Enhanced Graph Learning for Collaborative Filtering via Mutual Information Maximization

Yonghui Yang[1,2], Le Wu[1,2,3,*], Richang Hong[1,2], Kun Zhang[1,2], Meng Wang[1,2,3]
[1] Key Laboratory of Knowledge Engineering with Big Data, Hefei University of Technology, China
[2] School of Computer Science and Information Engineering, Hefei University of Technology, China
[3] Institute of Artificial Intelligence, Hefei Comprehensive National Science Center, China
{yyh.hfut,lewu.ustc,hongrc.hfut,zhang1028kun,eric.mengwang}@gmail.com

ABSTRACT

Neural graph based Collaborative Filtering (CF) models learn user and item embeddings based on the user-item bipartite graph structure, and have achieved state-of-the-art recommendation performance. In the ubiquitous implicit feedback based CF, users' unobserved behaviors are treated as unlinked edges in the user-item bipartite graph. As users' unobserved behaviors are mixed with dislikes and unknown positive preferences, the fixed graph structure input is missing with potential positive preference links. In this paper, we study how to better learn enhanced graph structure for CF. We argue that node embedding learning and graph structure learning can mutually enhance each other in CF, as updated node embeddings are learned from previous graph structure, and vice versa (i.e., newly updated graph structure are optimized based on current node embedding results). Some previous works provided approaches to refine the graph structure. However, most of these graph learning models relied on node features for modeling, which are not available in CF. Besides, nearly all optimization goals tried to compare the learned adaptive graph and the original graph from a local reconstruction perspective, whether the global properties of the adaptive graph structure are modeled in the learning process is still unknown. To this end, in this paper, we propose an enhanced graph learning network (*EGLN*) approach for CF via mutual information maximization. The key idea of *EGLN* is two folds: First, we let the enhanced graph learning module and the node embedding module iteratively learn from each other without any feature input. Second, we design a local-global consistency optimization function to capture the global properties in the enhanced graph learning process. Finally, extensive experimental results on three real-world datasets clearly show the effectiveness of our proposed model.

CCS CONCEPTS

• **Information systems → Recommender systems**.

Le Wu is the corresponding author.

KEYWORDS

Collaborative Filtering, Recommendation, Graph Learning, Mutual Information Maximization

ACM Reference Format:
Yonghui Yang[1,2], Le Wu[1,2,3,*], Richang Hong[1,2], Kun Zhang[1,2], Meng Wang[1,2,3]. 2021. Enhanced Graph Learning for Collaborative Filtering via Mutual Information Maximization . In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3462928

1 INTRODUCTION

Collaborative filtering provides personalized recommendations by collectively learning users' preference from user-item historical interaction behaviors [22, 26, 28, 34]. In most recommendation scenarios, it is more common for users to express their preferences with implicit feedback (e.g., viewing a movie, visiting a restaurant, and pining a picture) rather than explicit ratings. Under the ubiquity of implicit feedback, users' limited observed behaviors are denoted as positive preference set, while negative and unobserved positive preferences are mixed together to form a large unobserved preference set [15, 24, 26]. Therefore, how to learn users' preferences from implicit feedback based CF has become an important topic in both academia and industry.

State-of-the-art CF models rely on embedding techniques for the recommendation, due to the flexibility, and relatively high performance of these embedding models [4, 13, 26, 33]. Earlier works treated user-item implicit feedback as a user-item binary rating matrix, with the observed values as 1, and the unobserved values as 0, and adopted matrix factorization techniques for the user and item embedding learning [13, 26]. Bayesian Personalized Ranking (BPR) is a popular pairwise ranking matrix factorization based recommendation model that is specifically designed for implicit feedback. By projecting both users and items into a low latent space, BPR designs a pairwise ranking goal, assuming that a user prefers an observed item to an unobserved item [26]. As users' behaviors are naturally represented as a user-item bipartite graph, neural graph collaborative filtering models have been proposed to better model user-item bipartite graph structure [4, 12, 33]. These neural graph models perform graph convolution by updating user and item embeddings at current layer based on the aggregation of linked entity embeddings at previous layer. Furthermore, motivated by the advances of neural mutual information techniques for capturing structured information [32], researchers propose to learn node embeddings of graph-structured data by maximizing the mutual information between node representations and the corresponding summarized global representations of the graph structure [3, 32]. These neural graph models inject the graph structure in user and item embedding learning, and enhance representation learning with mutual information maximization on graph-structured data.

In fact, the performance of neural graph recommendation models relies on the input data of the user-item bipartite graph. In implicit feedback based recommendation, directly transforming the observed user-item behaviors as links is the default choice for current neural graph models. However, we argue that the fixed graph construction process neglects the unique properties of the implicit feedback recommendation by treating all unobserved behaviors as negative feedback. In practice, unobserved behaviors are mixed with negative feedback and unknown positive feedback. Simply treating all unobserved feedback as negative edges that are not linked in the user-item bipartite graph neglects the difference between the true negative and false negative behaviors. Such a default fixed graph structure is obviously noisy with missing false negative interactions, and would lead to suboptimal performance especially when users have sparse interaction records.

In this paper, we argue that instead of adopting a fixed user-item graph structure for embedding learning in CF, we also need to learn an adaptive user-item graph structure to better serve CF. This is a non-trivial task as we do not know whether a missing link from users' unobserved behaviors is true negative or false negative. Some researchers in the machine learning community also study a similar problem of learning better graph structures to facilitate downstream tasks. Researchers propose to learn graph topology and node embeddings from a unified perspective, either by edge reweighting [31] with self-attention or parameterized node similarity with input node features [19]. These models do not suit our task for two reasons. First, all of these models rely on the input node features for adaptive graph learning, while CF does not contain any node feature information. Second, most of these adaptive graph learning techniques use local optimization functions for graph learning, e.g., the learned node embeddings from adaptive graph need to reconstruct the original graph with first-order reconstruction errors. However, as the graph local-global structure matters, how to keep the local-global consistency of the adaptively learned graph is still unknown.

To this end, in this paper, we propose an enhanced graph learning approach for CF. We formulate the approach with two mutually enhanced parts: *enhanced graph learning module* and *node embedding learning module*. The rationale is that, we can design a better graph learning module with previously learned user and item embeddings, and a better user and item embedding learning with updated adaptive graph structure. Specifically, we first let the graph learning module and the node embedding learning module iteratively learn from each other without any feature input. As users' observed behaviors are ground-truth positive preferences, and the missing behaviors are mixed with false negative (i.e., unknown positive) preferences, the graph learning module is a residual graph learning to pick possible unknown preferences with confidence weights. Next, we design an enhanced graph optimization function, with the local-global consistency of the enhanced graph is also modeled to serve better graph structure and node embedding learning. The proposed model can be trained end-to-end. Finally, we perform extensive experimental results on three real-world datasets, and the experimental results clearly demonstrate the superiority and effectiveness of our proposed model.

