

분석 23기 미니프로젝트2 논문 발제

LightGCN

Simplifying and Powering Graph Convolution
Network for Recommendation



23기 분석 미니프로젝트2 4조 심심한 사과
윤왕규, 김경민, 백다은, 오영민

CONTENTS •

01 Abstract

02 Introduction

03 Preliminaries

- Collaborative Filtering
- Matrix Factorization
- NGCF

- NGCF

04 Method

05 Experiment

06 Conclusion

- LightGCN

1. ABSTRACT

KEYWORDS

Collaborative Filtering(CF), Embedding-propagation, Graph Neural Network(GNN)



▶ 연구 배경

01

Graph Convolution Network(GCN)은 협업 필터링에서 SOTA이지만, 효율성에 대한 ablation analysis가 불충분함.

* GCN은 graph classification task를 위해 고안되었으며, 많은 신경망 연산을 포함하고 있음.

02

경험적으로 확인한 GCN에서 협업 필터링의 성능을 저하하는 요소

- Feature transformation
- Non-linear activation

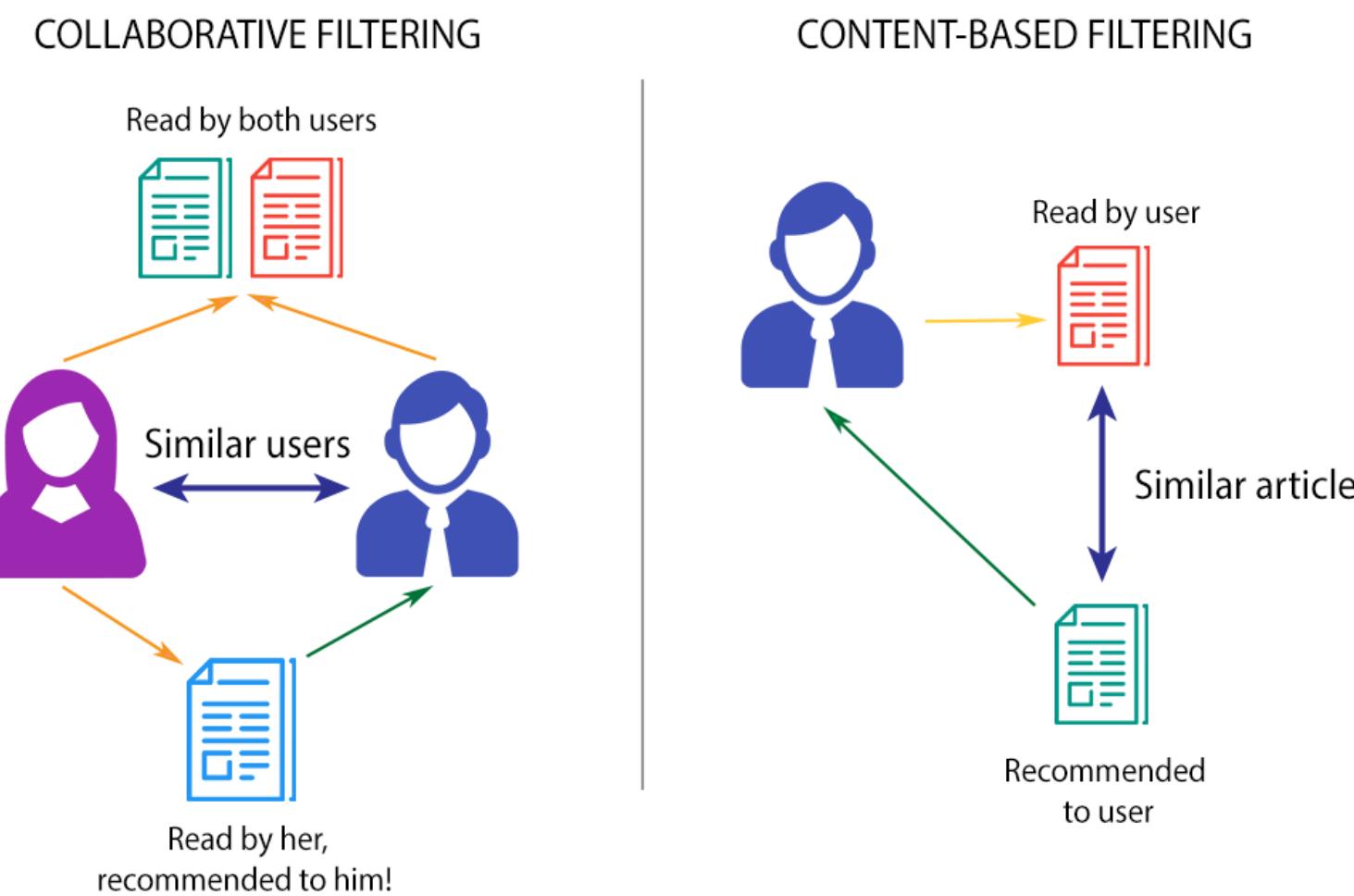
▶ Neighborhood aggregation 구조만을 포함하는 LightGCN을 제안

2. INTRODUCTION

Collaborative Filtering



- ▶ 추천시스템의 핵심은 사용자가 클릭, 평점, 구매와 같은 형태의 상호작용을 통해 특정 상품과 상호작용할지 여부를 예측하는 것.
- ▶ 협업시스템의 경우 위의 예측을 달성하기 위해 과거의 사용자와 상품 간의 상호작용을 활용하는데에 중점을 둔다. 가장 일반적인 패러다임은 user-item 을 적절하게 표현하는 latent features(embedding)을 학습하는 것.



