= \|f\|_{\mathcal{H}}\cdot\sqrt{\Delta\mathbf{x}^\top\mathbf{K}_{\mathrm{E}}\Delta\mathbf{x}}\cdot\sqrt{\mathbf{c}^\top\mathbf{K}_{\mathrm{O}}\mathbf{c}}.\end{aligned} \tag{32}$$

Let $\mathbf{c}=\mathbf{K}_{\mathrm{O}}^{-1}f(\Delta\mathbf{x})$, and then we have

$$\|f(\Delta\mathbf{x})\|^2_{\mathbf{K}_{\mathrm{O}}^{-1}}\le\|f\|_{\mathcal{H}}\cdot\|\Delta\mathbf{x}\|_{\mathbf{K}_{\mathrm{E}}}\cdot\|f(\Delta\mathbf{x})\|_{\mathbf{K}_{\mathrm{O}}^{-1}}. \tag{33}$$

Thus, $\|f(\mathbf{x}+\Delta x)-f(\mathbf{x})\|_{\mathbf{K}_{\mathrm{O}}^{-1}}=\|f(\Delta\mathbf{x})\|_{\mathbf{K}_{\mathrm{O}}^{-1}}\le\|f\|_{\mathcal{H}}\cdot\|\Delta\mathbf{x}\|_{\mathbf{K}_{\mathrm{E}}}$.

A.4 Proof for Proposition 3.4

The matrix $\mathbf{I}+\gamma\mathbf{L}_{\mathcal{G}}$ is an *M-matrix*, because it is strictly diagonally dominant with non-positive off-diagonal entries. Since the inverse of M-matrices are component-wise non-negative, so is $(\mathbf{I}+\gamma\mathbf{L}_{\mathcal{G}})^{-1}$. Because Laplacian matrices always have a zero eigenvalue corresponding to all-ones eigenvector $\mathbf{1}$,

$$(\mathbf{I}+\gamma\mathbf{L}_{\mathcal{G}})\cdot\mathbf{1}=\mathbf{1}+\gamma\cdot 0=\mathbf{1}\Rightarrow(\mathbf{I}+\gamma\mathbf{L}_{\mathcal{G}})^{-1}\cdot\mathbf{1}=\mathbf{1}.$$

Thus, each row (and hence each column by symmetry) of $(\mathbf{I}+\gamma\mathbf{L}_{\mathcal{G}})^{-1}$ sums to one.

A.5 Proof for Proposition 3.5

Since $\mathbf{K}_{\mathrm{O}}$ and $\mathbf{K}_{\mathrm{E}}$ are non-singular matrices, it is valid to change variables. Plugging $\Theta=(\mathbf{K}_{\mathrm{O}}^{1/2}\Phi_2)\cdot(\mathbf{K}_{\mathrm{E}}^{1/2}\Phi_1)^\top=\mathbf{K}_{\mathrm{O}}^{1/2}\Phi\mathbf{K}_{\mathrm{E}}^{1/2}$ into Equation (27),

$$\begin{aligned}\|f\|_{\mathcal{H}} &= \sqrt{\mathrm{tr}\left(\mathbf{K}_{\mathrm{E}}^{-1}\Theta\mathbf{K}_{\mathrm{O}}^{-1}\Theta^\top\right)}\\ &= \sqrt{\mathrm{tr}\left(\mathbf{K}_{\mathrm{E}}^{-1}\mathbf{K}_{\mathrm{E}}^{1/2}\Phi\left(\mathbf{K}_{\mathrm{O}}^{1/2}\mathbf{K}_{\mathrm{O}}^{-1}\mathbf{K}_{\mathrm{O}}^{1/2}\right)\Phi^\top\mathbf{K}_{\mathrm{E}}^{1/2}\right)}\\ &= \sqrt{\mathrm{tr}(\Phi\Phi^\top)}=\|\Phi\|_F.\end{aligned} \tag{34}$$

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] Mauricio A. Alvarez, Lorenzo Rosasco, Neil D. Lawrence, et al. 2012. Kernels for vector-valued functions: A review. *Found. Trends Mach. Learn.* 4, 3 (2012), 195–266.

[2] Onureena Banerjee, Laurent El Ghaoui, and Alexandre d'Aspremont. 2008. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *J. Mach. Learn. Res.* 9 (Mar. 2008), 485–516.

[3] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261.*

[4] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263.*

[5] Chris M. Bishop. 1995. Training with noise is equivalent to Tikhonov regularization. *Neural Comput.* 7, 1 (1995), 108–116.