2 RELATED WORKS

Collaborative Filtering. CF based approaches make personalized recommendations by collectively analyzing user-item behaviors [15, 22, 26]. Learning accurate user and item embeddings is the default architecture for modern recommender systems due to its flexibility and relatively high performance [28]. After that, the preference of a user to an item can be predicted as the inner product of the corresponding user and item embeddings. Most traditional CF models take a user-item matrix as input, and learn free user and item latent embeddings based on matrix factorization techniques [22, 26]. Recently, researchers argued that user behaviors can be naturally represented as a user-item bipartite graph structure. Thanks to the powerful modeling ability of graph neural networks for modeling complex graph structure [18], researchers proposed to learn user and item embeddings from the bipartite graph. Many graph based recommendation models, such as GC-MC [30], PinSage [38], NGCF [33], LR-GCCF [4] and LightGCN [12] have been proposed. Different neural graph recommendation models vary in the input graph data and the graph aggregation process. E.g., NGCF exploited the user-item bipartite graph structure as input, and injected the high-order collaborative signals for better user/item embedding learning with iterative graph convolutions[33]. PinSage took item-item correlation graph with item content features as input, and learned graph-smoothed item embeddings [38]. LR-GCCF [4] and LightGCN [12] simplified neural graph models with linear graph convolutions by considering the uniqueness of CF, and achieved state-of-the-art performance. Neural graph based recommendation models have shown dominating recommendation performance. Nearly all neural graph recommendation models are implemented with a fixed sparse user-item graph. In practice, the user/item embeddings learned from the input graph are predictable for the possible missing links of the original user-item graph. Based on this natural idea, instead of learning embeddings from fixed user-item graph at once, we consider how to iteratively perform graph learning based on current learned embeddings to enhance user-item graph for better recommendation.

Graph Learning based Models. Graph convolutional networks perform graph convolutions to integrate vertex features and the input graph topology for node embedding learning [18, 19]. Here, the graph structure is constructed by human experts or precomputed by the k-nearest neighbors of each node [1, 6]. As the input graph structure is not perfect and may contain missing or noisy edges, the graph structure is not optimized for graph based downstream tasks. GAT is proposed to attentively reweight edge importance based on the self-attention mechanism [31]. Researchers proposed to learn adaptive graph structure to facilitate downstream graph based tasks [16, 20, 39]. These approaches learned distance metrics for graph Laplacian matrix with parameterized similarity calculation. The parameterized similarity calculation is based on input node features, such as the covariance matrix of input features [20], feature weight vector [16]. E.g., given input features and a preliminary graph structure, GLCN is proposed to learn revised edge weights with weight vectors of input features [16]. Most of the above models only perform graph learning at once. As better graph structure can help better node embedding learning, and vice versa, an iterative deep graph learning model is proposed to let graph learners and downstream tasks enhance each other [5]. Since the input graph is not always available in real-world scenarios, researchers proposed to learn discrete and spare graph structure from scratch by solving a bilevel program that learns a discrete probability on the edges of the graph [7]. Since this model needs to learn the possible link of every node pair, the time complexity is square of the nodes, which prevents it from being applied to large graphs. In summary, all of these works showed promising results of learning and revising graph structure to facilitate downstream tasks. However, these models can not directly applied to CF as we do not have any node features. Inspired by the dropout technique in deep learning, researchers proposed to drop edges for robust graph learning, such to avoid fake edges or adversarial attacks [17, 27]. In contrast, we consider the scenario of sparse links in CF and how to add edges to enhance neural graph recommendation models.

Mutual Information Maximization. Maximizing the mutual information between the input and output is a fundamental quantity for measuring the dependency of input and output variables. Recently, many researchers paid attention to representation learning based on mutual information estimation [2, 14, 32]. Due to the difficulty of measuring mutual information in high dimensional space, Belghazi et al. proposed a neural estimation approach for mutual information estimation, where a statistical network is trained to distinguish samples coming from the joint distribution of two random variables. Deep InfoMax (DIM) argued that the structure matters, and investigated representation learning by maximizing the mutual information between local representation and global representation [14]. DIM significantly improved the representation learning performance via the local-global pair mutual information maximization. Inspired by DIM, Deep Graph InfoMax (DGI) is proposed for representation learning for graph structured data [32]. DGI maximizes mutual information between patch representations and corresponding high level summary of graph, which are learned based on GNNs. DGI is further extended to the heterogeneous graph embedding [23], community-aware graph embedding learning [40], consideration of measuring mutual information from both node features and topological structure [25], and graph-level representation learning [29]. In CF based recommendation, users and items are formed as a bipartite graph. Cao et al. proposed BiGI, a bipartite graph embedding learning method to suit recommendation scenario via mutual information maximization [3]. Specifically, BiGI first generated global graph representation from the composition of user/item embedding and local representation via a subgraph-level attention mechanism. Then, it recognized global properties through local-global mutual information maximization. BiGI showed superior recommendation performance compared to neural graph based recommendation models by modeling global properties of the graph. Our work also borrows the success of mutual information modeling on graph structure data. Different from these works, we apply mutual information on the enhanced graph to model the global graph properties for the enhanced graph learning.

3 THE PROPOSED MODEL

In this section, we would introduce the proposed *Enhanced Graph Learning Network (EGLN)* approach for recommendation. We first propose the overall architecture of the proposed model, followed by the two key components in our proposed model: enhanced graph learning with learned embeddings and node embedding learning with the enhanced graph structure.

3.1 Overall Architecture

In CF based recommender systems, there are two kinds of entries: a userset U ($|U| = M$) and an itemset V ($|V| = N$). Considering the implicit feedback is more common in most recommendation scenarios, we use interaction matrix $\mathbf{R} \in \mathbb{R}^{M\times N}$ to denote user-item interactions, where each element $\mathbf{r}_{ai} = 1$ if user a has interacted with item i, otherwise $\mathbf{r}_{ai} = 0$. Given the interaction matrix $\mathbf{R}$, most neural graph recommendation models define a user-item bipartite graph as: $\mathcal{G} = \{U \cup V, \mathbf{A}\}$, where the adjacency matrix is unweighted matrix defined as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}^{M\times M} & \mathbf{R} \\ \mathbf{R}^T & \mathbf{0}^{N\times N} \end{bmatrix}. \tag{1}$$

Given the above graph $\mathcal{G} = \{U \cup V, \mathbf{A}\}$, most graph based CF models take this fixed graph as input, and learn user and item embeddings based on the fixed graph structure. These models use iterative graph aggregation operations to encode the graph structure for node embedding learning. Let $\mathbf{P} \in \mathbb{R}^{D\times M}$ and $\mathbf{Q} \in \mathbb{R}^{D\times N}$ denote the free user embedding matrix and free item embedding matrix that need to be learned, the state-of-the-art graph based CF models learn the final user embedding matrix $\mathbf{U} \in \mathbb{R}^{D\times M}$ and final item embedding matrix $\mathbf{V} \in \mathbb{R}^{D\times N}$ by injecting high-order structure of the graph $\mathcal{G}$ for the final embedding learning. Without loss of generality, we use $[\mathbf{P}, \mathbf{Q}, \mathbf{U}, \mathbf{V}] = GCF(\mathcal{G})$ to represent the graph based CF approach. We use node to denote either a user or an item node without distinction.

Previous graph based recommendation models use fixed graph structure input $\mathcal{G}$ for user and item embedding learning. In this work, we argue that the fixed graph structure is not the optimal input for graph based CF models. The reason is that, in implicit feedback based CF, users' unobserved behaviors are mixed with negative and unknown positive preferences. Nevertheless, the fixed graph structure treats the negative and unknown preferences equally by regarding all unknown preferences as missing links on the fixed graph structure. Therefore, we argue instead of adopting a fixed user-item graph structure for embedding learning in CF, we also need to learn an enhanced user-item graph structure to better serve CF. We denote the enhanced graph structure as $\mathcal{G}^E = \{U \cup V, \mathbf{A}^E\}$, where $\mathbf{A}^E = \mathbf{A} + \mathbf{A}^R$. Specifically, $\mathbf{A}^R \in \mathbb{R}^{(M+N)\times(M+N)}$ denotes the residual non-negative edge weight matrix that needs to be learned. We use the residual graph learning structure as all existing edges in the original user-item bipartite graph denote the positive preferences of users, and are valuable for user and item embedding learning. By using the residual graph structure, the revised graph structure is enhanced by possible unknown positive preferences that are hidden in unobserved behaviors.