Latent Feature Embedding

1. User-item interaction matrix

Let M and N denote the number of users and items, respectively. We define the user-item interaction matrix $\mathbf{Y} \in \mathbb{R}^{M \times N}$ from users' implicit feedback as,

$$y_{ui} = \begin{cases} 1, & \text{if interaction (user } u, \text{ item } i\text{) is observed;} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

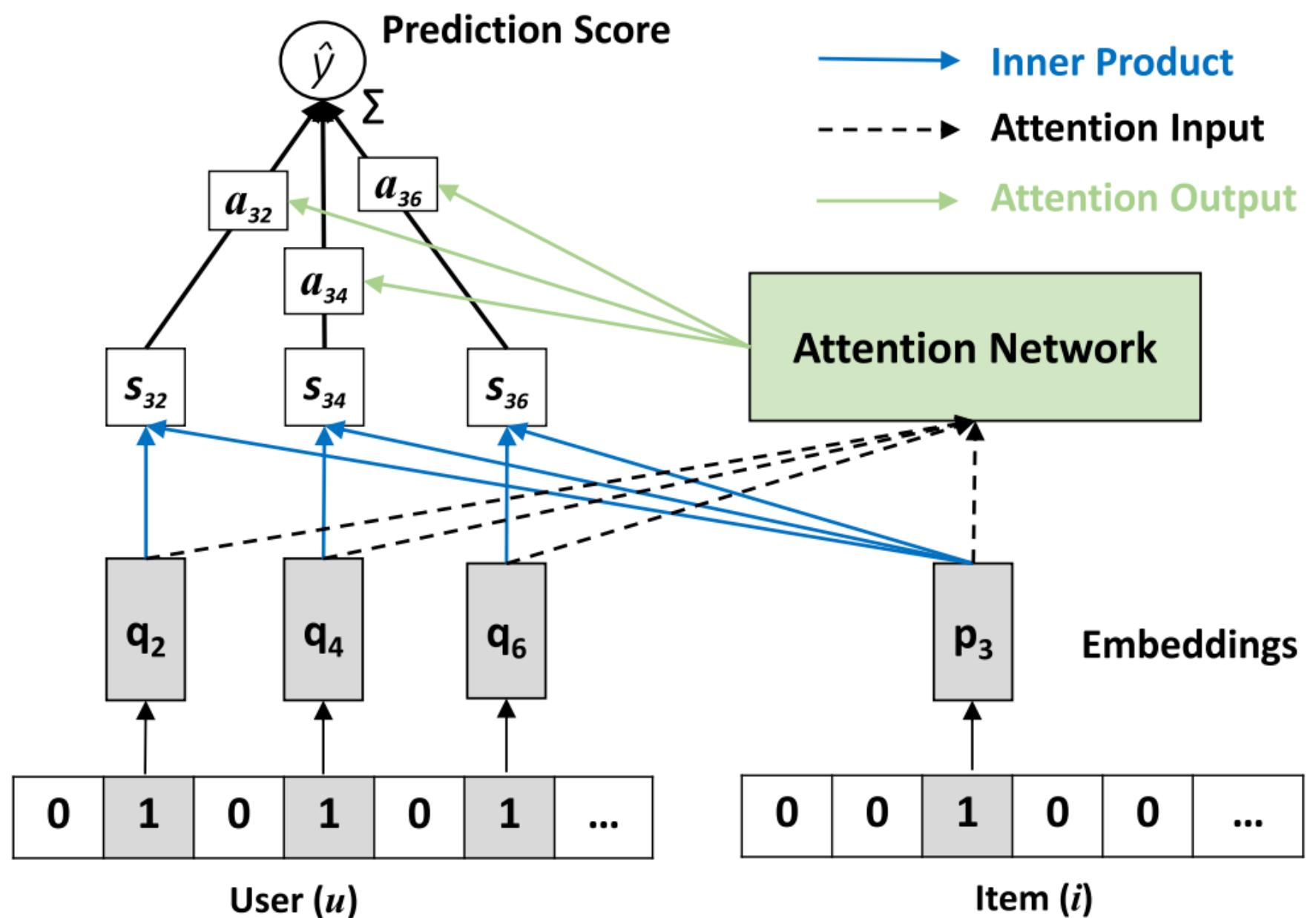
2. Matrix Factorization

$$\hat{y}_{ui} = f(u, i | \mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u^T \mathbf{q}_i = \sum_{k=1}^K p_{uk} q_{ik},$$

where K denotes the dimension of the latent space.

실제 사용자의 선호도를 모델링함에 있어서, 각각의 과거 아이템은 다른
기여도를 가질 것 → 어텐션 메커니즘 적용

3. Neural Attentive Item Similarity(NAIS)



Latent Feature Embedding

▶ 4. Neural Graph Collaborative Filtering (NGCF)

GCN의 propagation rule을 통해 임베딩:

01 Feature transformation

02 Neighborhood aggregation

03 Nonlinear activation

▶ Node classification의 경우, 각 노드들이 rich semantic attribute들을 갖고 있음.

▶ CF의 경우, semantic attribute들을 반영하고 있지 않기에 GCN 구조를 그대로 사용하는 것은 부적절함.