[6] O. Celma. 2010. *Music Recommendation and Discovery in the Long Tail.* Springer.

[7] Chih-Ming Chen, Chuan-Ju Wang, Ming-Feng Tsai, and Yi-Hsuan Yang. 2019. Collaborative similarity embedding for recommender systems. In *Proceedings of the World Wide Web Conference (WWW'19).* ACM, New York, NY, 2637–2643.

[8] Zhiyong Cheng, Jialie Shen, Lei Zhu, Mohan S. Kankanhalli, and Liqiang Nie. 2017. Exploiting music play sequence for music recommendation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*, Vol. 17. 3654–3660.

[9] 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)*. ACM, 191–198.

[10] H. E. Egilmez, E. Pavez, and A. Ortega. 2017. Graph learning from data under laplacian and structural constraints. *IEEE J. Select. Top. Signal Process.* 11, 6 (Sep. 2017), 825–841.

[11] 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.

[12] F. Maxwell Harper and Joseph A. Konstan. 2016. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.* 5, 4 (2016), 19.

[13] Ruining He, Chen Fang, Zhaowen Wang, and Julian McAuley. 2016. Vista: A visually, socially, and temporally aware model for artistic recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems (Recsys'16)*. ACM, 309–316.

[14] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *Proceedings of the 11th ACM Conference on Recommender Systems*. ACM, 161–169.

[15] 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)*. International World Wide Web Conferences Steering Committee, 173–182.

[16] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, New York, NY, 843–852.

[17] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *Proceedings of 4th International Conference on Learning Representation (ICLR'16)*.

[18] Liang Hu, Longbing Cao, Jian Cao, Zhiping Gu, Guandong Xu, and Jie Wang. 2017. Improving the quality of recommendations for users and items in the tail of distribution. *ACM Trans. Info. Syst.* 35, 3 (2017), 1–37.

[19] Yogesh Jhamb, Travis Ebesu, and Yi Fang. 2018. Attentive contextual denoising autoencoder for recommendation. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval (SIGIR'18)*. ACM, 27–34.

[20] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'18)*. IEEE, 197–206.

[21] George S Kimeldorf and Grace Wahba. 1970. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *Ann. Math. Stat.* 41, 2 (1970), 495–502.

[22] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations (ICLR'17)*.

[23] Lingpeng Kong, Gabor Melis, Wang Ling, Lei Yu, and Dani Yogatama. 2019. Variational smoothing in recurrent neural network language models. In *Proceedings of the International Conference on Learning Representations (ICLR'19)*.

[24] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[25] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the ACM on Conference on Information and Knowledge Management (CIKM'17)*. ACM, New York, NY, 1419–1428.

[26] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M. Blei. 2016. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th ACM Conference on Recommender Systems (Recsys'16)*. ACM, 59–66.

[27] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the World Wide Web Conference on World Wide Web (WWW'18)*. International World Wide Web Conferences Steering Committee, 689–698.

[28] Daryl Lim, Julian McAuley, and Gert Lanckriet. 2015. Top-n recommendation with missing implicit feedback. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys'15)*. ACM, 309–312.

[29] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Comput.* 1 (2003), 76–80.

[30] Hao Ma, Irwin King, and Michael R. Lyu. 2007. Effective missing data prediction for collaborative filtering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07)*. 39–46.

[31] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining (WSDM'11)*. ACM, 287–296.

[32] Benjamin M. Marlin. 2004. Modeling user rating profiles for collaborative filtering. In *Advances in Neural Information Processing Systems (NIPS'04)*. MIT Press, 627–634.

[33] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'15)*. ACM, New York, NY, 43–52.

[34] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS'13)*. MIT Press, 3111–3119.

[35] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N. Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM'08)*. IEEE, 502–511.

[36] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14)*. ACM, 701–710.

[37] Nikhil Rao, Hsiang-Fu Yu, Pradeep K. Ravikumar, and Inderjit S. Dhillon. 2015. Collaborative filtering with graph information: Consistency and scalable methods. In *Advances in Neural Information Processing Systems (NIPS'15)*. MIT Press, 2107–2115.

[38] Chuan Shi, Binbin Hu, Xin Zhao, and Philip Yu. 2018. Heterogeneous information network embedding for recommendation. *IEEE Trans. Knowl. Data Eng.* 31, 2 (2018), 357–370.