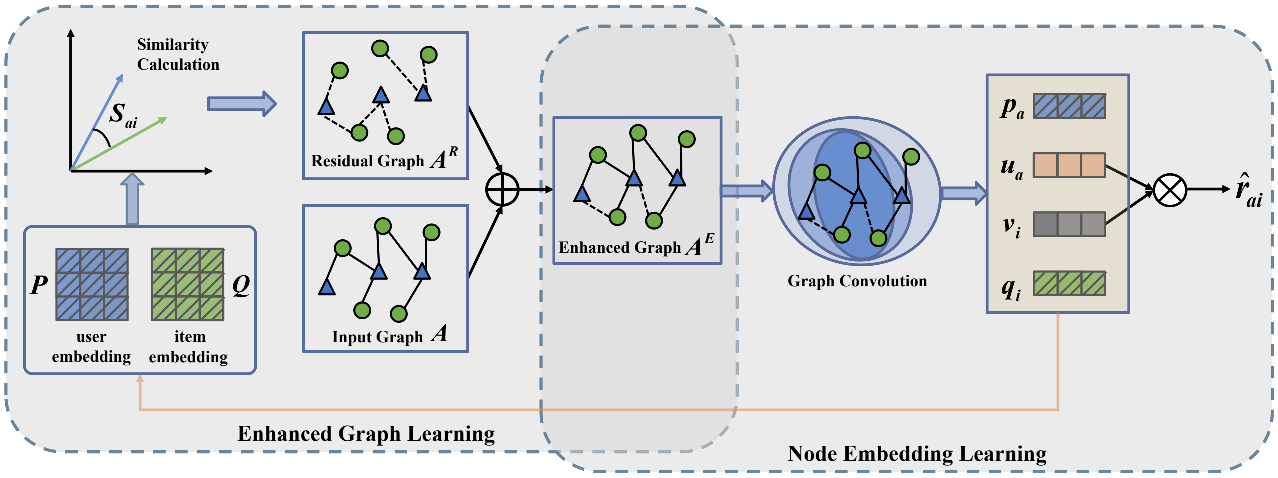


Figure 1: The overall framework of our proposed model.

Our goal turns to find a better residual graph with edge weight matrix $\mathbf{A}^R$, such that we could better serve user and item embedding learning for improving CF performance. Now, the graph based CF consists of two modules: residual graph learning and node embedding learning. These two modules are not isolated but closely related. On one hand, the residual graph learning module needs to rely on the currently learned user and item embeddings, in order to find possible links of $\mathbf{A}^E$. As we can predict users' preferences based on the learned user and item embeddings, the possible unknown residual link matrix $\mathbf{A}^R$ can be formulated as: $\mathbf{A}^R = GL(\mathbf{P}, \mathbf{Q}, \mathbf{U}, \mathbf{V})$, where GL is the graph learning module based on the output of the graph based CF module. On the other hand, after we have obtained residual graph structure, we could use the graph based CF module to find better user and item embeddings as: $[\mathbf{P}, \mathbf{Q}, \mathbf{U}, \mathbf{V}] = GCF(\mathbf{A} + \mathbf{A}^R)$ based on the enhanced graph structure. Therefore, these two modules can be performed in an iterative way, with better graph based CF module facilitates the better graph learning module, and vice versa. For better illustration, we show the overall framework of *EGLN* in Figure 1 and give the detailed structure of two modules in the following subsections.

3.2 Enhanced Graph Learning with Learned Embeddings

Given the node embeddings learned from the previous graph based CF model as: $[\mathbf{P}, \mathbf{Q}, \mathbf{U}, \mathbf{V}] = GCF(\mathbf{A} + \mathbf{A}^R)$, our goal is to learn a residual graph structure with residual weight matrix $\mathbf{A}^R$. Since the residual graph structure is also a user-item bipartite graph, the weight matrix $\mathbf{A}^R$ is actually composed as:

$$\mathbf{A}^R = \begin{bmatrix} \mathbf{0}^{M\times M} & \mathbf{S} \\ \mathbf{S}^T & \mathbf{0}^{N\times N} \end{bmatrix}, \tag{2}$$

where $\mathbf{S} \in \mathbb{R}^{M\times N}$ is the residual user-item preference matrix that needs to be learned. We calculate the user-item similarity matrix $\mathbf{S}$ as:

$$s_{ai} = \sigma\left(\frac{< \mathbf{p}_a \times \mathbf{W}_1, \mathbf{q}_i \times \mathbf{W}_2 >}{|\mathbf{p}_a \times \mathbf{W}_1||\mathbf{q}_i \times \mathbf{W}_2|}\right), \tag{3}$$

where $\sigma(x)$ is a sigmoid function that transforms the computed similarity into range $(0, 1)$, $\mathbf{W}_1$ and $\mathbf{W}_2$ are two trainable matrices that map user/item representations from free latent space into similarity space, and $<,>$ denotes vector inner product operation.

From Eq. (3), we have the learned user-item similarity matrix. However, the learned similarity matrix $\mathbf{S}$ is dense and hard to be used for graph convolution. Similar to many graph construction models [16, 39], we sparse the learned similarity matrix for residual graph construction. Generally, there are two kinds of methods for graph sparseness: *L1 normalization* [16, 17] and *threshold interception* [39]. In this paper, we use threshold interception as it can flexibly control the edges of the learned graph. In practice, for each user, we retain the edges with top-K computed similarities. The sparsified similarity matrix is calculated as follows:

$$s_{ai} = \begin{cases} s_{ai}, & s_{ai} \in topK(s_a) \\ 0, & s_{ai} \notin topK(s_a), \end{cases} \tag{4}$$

where $s_a = [s_{a1}, ..., s_{aN}]$ is the learned similarity vector of each user based on Eq.(3).

There may exist a question that why we use the free user and item embedding matrices (i.e., $\mathbf{P}$ and $\mathbf{Q}$) instead of the final user and item embedding matrices (i.e.,$\mathbf{U}$ and $\mathbf{V}$) for residual graph weight matrix learning. In practice, we also try to use the final embedding matrices for residual graph learning, but it does not perform as well as Eq.(3). We guess a possible reason is that, the final embedding matrices are injected into the node-centric high-order graph structure. By using the final node embeddings, most of the non-zero $\mathbf{s}_{ij}$ values are from the users (items) that already have links in the original graph. If we exclude the original fixed graph structure for finding the possibly unlinked user-item pairs, most of these user-item pairs are close in the graph structure and can already be modeled with graph CF models. In contrast, the free embeddings capture the original characteristics of nodes for updated graph learning, and are complementary to the final node embeddings that already modeled the current graph structure.

Please note that the edge weights $\mathbf{A}$ in the original graph are equal to one but vary in the enhanced graph with weight matrix $\mathbf{A}^E$. The reason is that the learned residual graph has two kinds of edges: one is the old edges that already appear in the original graph and another is the newly added edges compared to the original graph. Thus, in the enhanced graph with residual graph structure, old edges are reweighted with weight value larger than 1 while new edges are weighted with value less than 1. This indicates that our enhanced graph could be able to add missing edges and reweight the existing edges simultaneously.