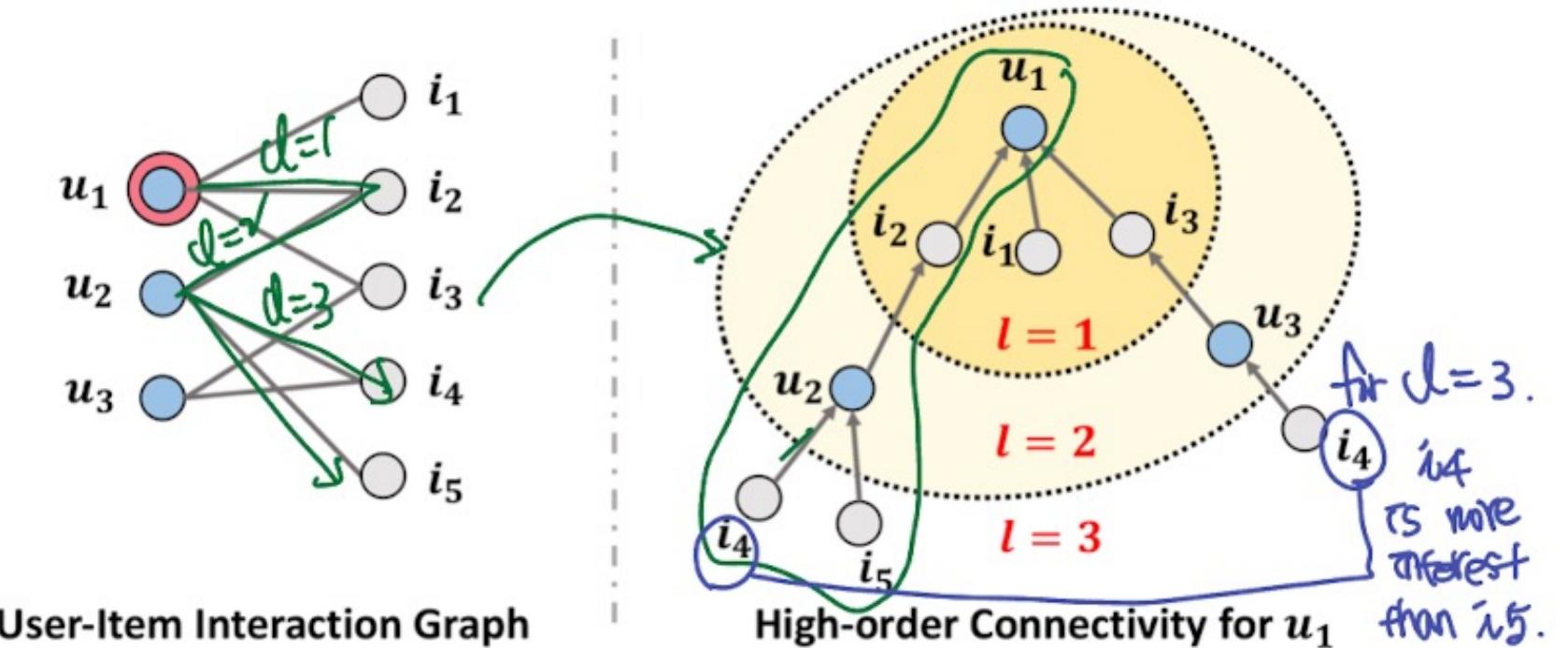


Figure 1: An illustration of the user-item interaction graph and the high-order connectivity. The node u_1 is the target user to provide recommendations for.

3. PRELINIMARIES

Brief NGCF



➤ NGCF의 propagation rule:

$$\mathbf{e}_u^{(k+1)} = \sigma \left(\mathbf{W}_1 \mathbf{e}_u^{(k)} + \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}} (\mathbf{W}_1 \mathbf{e}_i^{(k)} + \mathbf{W}_2 (\mathbf{e}_i^{(k)} \odot \mathbf{e}_u^{(k)})) \right),$$

$$\mathbf{e}_i^{(k+1)} = \underline{\sigma} \left(\mathbf{W}_1 \mathbf{e}_i^{(k)} + \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}} (\mathbf{W}_1 \mathbf{e}_u^{(k)} + \mathbf{W}_2 (\mathbf{e}_u^{(k)} \odot \mathbf{e}_i^{(k)})) \right),$$