[39] Chuan Shi, Zhiqiang Zhang, Ping Luo, Philip S. Yu, Yading Yue, and Bin Wu. 2015. Semantic path-based personalized recommendation on weighted heterogeneous information networks. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM'15)*. ACM, 453–462.

[40] Alexander J. Smola and Risi Kondor. 2003. Kernels and regularization on graphs. In *Learning Theory and Kernel Machines*. Springer, 144–158.

[41] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM'19)*. ACM, New York, NY.

[42] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proc. VLDB Endow.* 4, 11 (2011), 992–1003.

[43] Jiaxi Tang and Ke Wang. 2018. Personalized top-N sequential recommendation via convolutional sequence embedding. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM'18)*. ACM, New York, NY, 565–573.

[44] Thanh Tran, Kyumin Lee, Yiming Liao, and Dongwon Lee. 2018. Regularizing matrix factorization with user and item embeddings for recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM'18)*. ACM, 687–696.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS'17)*. MIT Press, 5998–6008.

[46] Saurabh Verma and Zhi-Li Zhang. 2019. Stability and generalization of graph convolutional neural networks. In *Proceedings of the 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD'19)*.

[47] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripple network: Propagating user preferences on the knowledge graph for recommender systems. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM'18)*.

[48] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Exploring high-order user preference on the knowledge graph for recommender systems. *ACM Trans. Info. Syst.* 37, 3 (2019), 1–26.

[49] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep knowledge-aware network for news recommendation. In *Proceedings of the World Wide Web Conference on World Wide Web (WWW'18)*. International World Wide Web Conferences Steering Committee, 1835–1844.

[50] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-task feature learning for knowledge graph enhanced recommendation. In *Proceedings of the World Wide Web Conference (WWW'19)*. ACM, New York, NY, 2000–2010.

[51] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge graph convolutional networks for recommender systems. In *Proceedings of the World Wide Web Conference (WWW'19)*. ACM, 3307–3313.

[52] 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)*.

[53] Zihan Wang, Ziheng Jiang, Zhaochun Ren, Jiliang Tang, and Dawei Yin. 2018. A path-constrained framework for discriminating substitutable and complementary products in e-commerce. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM'18)*. ACM, 619–627.

[54] Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19)*. Association for Computational Linguistics, Hong Kong, China, 6383–6389.

[55] Christopher K. I. Williams and Carl Edward Rasmussen. 2006. *Gaussian Processes for Machine Learning*. Vol. 2. MIT Press, Cambridge, MA.

[56] 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 (ICML'19)*. PMLR, 6861–6871.

[57] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 346–353.

[58] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining (WSDM'16)*. ACM, 153–162.

[59] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*.

[60] Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models. In *Proceedings of the International Conference on Learning Representations (ICLR'17)*.

[61] Xin Xin, Xiangnan He, Yongfeng Zhang, Yongdong Zhang, and Joemon Jose. 2019. Relational collaborative filtering: Modeling multiple item relations for recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*.

[62] Feng Xue, Xiangnan He, Xiang Wang, Jiandong Xu, Kai Liu, and Richang Hong. 2019. Deep item-based collaborative filtering for top-n recommendation. *ACM Trans. Info. Syst.* 37, 3 (2019), 1–25.

[63] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. 2018. HOP-rec: High-order proximity for implicit recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems (Recsys'18)*. ACM, 140–144.

[64] 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)*. ACM, 974–983.

[65] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'16)*. ACM, New York, NY, 729–732.

[66] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM'14)*. ACM, 283–292.

[67] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining (WSDM'19)*. ACM, New York, NY, 582–590.

[68] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16)*. ACM, 353–362.

[69] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NIPS'15)*. MIT Press, 649–657.

[70] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W. Bruce Croft. 2017. Joint representation learning for top-n recommendation with heterogeneous information sources. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM'17)*. ACM, 1449–1458.

[71] Yuan Zhang, Xiaoran Xu, Hanning Zhou, and Yan Zhang. 2020. Distilling structured knowledge into embeddings for explainable and accurate recommendation. In *Proceedings of the 13th ACM International Conference on Web Search and Data Mining (WSDM'2020)*.

[72] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-graph-based recommendation fusion over heterogeneous information networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'17)*. ACM, 635–644.

[73] 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)*. ACM, New York, NY, 311–319.

[74] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning tree-based deep model for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'18)*. ACM, 1079–1088.

Received January 2020; revised June 2020; accepted July 2020