3.3 Embedding Learning with Enhanced Graph Structure

Given the enhanced graph structure with weight matrix $\mathbf{A}^E$, the embedding learning module tries to learn better user and item embedding with enhanced graph structure as: $[\mathbf{P}, \mathbf{Q}, \mathbf{U}, \mathbf{V}] = GCF(\mathbf{A}^E)$. In the following, we introduce the architecture of the node embedding learning module.

GCNs are state-of-the-art techniques in representation learning, which encode local graph structure into node representation [18]. Extensive works show the effectiveness of GCNs on graph based recommendations [33, 35, 36, 38]. Therefore, we also use GCN as encoder and feed the enhanced graph $\mathcal{G}_E$ into the encoder to generate node representations.

Generally, given the initial user/item embedding matrix $\mathbf{P}^0 = \mathbf{P}, \mathbf{Q}^0 = \mathbf{Q}$,the enhanced graph weight matrix $\mathbf{A}^E$ and a pre-defined embedding propagation depth K, graph convolution encoder outputs the final user embedding matrix $\mathbf{U}$ and final item embedding matrix $\mathbf{V}$. At (k+1)-th layer, each user's and each item's embeddings are updated with the aggregation of neighbors' embeddings as:

$$\mathbf{p}_a^{k+1} = AGG(\mathbf{p}_a^k, \{\mathbf{q}_j^k : j \in \mathbf{A}_a^E\}) \tag{5}$$

$$\mathbf{q}_i^{k+1} = AGG(\mathbf{q}_i^k, \{\mathbf{p}_b^k : b \in \mathbf{A}_{(M+i)}^E\}) \tag{6}$$

where $A_a^E = \{j|A_{aj}^E > 0\} \subseteq V$ is the item set that user a links with, and $A_{M+i}^E = \{b|A_{(M+i)b}^E > 0\} \subseteq U$ is the user set who linked with item i. $AGG()$ denotes the aggregation operation, which can be implemented with many optional functions, such as concatenation, weighted sum and neural network based aggregation. Traditional GCN methods usually adopt non-linear activation and trainable weight transformation on feature propagation [9, 18, 30]. However, recent works have demonstrated that non-linear activation and feature transformation are unnecessary and bring unnecessary complexity in graph based CF [4, 12]. Therefore, we only use the key component: neighbor aggregation in graph convolution encoder. Given these findings, we implement Eq.(7) and Eq.(8) as follows:

$$\mathbf{p}_a^{k+1} = \mathbf{p}_a^k + \frac{1}{\sum_{j=0}^{M+N-1} \mathbf{A}_{aj}^E} \sum_{j \in \mathbf{A}_a^E} \mathbf{A}_{aj}^E \mathbf{q}_j^k, \tag{7}$$

$$\mathbf{q}_i^{k+1} = \mathbf{q}_i^k + \frac{1}{\sum_{b=0}^{M+N-1} \mathbf{A}_{(M+i)b}^E} \sum_{b \in \mathbf{A}_{M+i}^E} \mathbf{A}_{(M+i)b}^E \mathbf{p}_b^k. \tag{8}$$

Given the pre-defined embedding propagation depth K, we could obtain the final user embedding matrix $\mathbf{U} = \mathbf{P}^K$ and final item embedding matrix $\mathbf{V} = \mathbf{Q}^K$. After that, the preference of user a to item i can be predicted as follows:

$$\hat{\mathbf{r}}_{\mathbf{ai}} =< \mathbf{u}_a, \mathbf{v}_i >, \tag{9}$$

where $<,>$ denotes vector inner product operation. In order to illustrate the embedding propagation process more clearly and facilitate the batch implementation, we formulate Eq.(7) and Eq.(8) in a matrix form. Let matrix $\mathbf{P}^k$ and matrix $\mathbf{Q}^k$ denote the embedding matrices of users and items after k-th propagation, then the updated embedding matrices on (k+1)-th propagation are formulated as follows:

$$\begin{bmatrix} \mathbf{P}^{k+1} \\ \mathbf{Q}^{k+1} \end{bmatrix} = (\begin{bmatrix} \mathbf{P}^k \\ \mathbf{Q}^k \end{bmatrix} + \mathbf{D}^{-1}\mathbf{A}^E \times \begin{bmatrix} \mathbf{P}^k \\ \mathbf{Q}^k \end{bmatrix}), \tag{10}$$

where k=0,1,2,...,K-1 (K is the pre-defined propagation depth). D is the degree matrix of the enhanced graph with weight matrix $\mathbf{A}^E$.

4 OPTIMIZATION DESIGN AND MODEL TRAINING

In this part, we first introduce the overall optimization function design for enhanced graph learning with mutual information maximization. After that, we illustrate the model training process.

4.1 Mutual Information Maximization for Enhanced Graph Learning

4.1.1 Constraint on Edge Level. Given the learned residual graph structure $\mathbf{A}^R$, a natural idea is to use reconstruction based loss as the graph learning module optimization goal:

$$\arg\min_{\Theta_G} \mathcal{L}_s = ||\mathbf{A} - \mathbf{A}^R||_F^2 \tag{11}$$

where $\Theta_G = [\mathbf{P}, \mathbf{Q}, \mathbf{W_1}, \mathbf{W_2}]$ are the parameters in graph learning. The above Euclidean distance minimization constraint encourages the learned residual graph to be close to the original graph from the edgewise level. We treat the edges of the original graph as the ground truth positive feedback. Thus, the learned residual graph should retain these edges. However, this edgewise constraint only learns the individual link correlation but fail to capture the global graph properties. In the following, we introduce the global graph properties through local-global consistency learning.

4.1.2 Constraint on Graph Level. Given the enhanced graph $\mathcal{G}^E = \{U \cup V, \mathbf{A}^E\}$, for each edge (u_a, v_i), we summarize the sub-graph centered around the user-item pair as the local representations. In practice, we use the final user and item embeddings output by the node embedding learning module in Section 3.3 as the local representations:

$$\mathbf{h}_{ai} = [\sigma(\mathbf{u}_a), \sigma(\mathbf{v}_i)], \tag{12}$$

where σ is a sigmoid activation function. The combination method we adopted is concatenation. After obtaining the local representations, we aim to seek the global representations to capture the information of the entire graph. Similar to existing representation learning works [14, 29, 32], we obtain the global representations $\mathbf{g}$ by leveraging a readout function: $\mathbb{R}^{Z\times 2D} \rightarrow \mathbb{R}^{2D}$, where Z is the number of edges on the enhanced graph $\mathcal{G}^E$. The detailed process is shown as follows:

$$Z = \sum_{a=0}^{M-1} \sum_{i=M}^{M+N-1} sign(\mathbf{A}_{ai}^E), \tag{13}$$

$$\mathbf{g} = \sum_{[a,i] \in \mathcal{G}^E} \frac{\mathbf{h}_{ai}}{Z}, \tag{14}$$

where $sign(x)$ is logical function, which equals to 1 if x >0, equals to 0 if $x = 0$ and equals to -1 if x <0.