1. Message Construction

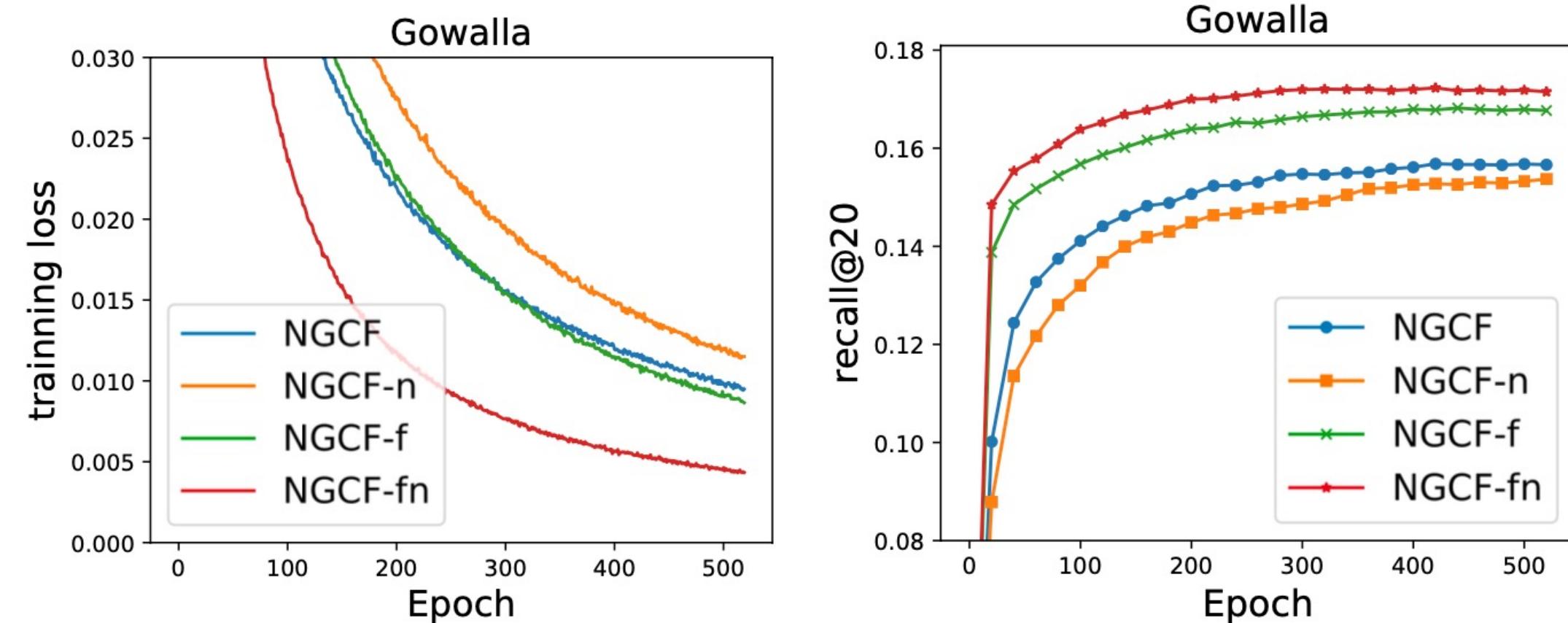
2. Message Aggregation

(High order connectivity) L개의 layer를 통과하여, user embedding, item embedding vector $(\mathbf{e}_u^{(0)}, \mathbf{e}_u^{(1)}, \dots, \mathbf{e}_u^{(L)}), (\mathbf{e}_i^{(0)}, \mathbf{e}_i^{(1)}, \dots, \mathbf{e}_i^{(L)})$ 를 각각 계산.

➤ 각 노드들은 semantic attributes가 없기에, nonlinear activation과 feature transformation은 쓸모가 없음.

3. PRELIMINARIES

Empirical Explorations on NGCF



- NGCF-f: feature transformation 제거
- NGCF-n: nonlinear activation 제거
- NGCF-fn: feature transformation, nonlinear activation 모두 제거

- Feature transformation은 학습에 악영향을 끼친다.
- Feature transformation이 있을 때, nonlinear activation의 유무는 큰 상관이 없으며, feature transformation이 없으면 nonlinear activation을 사용하지 않는 것이 좋음.
- 불필요한 연산의 추가는 모델의 학습을 어렵게 하고, 결과적으로 모델의 성능을 저하시킨다.

- ▶ GCN은 일부 레이블이 지정된 노드를 입력으로 하는 그래프로 가지고, 모든 그래프 노드에 대한 레이블 예측을 하는 Transductive Learning이다. 만약 C개의 레이블을 갖는 분류문제라고 한다면,

$$G = (\mathcal{V}, A),$$

where $\mathcal{V} = \{v_1, \dots, v_n\}$: set of nodes., A : adjacency matrix.

$D \triangleq \text{diag}(d_1, \dots, d_n)$: degree matrix where $d_i = \sum_j A_{ij}$: # interacts for node i .

$X \triangleq (x_1, \dots, x_n)^T \in \mathbb{R}^{n \times d}$ where x_n is feature vector corresponding to node n .

$x_n \sim y_n$ where $y_n \in \{0, 1\}^C$: C-class one-hot encoded vector.

단지 일부 노드에 대한 레이블만이 주어져 있기에, 이를 기반으로 나머지 노드에 대한 레이블을 feature vector x 를 통해 예측하는 문제이다.

- 1. Initialization $\mathbf{H}^{(0)} = \mathbf{X}$,
- 2. Feature propagation(local averaging or smoothing): hidden layer는 각 노드의 이웃 노드에 대응하는 feature vector들의 평균으로 업데이트된다.

$$\bar{\mathbf{h}}_i^{(k)} \leftarrow \frac{1}{d_i + 1} \mathbf{h}_i^{(k-1)} + \sum_{j=1}^n \frac{a_{ij}}{\sqrt{(d_i + 1)(d_j + 1)}} \mathbf{h}_j^{(k-1)}$$

이를 행렬로 나타내면 다음과 같다.

$\mathbf{S} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}$, where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$.

$$\bar{\mathbf{H}}^{(k)} \leftarrow \mathbf{SH}^{(k-1)}$$

- 3. Feature transformation and nonlinear activation: $\mathbf{H}^{(k)} \leftarrow \text{ReLU}(\bar{\mathbf{H}}^{(k)} \Theta^{(k)})$
- 4. Model prediction: $\mathbf{H}^{(k)} \leftarrow \text{ReLU}(\bar{\mathbf{H}}^{(k)} \Theta^{(k)})$

NOTE. 3과 4에서 Simplified GCN(SGCN, Felix Wu et al. 2019)의 경우 nonlinearity는 모델 성능에 영향이 미비하므로 다음과 같이 업데이트 한다:

$$\hat{\mathbf{Y}} = \text{softmax} \left(\mathbf{S} \dots \mathbf{S} \mathbf{S} \mathbf{X} \Theta^{(1)} \Theta^{(2)} \dots \Theta^{(K)} \right) 즉, \hat{\mathbf{Y}}_{\text{SGC}} = \text{softmax} \left(\mathbf{S}^K \mathbf{X} \Theta \right)$$