After obtaining the local representations and global representations, we then maximize the mutual information between them to keep the local-global consistency. Similar to many existing works about mutual information maximization [3, 14, 32], we employ a discriminator $\mathcal{D}$ to compute the score that assigned to a <local,global> pair with a bilinear mapping function:

$$\mathcal{D}(\mathbf{h}, \mathbf{g}) = \mathbf{h}^{\mathbf{T}}\mathbf{W}_{\mathbf{d}}\mathbf{g} \tag{15}$$

where $\mathbf{W}_{\mathbf{d}} \in \mathbb{R}^{2D\times 2D}$ is a transformation weight matrix. Without confusion, we use vector $\mathbf{h}$ to denote the local representations of any edge on the enhanced graph $\mathcal{G}^E$. Then, we combine the local representations $\mathbf{h}$ and the global representations $\mathbf{g}$ as positive sample $[\mathbf{h}, \mathbf{g}]$ for the discriminator $\mathcal{D}$.

In order to perform contrastive learning for the discriminator $\mathcal{D}$, we need negative samples for discriminator optimization. Let $\mathbf{F}$ denote node embedding matrix of graph $\mathcal{G}^E$, then $\mathcal{G}^E$ can be described as $(\mathbf{A}^E, \mathbf{F})$. We implement three kinds of negative samplings $[\tilde{\mathbf{h}}, \mathbf{g}]$ from data augment perspective.

Fake Edge. Given the enhanced graph $(\mathbf{A}^E, \mathbf{F})$, we randomly sample a fake edge (u_a, v_j), where $\mathbf{A}^E_{aj} = 0$. Then, the local representations $\tilde{\mathbf{h}}$ of fake edge, and the global representations $\mathbf{g}$ are combined as a negative sample.

Feature shuffling. It generates a corrupted graph $(\mathbf{A}^E, \tilde{\mathbf{F}})$ by randomly shuffling a certain percentage of features. Negative samples are achieved by pairing local representations $\tilde{\mathbf{h}}$ from $(\mathbf{A}^E, \tilde{\mathbf{F}})$ and global representations $\mathbf{g}$ from $(\mathbf{A}^E, \mathbf{F})$.

Structure perturbation. It generates a corrupted graph $(\tilde{\mathbf{A}}^E, \mathbf{F})$ by randomly adding or dropping certain ratio of edges. The local representations $\tilde{\mathbf{h}}$ from $(\tilde{\mathbf{A}}^E, \mathbf{F})$ combine the global representations $\mathbf{g}$ from $(\mathbf{A}^E, \mathbf{F})$ as negative samples for the discriminator $\mathcal{D}$.

After obtaining positive and negative samples, the positive samples are labeled by 1 and the negative samples are labeled by 0. The local-global consistency mutual information maximization tries to correctly discriminate positive and negative local-global pairs [3, 32]. Therefore, we use binary cross entropy loss as $\mathcal{L}_d$:

$$\arg\min_{\Theta_D} \mathcal{L}_d = -\sum_{z=0}^{Z-1} \frac{log\mathcal{D}(\mathbf{h}, \mathbf{g}) + (1 - log\mathcal{D}(\tilde{\mathbf{h}}, \mathbf{g}))}{Z} \tag{16}$$

where $\Theta_D = \mathbf{W}_d$ is the discriminator parameter.

4.2 Model Training

For rating prediction, we employ the pairwise ranking based BPR loss [24, 26], which assumes that the observed items' prediction values should be higher than those unobserved. The objective function can be formulated as follows:

$$\arg\min_{\Theta_G} \mathcal{L}_r = \sum_{a=0}^{M-1} \sum_{(i,j)\in D_a} -\mathbf{ln}\sigma(\hat{r}_{ai} - \hat{r}_{aj}) + \lambda||\mathbf{P}||^2 + \lambda||\mathbf{Q}||^2, \tag{17}$$

where $\sigma(x)$ is a sigmoid function, $\Theta_G = [\mathbf{P}, \mathbf{Q}, \mathbf{W_1}, \mathbf{W_2}]$ are graph learning parameters, λ is the regularization coefficient. $D_a = \{(i, j)|i \in R_a \wedge j \notin R_a\}$ denotes the pairwise training data for user a. R_a represents the item set that user a has interacted.

Given rating loss $\mathcal{L}_r$(Eq.(17)), the local graph learning based loss $\mathcal{L}_s$(Eq.(11)) and the global graph learning based loss $\mathcal{L}_d$(Eq.(16)), we combine them as the final optimization. We set two parameters α and β to balance these three losses, respectively. Finally, the overall objective function can be formulated as follows:

Table 1: The statistics of three datasets.

Dataset	Users	Items	Ratings	Sparsity
Movielens-1M	6040	3952	226310	99.052%
Amazon-Video Games	31,207	33,899	300003	99.972%
Pinterest	55187	9916	1500809	99.726%

$$\arg\min_{\Theta} \mathcal{L} = \mathcal{L}_r + \alpha\mathcal{L}_s + \beta\mathcal{L}_d, \tag{18}$$

where $\Theta = [\Theta_G, \Theta_D]$ are all parameters in the final objective function. We implement the proposed model with TensorFlow[1].

5 EXPERIMENTS

In this section, we conduct extensive experiments on three real-world datasets to evaluate the effectiveness our proposed *EGLN* model.

5.1 Experimental Settings

5.1.1 Datasets Description. To evaluate the effectiveness of our proposed model, we conduct experiments on three public datasets: Movielens-1M [10], Amazon-Video Games [11, 21, 37], and Pinterest [8]. For Movielens-1M, we transform explicit ratings into implicit feedback, where each entry is viewed as 1 only when rating equals 5. For Amazon-Video Games, we adopt the processed dataset [37] which each user have at least 5 records. For Pinterest, we also use the processed dataset [13] which each user have at least 20 records. After data pre-processing, we randomly split historical interactions into training, validation, and test parts with a certain ratio(7:1:2 on Movielens-1M and 8:1:1 on Amazon-Video Games and Pinterest).

5.1.2 Baselines and Evaluate Metrics. We compare our proposed model with several competing baselines which could be categorized into three groups. The first is classical matrix factorization based method BPR [26]. The second is neural graph based CF models with fixed graph structure, including NGCF [33], BiGI [3], LR-GCCF [4] and LightGCN [12]. The last, we compare our model with three graph learning based models, including GAT [31], DropEdge [27] and GLCN[16]. GAT is an efficient graph neural network with edges reweighting by self-attention mechanism. DropEdge is a robust graph learning method by randomly removing edges. GLCN leverages the input features to graph reconstruction. In practice, we use the learned embeddings from the strongest baseline (LightGCN) as the input features.

As we focus on ranking and recommending Top-K items to users, we adopt two widely used ranking metrics: Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG). Specifically, HR measures the number of successfully predicted items in the top-N ranking list that the user likes in the test data. NDCG considers the hit positions of the items and will give a higher score if the hit items are in the top positions. In order to reduce the randomness caused by negative samplings, we select all unrated items as negative samples for each user, and combine them with the positive items that the user likes in the ranking process. For each model, we repeat experiments 10 times and report the average results.

[1]https://www.tensorflow.org

Table 2: HR@k and NDCG@k comparisons for Movielens-1M dataset.

Models	HR@k						NDCG@k					
	k=5	k=10	k=15	k=20	k=25	k=30	k=5	k=10	k=15	k=20	k=25	k=30
BPR	0.14949	0.20061	0.24543	0.28940	0.32519	0.35469	0.13634	0.15521	0.17133	0.18567	0.19667	0.20559
NGCF	0.15477	0.21055	0.26015	0.30107	0.33670	0.36503	0.14149	0.16207	0.17950	0.19293	0.20404	0.21259
BiGI	0.15831	0.21225	0.25930	0.29936	0.33500	0.36708	0.14521	0.16483	0.18124	0.19435	0.20531	0.21477
LR-GCCF	0.15932	0.21160	0.26004	0.29888	0.33284	0.36290	0.14696	0.16595	0.18283	0.19575	0.20654	0.21569
LightGCN	0.16597	0.22381	0.27327	0.31322	0.34957	0.37971	0.15355	0.17439	0.19191	0.20511	0.21645	0.22563
GAT	0.16025	0.21813	0.26682	0.31148	0.34820	0.37818	0.14613	0.16777	0.18487	0.19955	0.21099	0.22021
DropEdge	0.16010	0.21660	0.26509	0.30735	0.34566	0.37785	0.14517	0.16609	0.18326	0.19725	0.20910	0.21887
GLCN	0.16105	0.21989	0.26871	0.31301	0.34752	0.38025	0.14898	0.17080	0.18801	0.20246	0.21338	0.22314
EGLN	**0.16990**	**0.22961**	**0.27905**	**0.31680**	**0.35071**	**0.38335**	**0.15657**	**0.17842**	**0.19565**	**0.20834**	**0.21905**	**0.22857**

Table 3: HR@k and NDCG@k comparisons for Amazon-Video Games dataset.

Models	HR@k						NDCG@k					
	k=5	k=10	k=15	k=20	k=25	k=30	k=5	k=10	k=15	k=20	k=25	k=30
BPR	0.04036	0.06758	0.08729	0.10208	0.11610	0.12741	0.02985	0.03815	0.04372	0.04744	0.05069	0.05319
NGCF	0.05360	0.08268	0.10739	0.12575	0.14194	0.15680	0.03693	0.04680	0.05375	0.05839	0.06217	0.06544
BiGI	0.05587	0.08644	0.11002	0.12871	0.14662	0.16213	0.03893	0.04930	0.05591	0.06062	0.06479	0.06820
LR-GCCF	0.05710	0.08716	0.11064	0.13127	0.14827	0.16383	0.03995	0.05018	0.05680	0.06195	0.06591	0.06936
LightGCN	0.05957	0.09316	0.11709	0.13783	0.15489	0.17145	0.04149	0.05284	0.05955	0.06478	0.06877	0.07242
GAT	0.05614	0.08843	0.11134	0.13070	0.14753	0.16254	0.03918	0.05009	0.05656	0.06142	0.06536	0.06867
DropEdge	0.05365	0.08394	0.10571	0.12417	0.13969	0.15492	0.03717	0.04742	0.05359	0.05825	0.06189	0.06525
GLCN	0.05954	0.09069	0.11530	0.13546	0.15269	0.16845	0.04153	0.05211	0.05900	0.06410	0.06812	0.07157
EGLN	**0.06414**	**0.09754**	**0.12189**	**0.14289**	**0.16023**	**0.17633**	**0.04433**	**0.05567**	**0.06253**	**0.06781**	**0.07189**	**0.07544**

Table 4: HR@k and NDCG@k comparisons for Pinterest dataset.

Models	HR@k						NDCG@k					
	k=5	k=10	k=15	k=20	k=25	k=30	k=5	k=10	k=15	k=20	k=25	k=30
BPR	0.04839	0.08323	0.11308	0.13879	0.16185	0.18311	0.04293	0.05874	0.06993	0.07854	0.08570	0.09191
NGCF	0.04906	0.08399	0.11290	0.13908	0.16193	0.18335	0.04380	0.05966	0.07050	0.07929	0.08638	0.09263
BiGI	0.05003	0.08471	0.11441	0.14004	0.16464	0.18563	0.04457	0.06026	0.07139	0.07997	0.08760	0.09373
LR-GCCF	0.05062	0.08566	0.11469	0.14085	0.16416	0.18616	0.04516	0.06104	0.07192	0.08065	0.08787	0.09430
LightGCN	0.05472	0.09160	0.12354	0.15033	0.17550	0.19775	0.04917	0.06590	0.07785	0.08680	0.09461	0.10112
GAT	0.05170	0.08953	0.12067	0.14807	0.17313	0.19592	0.04581	0.06287	0.07452	0.08368	0.09147	0.09813
DropEdge	0.05390	0.09203	0.12391	0.15168	0.17668	0.20027	0.04762	0.06484	0.07678	0.08607	0.09383	0.10073
GLCN	0.05409	0.09366	0.12596	0.15409	0.17979	0.20323	0.04785	0.06573	0.07780	0.08724	0.09520	0.10205
EGLN	**0.05593**	**0.09468**	**0.12704**	**0.15537**	**0.18112**	**0.20393**	**0.05003**	**0.06756**	**0.07965**	**0.08914**	**0.09713**	**0.10381**

5.1.3 Parameter Settings. For all embedding based models, the free embedding dimension is fixed as 32, and we initialize the embedding matrices with a Gaussian Distribution with the mean value of 0 and the standard variance of 0.01. For those gradient descent-based methods, we use Adam as the optimizing method with a suitable initial learning rate. We stop the model learning process when the performance decreases in the validation data. Similar to many graph based CF models [4, 12, 33], we set the embedding propagation depth K in range of {0,1,2,3,4}, and analyze its impact on the experiment results. Since we have pair-wise ranking based loss, for each observed user-item interaction, we randomly select one unobserved item as candidate negative sample to compose a triple data for training. There are several other parameters in the baselines, we tune all these parameters to ensure the best performance of the baselines for fair comparison.

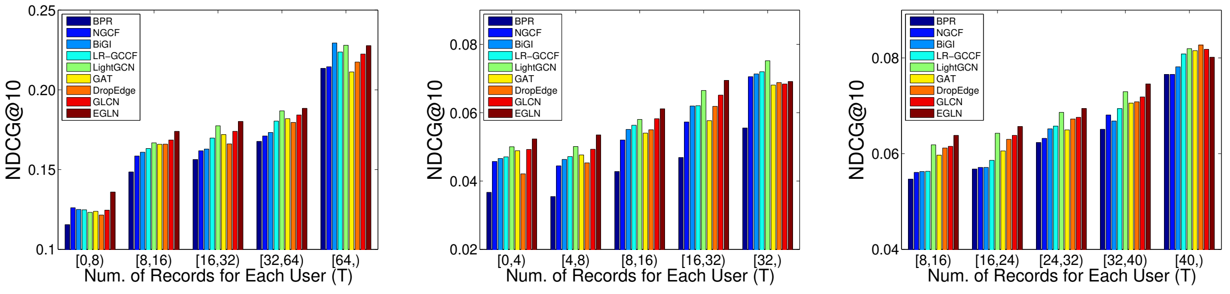
5.2 Overall Performance Comparisons

The overall performance on top-k recommendation as shown in Table2, Table3 and Table4, we report HR@K and NDCG@K values of various models on three datasets. We observe that all graph based models outperform BPR since they encode user/item embeddings with high-order graph structure information, which considerably alleviate the interaction sparsity issue. Specifically, NGCF improves over BPR by injecting the neighbors embeddings for node representation learning. BiGI consistently improves over NGCF by leveraging the global properties of the bipartite graph. It shows the effectiveness of maximizing the local-global graph representation in graph based recommendation. LR-GCCF improves over NGCF by designing residual preference predictions and linear embedding propagation. LightGCN is the best baselines, which removes the excess components: nonlinear activation and feature transformation in GCNs. The performance of these graph based CF models indicates that the simplified neighbors aggregation approach is more efficient for graph based CF.