4. Method

LightGCN

Light Graph Convolution(LGC)



- ▶ GCN의 기본적인 아이디어는 반복적인 graph convolution연산을 즉, neighborhood aggregation이다:

$$\mathbf{e}_u^{(k+1)} = \text{AGG}(\mathbf{e}_u^{(k)}, \{\mathbf{e}_i^{(k)} : i \in N_u\}).$$

- AGG는 aggregation 함수로, graph convolution의 핵심 구조이다.
- AGG에 가중합을 이용한 GIN (Keyulu Xu et al, 2018), LSTM을 이용한 GraphSAGE (Hamilton et al, 2017), bilinear intreaction을 이용한 BGNN (Zhao et al, 2019)등 다양한 방법론들이 제시되었는데, 이는 **모두 feature transformation 또는 nonlinear activation을 포함**한다.
- 이는 semantic input을 갖지 않는 CF task에서는 부담을 줄 수 있다.

- ▶ LightGCN의 aggregator은 다음과 같이 단순히 가중합의 형태를 취한다:

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in N_u} \frac{1}{\sqrt{|N_u|} \sqrt{|N_i|}} \mathbf{e}_i^{(k)},$$
$$\mathbf{e}_i^{(k+1)} = \sum_{u \in N_i} \frac{1}{\sqrt{|N_i|} \sqrt{|N_u|}} \mathbf{e}_u^{(k)}.$$

NOTE 1. User, item embedding vector 모두 같은 normalization term을 갖는다.

NOTE 2. Aggregation 과정에서 자기 자신(self connection)을 포함하지 않는다.

4. Method

LightGCN

Layer Combination and Model Prediction



- ▶ LightGCN에서 학습가능한 파라미터는 초기 상태의 user, item 임베딩 벡터만이 있다. 임베딩 벡터가 초기화 되었으면, K개의 LGC layer를 통과하며 aggregator을 통해 (K+1)개의 임베딩 벡터가 생성된다.

- ▶ 최종 임베딩 벡터는 다음과 같이 각 layer로부터 얻은 임베딩들의 선형결합으로 표현할 수 있다:

$$\mathbf{e}_u = \sum_{k=0}^K \alpha_k \mathbf{e}_u^{(k)}; \quad \mathbf{e}_i = \sum_{k=0}^K \alpha_k \mathbf{e}_i^{(k)},$$

$\alpha_k \geq 0$ 는 해당 논문에서 $1/(K+1)$ 로 모두 균등하게 세팅

- 일반적으로 좋은 성능을 보임.
- 모델 복잡도를 증가시키지 않음.

- ▶ 최종 임베딩을 선형결합으로 나타낸 이유:

1. K가 증가할수록 임베딩이 over-smoothing된다. (과적합을 유발)
2. 각각의 layer의 임베딩들은 서로 다른 semantic feature들을 포착한다. 따라서 이에 대한 선형결합은 더 포괄적인 특징을 포착한다.
3. 선형 결합은 각 임베딩들의 가중합을 볼 수 있으며, 이는 **self-connection을 통한 graph convolution 효과를 포착**한다.

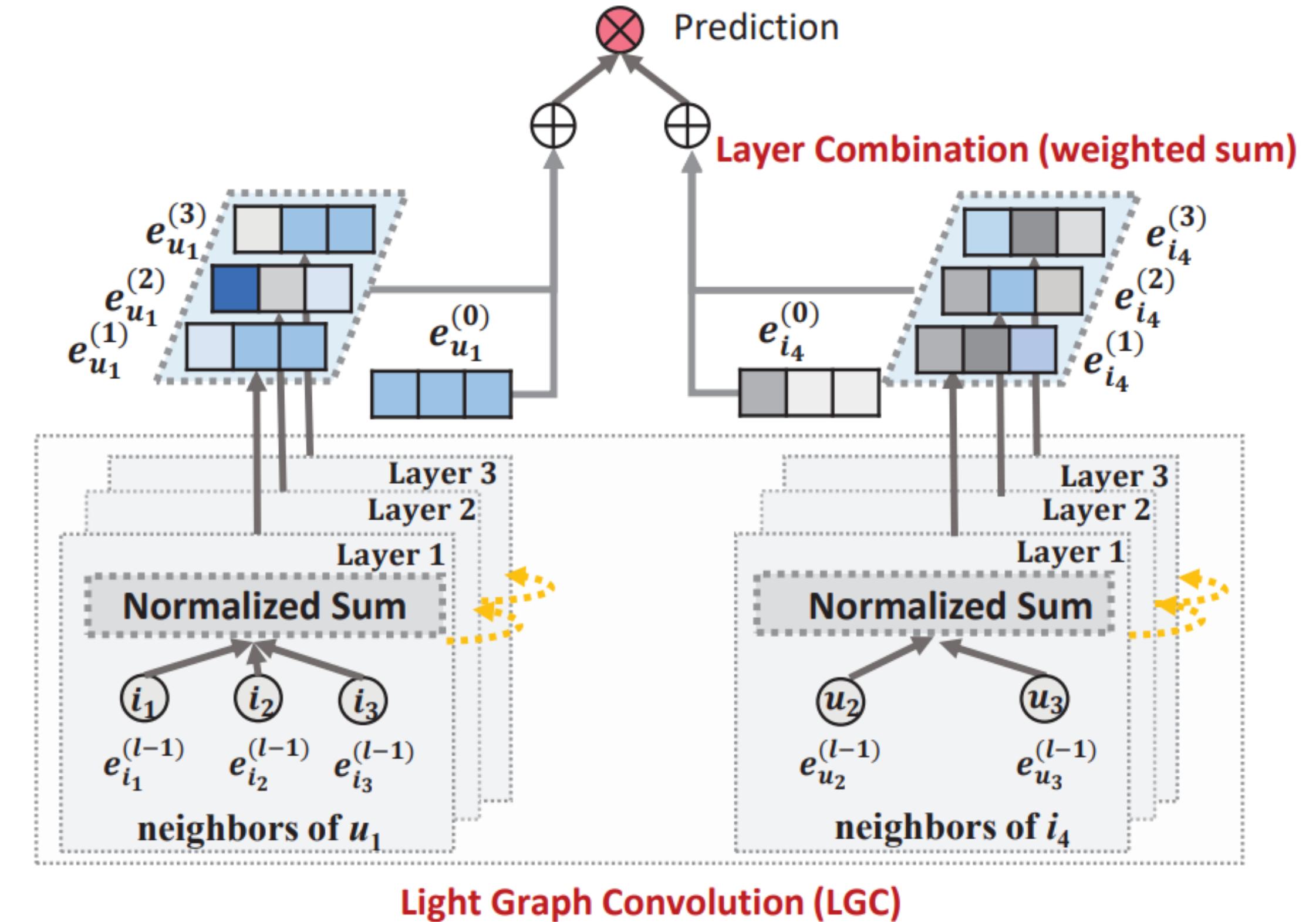
- ▶ Model prediction은 ranking score로 다음과 같다:

$$\hat{y}_{ui} = \mathbf{e}_u^T \mathbf{e}_i,$$

4. Method

LightGCN

Model Architecture



Matrix Form

M : # users, N : # items.

[User-Item interaction matrix].

$$R = \begin{pmatrix} u(1)^T \\ \vdots \\ u(N)^T \end{pmatrix} = \begin{pmatrix} o_i(1) & \dots & o_i(N) \end{pmatrix} \in \mathbb{R}^{M \times N} \text{ where } \begin{aligned} u(m) &= (u(m,1), \dots, u(m,N))^T \\ o_i(n) &= (o_i(n,1), \dots, o_i(n,M))^T \end{aligned}$$

→ If user m & item n are interested,

$$\text{then } u(m,n) = o_i(n,m) = 1.$$

[Adjacency matrix].

$$A = \begin{pmatrix} 0 & R \\ R^T & 0 \end{pmatrix}$$

[Embedding matrix] $E^{(k)} \in \mathbb{R}^{(M+N) \times T}$

$$E^{(k)} = \begin{pmatrix} e_{u(1)}^{(k)T} \\ \vdots \\ e_{u(M)}^{(k)T} \\ e_{o_i(1)}^{(k)T} \\ \vdots \\ e_{o_i(N)}^{(k)T} \end{pmatrix} = \begin{pmatrix} E_U^{(k)} \\ E_I^{(k)} \end{pmatrix}$$

where $e_\alpha^{(k)} \in \mathbb{R}^T$, $\alpha = \{u(1), \dots, o_i(N)\}$.

$$\begin{aligned} E_U^{(k)} &= (e_{u(1)}^{(k)} \dots e_{u(M)}^{(k)})^T \in \mathbb{R}^{M \times T} \\ E_I^{(k)} &= (e_{o_i(1)}^{(k)} \dots e_{o_i(N)}^{(k)})^T \in \mathbb{R}^{N \times T} \end{aligned}$$

Matrix Form

[LGC layer] $\hat{e}_{u(m)}^{(k+1)} = \sum_{\alpha \in N_{u(m)}} \frac{1}{\sqrt{|N_{u(m)}|} \sqrt{|N_{\alpha}|}} e_{\alpha}^{(k)}$. for user-embedding.

equivalently,
 $L: u(m), \alpha = 1$

\rightarrow let $D = \text{diag}(|N_{u(1)}|, \dots, |N_{u(m)}|, |N_{v(1)}|, \dots, |N_{v(N)}|)$.

$$= \begin{pmatrix} D_U & 0 \\ 0 & D_I \end{pmatrix}, \begin{cases} D_U = \text{diag}(|N_{u(1)}|, \dots, |N_{u(m)}|) \\ D_I = \text{diag}(|N_{v(1)}|, \dots, |N_{v(N)}|). \end{cases}$$

$$\rightarrow D^{-1/2} A D^{-1/2} = \begin{pmatrix} 0 & R' \\ R'^T & 0 \end{pmatrix}, R' = D_U^{-1/2} R D_I^{-1/2} \quad \text{i.e. } \left\{ \begin{array}{l} \frac{1}{\sqrt{|N_{u(m)}|} \sqrt{|N_{v(n)}|}} \\ 0 \end{array} \right. \quad \text{if } \sum R'^2_{mn} = 1$$

$$\rightarrow (D^{-1/2} A D^{-1/2}) E^{(F)} = \begin{pmatrix} 0 & R' \\ R'^T & 0 \end{pmatrix} \begin{pmatrix} E_U^{(F)} \\ E_I^{(F)} \end{pmatrix} = \begin{pmatrix} R' E_I^{(k)} \\ R'^T E_U^{(k)} \end{pmatrix}. \quad \therefore E^{(k+1)} = (D^{-1/2} A D^{-1/2}) E^{(F)}$$

4. Method

LightGCN

Matrix Form



▶ 따라서, 최종 임베딩은 다음과 같이 나타낼 수 있다:

$$\mathbf{E} = \alpha_0 \mathbf{E}^{(0)} + \alpha_1 \mathbf{E}^{(1)} + \alpha_2 \mathbf{E}^{(2)} + \dots + \alpha_K \mathbf{E}^{(K)}$$

$$= \alpha_0 \mathbf{E}^{(0)} + \alpha_1 \tilde{\mathbf{A}} \mathbf{E}^{(0)} + \alpha_2 \tilde{\mathbf{A}}^2 \mathbf{E}^{(0)} + \dots + \alpha_K \tilde{\mathbf{A}}^K \mathbf{E}^{(0)},$$

where $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized matrix.