Apart from comparing the performances on various graph based CF models with fixed structure, we also compare several advanced graph learning based models. For all graph learning based models, we use the same neighbors aggregation mechanism as LightGCN for fair comparisons. In practice, we find that although GAT shows

Table 5: Ablation results of our proposed model (*EGLN* -E denote *EGLN* with fixed graph, *EGLN* -M denote *EGLN* without mutual infomax, *EGLN* -FE denote *EGLN* under fake edge, *EGLN* -FS denote *EGLN* under feature shuffling, *EGLN* -SP denote *EGLN* under structure perturbation).

Models	Movielens-1M		Amazon-Video Games		Pinterest	
	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
EGLN -E	0.21507(-)	0.16650(-)	0.09104(-)	0.05171(-)	0.09190(-)	0.06440(-)
EGLN -M	0.22660(+5.36%)	0.17424(+4.65%)	0.09392(+3.16%)	0.05304(+2.57%)	0.09338(+1.61%)	0.06594(+2.39%)
EGLN -FE	**0.22961(+6.76%)**	**0.17842(+7.16%)**	**0.09754(+7.15%)**	**0.05567(+7.27%)**	0.09402(+2.31%)	0.06711(+4.21%)
EGLN -FS	0.22924(+6.59%)	0.17606(+5.74%)	0.09570(+5.12%)	0.05381(+4.06%)	0.09455(+2.88%)	0.06730(+4.50%)
EGLN -SP	0.22520(+4.71%)	0.17435(+4.71%)	0.09724(+6.81%)	0.05514(+6.63%)	**0.09468(+3.03%)**	**0.06756(+4.91%)**



(a) Movielens-1M (b) Amazon-Video Games (c) Pinterest

Figure 2: Item recommendation performance under different user group.

a better performance than GCN on semi-supervised node classification, it does not show superiority compared with LightGCN on recommendation task. We speculate the possible reason is that the user-item bipartite graph is too sparse, which is not suitable for attentive weight learning. DropEdge is also weaker than LightGCN, since all observed interactions are strong positive signals for model optimization and dropping edges are not effective in recommendation scenarios. GLCN achieves the best performance among these graph learning based models, we revise the weight matrix with the node embeddings learned from LightGCN. However, this method only performs graph learning once, which can not capture the better graph structure for recommendation.

Comparing with all the baselines, we empirically find that our proposed *EGLN* consistently shows the best performance on all the datasets. The detailed improvement rate varies across different datasets, but the same overall trend is shared. E.g., *EGLN* improves 4.70% and 5.36% over the strongest baseline(LightGCN) of HR@10 and NDCG@10 on Amazon-Video Games dataset. It demonstrates the effectiveness of adaptive graph learning to learn better node embeddings to facilitate recommendation. Compared with graph based CF with fixed graph structure, *EGLN* captures the false negative edges on the graph and revises the input graph to better serve CF. Compared to GAT, *EGLN* reweights edges weights with the learned residual graph instead of self-attention. Compared to Dropedge, *EGLN* revises the graph structure by capturing the missing edges instead of randomly dropping edges. Compared to GLCN, *EGLN* achieves the revised graph structure by iteratively performing residual graph learning and node embedding learning modules. The above analysis detailed illustrates the reasons why *EGLN* shows a better performance. Later, we give an ablation study to investigate the role each component plays in *EGLN* .

5.3 Ablation Study of *EGLN*

As shown in Table 5, we report HR@10 and NDCG@10 of *EGLN* and it's simplified version. Specifically, *EGLN* -E denotes we implement *EGLN* with fixed graph ($\alpha = \beta = 0$). *EGLN* -M denotes simplified *EGLN* without the constraint of mutual information maximization ($\beta = 0$). Besides, *EGLN* -FE denotes *EGLN* under fake edge, *EGLN* -FS denotes *EGLN* under feature shuffling, *EGLN* -SP denotes *EGLN* under structure perturbation. Compared with *EGLN* -E, *EGLN* -M achieves obviously improvements on all three datasets, which verifies the effectiveness of the enhanced graph learning. *EGLN* -FE, *EGLN* -FS, and *EGLN* -SP all show better performances than *EGLN* -M, indicating that the local-global consistency optimization is effective to enhance graph learning. In practice, we try three implementations of *EGLN* , and find *EGLN* -FE yields the best performance on Movielens-1M and Amazon-Video Games datasets, and *EGLN* -FS shows the best performance on Pinterest datasets. We speculate the possible reason is that Pinterest dataset contains more interaction records, so fake edge can not provide same weight signals for graph learning as same as another two datasets.

5.4 Performance under Different Data Sparsity

The interaction sparsity issue usually limits the capability of CF, since precise user preference modeling needs sufficient interaction data. To this end, we conduct experiments to explore the performance of various models under different data sparsity. We divide all test users into several groups based on their observed interactions in training data. Taking the Movielens-1M dataset as the example, we split users into five groups, the interaction numbers for each user are more than 0, 8, 16, 32 and 64, respectively. Figure 2 illustrates NDCG@10 of different groups on Movielens-1M, Amazon-Video

Table 6: Performance comparisons of different propagation depth K on three datasets.

Depth K	Movielens-1M		Amazon-Video Games		Pinterest	
	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
K=0	0.20061(-)	0.15521(-)	0.06758(-)	0.03815(-)	0.08323(-)	0.05874(-)
K=1	0.21917(+8.98%)	0.16899(+8.88%)	0.08603(+27.30%)	0.04725(+23.85%)	0.08745(+5.07%)	0.06198(+5.52%)
K=2	**0.22961(+14.46%)**	**0.17842(+15.25%)**	0.09524(+40.77%)	0.05398(+41.49%)	0.09165(+10.12%)	0.06518(+10.96%)
K=3	0.22689(+13.10%)	0.17541(+13.01%)	**0.09754(+44.35%)**	**0.05567(+45.40%)**	0.09292(+11.64%)	0.06607(+12.48%)
K=4	0.22514(+12.23%)	0.17337(+11.70%)	0.09293(+37.51%)	0.05273(+38.22%)	**0.09468(+13.50%)**	**0.06756(+13.77%)**

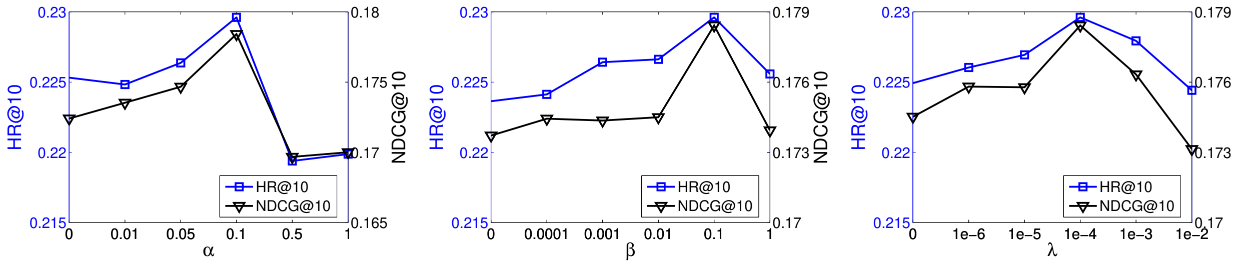


Figure 3: Item recommendation performance with different hyper-parameters on Movielens-1M dataset.