4. Method

LightGCN

Self Connection



- Simplified GCN(SGCN, Felix Wu et al. 2019)과의 관계:

SGCN의 graph convolution:

$$\mathbf{E}^{(k+1)} = (\mathbf{D} + \mathbf{I})^{-\frac{1}{2}} \underline{(\mathbf{A} + \mathbf{I})} (\mathbf{D} + \mathbf{I})^{-\frac{1}{2}} \mathbf{E}^{(k)},$$

항등 행렬을 더한 Self connection 구조

$(\mathbf{D} + \mathbf{I})^{-\frac{1}{2}}$ 항은 scale term이므로, 이후 논의에서는 표기상의 편의를 위해 생략

이항정리를 이용하여 이를 다음과 같이 쓸 수 있다:

$$\begin{aligned}\mathbf{E}^{(K)} &= (\mathbf{A} + \mathbf{I})\mathbf{E}^{(K-1)} = (\mathbf{A} + \mathbf{I})^K \mathbf{E}^{(0)} \\ &= \binom{K}{0} \mathbf{E}^{(0)} + \binom{K}{1} \mathbf{A} \mathbf{E}^{(0)} + \binom{K}{2} \mathbf{A}^2 \mathbf{E}^{(0)} + \dots + \binom{K}{K} \mathbf{A}^K \mathbf{E}^{(0)}.\end{aligned}$$

- 즉, 각 LGC layer의 임베딩들에 대한 선형결합으로 self-connection 구조를 유도해낼 수 있다.

4. Method

LightGCN

Over-smoothing problem



- Approximate Personalized Propagation of Neural Prediction(APPNP, Klicpera et al, 2019)간의 관계:

(Teleport design) APPNP의 propagation layer:

$$\mathbf{E}^{(k+1)} = \underline{\beta} \mathbf{E}^{(0)} + (1 - \beta) \tilde{\mathbf{A}} \mathbf{E}^{(k)},$$

Teleport probability

초기 임베딩과 locality간의 균형을 통해 layer가 깊어질수록 임베딩이 over-smoothing되는 것을 방지

$$\begin{aligned}\mathbf{E}^{(K)} &= \beta \mathbf{E}^{(0)} + (1 - \beta) \tilde{\mathbf{A}} \mathbf{E}^{(K-1)}, \\ &= \beta \mathbf{E}^{(0)} + \beta(1 - \beta) \tilde{\mathbf{A}} \mathbf{E}^{(0)} + (1 - \beta)^2 \tilde{\mathbf{A}}^2 \mathbf{E}^{(K-2)} \\ &= \beta \mathbf{E}^{(0)} + \beta(1 - \beta) \tilde{\mathbf{A}} \mathbf{E}^{(0)} + \beta(1 - \beta)^2 \tilde{\mathbf{A}}^2 \mathbf{E}^{(0)} + \dots + (1 - \beta)^K \tilde{\mathbf{A}}^K \mathbf{E}^{(0)}\end{aligned}$$

- $\alpha_k \geq 0$ 를 적절히 조절하면 마찬가지로 LightGCN을 통한 구현이 가능하다.

4. Method

Model Training

- LightGCN에서 학습가능한 파라미터는 user, item의 초기 임베딩 뿐이며, 계산 복잡도는 기본적인 MF와 같다.

- BPR(Baysian Personalized Ranking) Loss

BPR loss: 관찰된 항목의 예측 점수가 관찰되지 않은 항목보다 높도록하는 pair-wise loss

$$L_{BPR} = - \sum_{u=1}^M \sum_{i \in N_u} \sum_{j \notin N_u} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda \|\mathbf{E}^{(0)}\|^2$$

M: 유저의 수

i: 실제 구매한 item

j: 구매하지 않은 아이템

λ : L2 regularization의 강도를 제어



5. Experiments

Dataset

Table 2: Statistics of the experimented data.

Dataset	User #	Item #	Interaction #	Density
Gowalla	29,858	40,981	1,027,370	0.00084
Yelp2018	31,668	38,048	1,561,406	0.00130
Amazon-Book	52,643	91,599	2,984,108	0.00062

→ NGCF와 동일한 데이터 사용

Performance Comparison with NGCF

Table 3: Performance comparison between NGCF and LightGCN at different layers.