Games and Pinterest datasets. We observe that all models' performances increase when the count of interactions increase, which means high quality user representation needs more interactions. In general, our proposed *EGLN* shows the best performance on most groups but fails in the densest group. We guess the reason is that CF models could achieve good performance with sufficient interactions, so our proposed enhanced graph learning module is not required in this scenes. Besides, we also find that *EGLN* achieves more improvements on sparse groups than dense groups(e.g., 17.754% and 6.674% improvements over BPR on the sparsest and the densest group on Movielens-1M dataset). This indicates *EGLN* is more beneficial to sparse users, because *EGLN* introduces weak supervised signals by adding missing edges which not appear in the input graph.

5.5 Detailed Model Analysis

In this part, we first analyze the effect of different embedding propagation depth K. Followed, we analyze the parameters sensitivities.

5.5.1 The Propagation Layer Depth. In order to investigate the effect of multiple propagation layers, we search the propagation depth K in {0,1,2,3,4}. Please note that, when K=0, the graph convolution part disappears, our model degenerates to BPR. As shown in Table 6, we summarize the experimental results on different propagation layers and compare the performance improvements over BPR. When K increases from 0 to 1, the performance increases quickly on three datasets, showing that the embedding propagation part effectively alleviates the data sparsity issue. As K continues to increase, we find the performance increases at first, and then drops after a certain value. Specifically, our model reaches the best performance with K=2 in Movielens-1M, K=3 in Amazon-Video Games, and K=4 in Pinterest, respectively. The reason is that Amazon-Video Games and Pinterest datasets have more sparse interactions compared to Movielens-1M. For the sparse dataset, deeper graph convolution could help to aggregate more neighbors, which benefits the representation learning. However, for the dense dataset, too deeper propagation layer will easily lead to over-smoothing on the graph.

5.5.2 Parameters Sensitivities. Here we analyze the performances of *EGLN* with different hyper-parameters. Limited to the space, we only show the results on Movielens-1M dataset. As shown in Figure 3, we illustrate HR@10 and NDCG@10 values with three kinds of hyper-parameters: similarity constraint coefficient α, mutual objective coefficient β and regularization coefficient λ. We observe that *EGLN* achieves the best performance with $\alpha = 0.1$, $\beta = 0.1$ and $\lambda = 1e - 4$. For the regularization coefficient λ, we observe that the performance increases when λ increases from 0 to $1e - 4$ and decreases quickly when λ is larger than $1e - 3$. It indicates that suitable regularization could effectively prevent over-fitting issues, however too strong regularization will restrict the model optimization. Same as λ, for the objective balance parameters ha and β, the selections of appropriate parameters are also important for overall objective optimization.

6 CONCLUSION

In this work, we argued that previous graph based CF with fixed graph structure is sub-optimal for user/item embedding learning. Therefore, we proposed an enhanced graph learning network (*EGLN*) approach for better serving CF. To revise the enhanced graph structure, *EGLN* was designed by two folds: First, the enhanced graph learning module and the node embedding module iteratively learned from each other without any feature input. Second, we designed a local-global consistency optimization function to let the enhanced graph capture the global properties in the graph learning process. Finally, extensive experimental results on three real-world datasets clearly demonstrated the superiority and effectiveness of our proposed model.

ACKNOWLEDGEMENTS

This work was supported in part by grants from the National Natural Science Foundation of China (Grant No. 61972125, U19A2079, 61932009, 61732008, 62006066), the Fundamental Research Funds for the Central Universities, HFUT and the Young Elite Scientists Sponsorship Program by CAST and ISZS.

REFERENCES

[1] David C Anastasiu and George Karypis. 2015. L2knng: Fast exact k-nearest neighbor graph construction with l2-norm pruning. In *CIKM*. 791–800.

[2] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. 2018. Mutual information neural estimation. In *ICML*. 531–540.

[3] Jiangxia Cao, Xixun Lin, Shu Guo, Luchen Liu, Tingwen Liu, and Bin Wang. 2021. Bipartite Graph Embedding via Mutual Information Maximization. In *WSDM*. 635–643.

[4] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In *AAAI*. 27–34.

[5] Yu Chen, Lingfei Wu, and Mohammed Zaki. 2020. Iterative Deep Graph Learning for Graph Neural Networks: Better and Robust Node Embeddings. *NIPS* 33 (2020).

[6] Wei Dong, Charikar Moses, and Kai Li. 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In *WWW*. 577–586.

[7] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2019. Learning discrete structures for graph neural networks. In *ICML*. 1972–1982.

[8] Xue Geng, Hanwang Zhang, Jingwen Bian, and Tat-Seng Chua. 2015. Learning image and user features for recommendation in social networks. In *ICCV*. 4274–4282.

[9] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*. 1024–1034.

[10] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *TIIS* 5, 4 (2015), 1–19.

[11] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*. 507–517.

[12] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*. 639–648.

[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[14] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. *ICLR*.

[15] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM*. 263–272.

[16] Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo. 2019. Semi-supervised learning with graph learning-convolutional networks. In *CVPR*. 11313–11320.

[17] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *SIGKDD*. 66–74.

[18] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[19] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*. 3538–3545.

[20] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. 2018. Adaptive graph convolutional neural networks. In *AAAI*. 3546–3553.

[21] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*. 43–52.

[22] Andriy Mnih and Russ R Salakhutdinov. 2008. Probabilistic matrix factorization. In *NIPS*. 1257–1264.

[23] Chanyoung Park, Donghyun Kim, Jiawei Han, and Hwanjo Yu. 2020. Unsupervised Attributed Multiplex Network Embedding. In *AAAI*. 5371–5378.

[24] Seung-Taek Park and Wei Chu. 2009. Pairwise preference regression for cold-start recommendation. In *RecSys*. 21–28.

[25] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. 2020. Graph Representation Learning via Graphical Mutual Information Maximization. In *WWW*. 259–270.

[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.

[27] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. Dropedge: Towards deep graph convolutional networks on node classification. In *ICLR*.

[28] Xiaoyuan Su and Taghi M Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence* (2009).

[29] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2021. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000* (2021).

[30] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph Convolutional Matrix Completion. *STAT* 1050 (2017), 7.

[31] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[32] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep graph infomax. *ICLR* (2019).

[33] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*. 165–174.

[34] Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. 2021. A Survey on Neural Recommendation: From Collaborative Filtering to Content and Context Enriched Recommendation. *arXiv preprint arXiv:2104.13030* (2021).

[35] Le Wu, Junwei Li, Peijie Sun, Richang Hong, Yong Ge, and Meng Wang. 2020. DiffNet++: A Neural Influence and Interest Diffusion Network for Social Recommendation. *TKDE* (2020).

[36] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A Neural Influence Diffusion Model for Social Recommendation. In *SIGIR*. 235–244.

[37] Le Wu, Yonghui Yang, Kun Zhang, Richang Hong, Yanjie Fu, and Meng Wang. 2020. Joint Item Recommendation and Attribute Inference: An Adaptive Graph Convolutional Network Approach. In *SIGIR*. 679–688.

[38] 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 *SIGKDD*. 974–983.

[39] Donghan Yu, Ruohong Zhang, Zhengbao Jiang, Yuexin Wu, and Yiming Yang. 2019. Graph-Revised Convolutional Network. *arXiv preprint arXiv:1911.07123* (2019).

[40] Tianqi Zhang, Yun Xiong, Jiawei Zhang, Yao Zhang, Yizhu Jiao, and Yangyong Zhu. 2020. CommDGI: Community Detection Oriented Deep Graph Infomax. In *CIKM*. 1843–1852.