Dataset		Gowalla		Yelp2018		Amazon-Book	
Layer #	Method	recall	ndcg	recall	ndcg	recall	ndcg
1 Layer	NGCF	0.1556	0.1315	0.0543	0.0442	0.0313	0.0241
	LightGCN	0.1755(+12.79%)	0.1492(+13.46%)	0.0631(+16.20%)	0.0515(+16.51%)	0.0384(+22.68%)	0.0298(+23.65%)
2 Layers	NGCF	0.1547	0.1307	0.0566	0.0465	0.0330	0.0254
	LightGCN	0.1777(+14.84%)	0.1524(+16.60%)	0.0622(+9.89%)	0.0504(+8.38%)	0.0411(+24.54%)	0.0315(+24.02%)
3 Layers	NGCF	0.1569	0.1327	0.0579	0.0477	0.0337	0.0261
	LightGCN	0.1823(+16.19%)	0.1555(+17.18%)	0.0639(+10.38%)	0.0525(+10.06%)	0.0410(+21.66%)	0.0318(+21.84%)
4 Layers	NGCF	0.1570	0.1327	0.0566	0.0461	0.0344	0.0263
	LightGCN	0.1830(+16.56%)	0.1550(+16.80%)	0.0649(+14.58%)	0.0530(+15.02%)	0.0406(+17.92%)	0.0313(+18.92%)

*The scores of NGCF on Gowalla and Amazon-Book are directly copied from Table 3 of the NGCF paper (<https://arxiv.org/abs/1905.08108>)

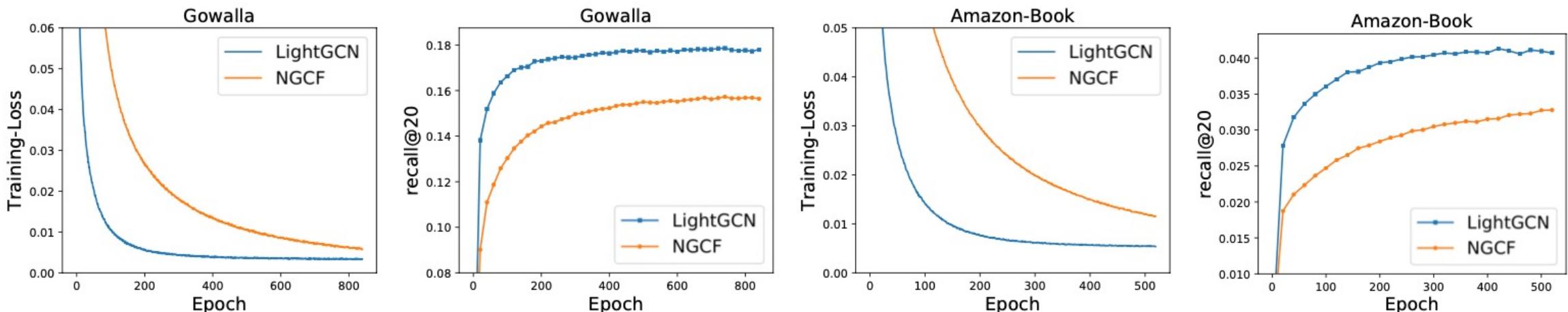


Figure 3: Training curves of LightGCN and NGCF, which are evaluated by training loss and testing recall per 20 epochs on Gowalla and Amazon-Book (results on Yelp2018 show exactly the same trend which are omitted for space).

5. Experiments

➤ Performance Comparison with State-of-the-Arts

Table 4: The comparison of overall performance among LightGCN and competing methods.

Dataset	Gowalla		Yelp2018		Amazon-Book	
Method	recall	ndcg	recall	ndcg	recall	ndcg
NGCF	0.1570	0.1327	0.0579	0.0477	0.0344	0.0263
Mult-VAE	0.1641	0.1335	0.0584	0.0450	0.0407	0.0315
GRMF	0.1477	0.1205	0.0571	0.0462	0.0354	0.0270
GRMF-norm	0.1557	0.1261	0.0561	0.0454	0.0352	0.0269
LightGCN	0.1830	0.1554	0.0649	0.0530	0.0411	0.0315



5. Experiments

Ablation and Effectiveness Analyses



▶ Impact of Layer Combination

- LightGCN: Layer Combination 적용
- LightGCN-single: Layer Combination 적용x, 각각 k번째 임베딩 레이어

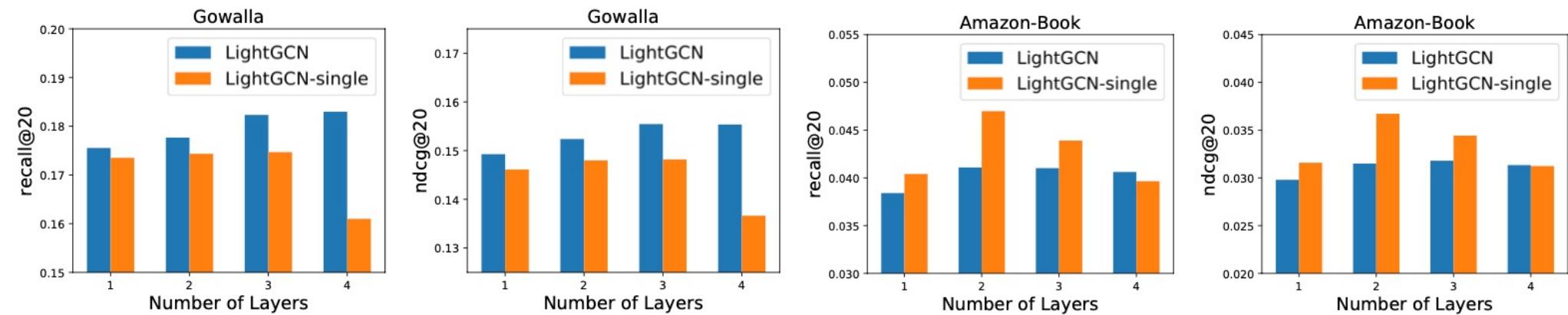


Figure 4: Results of LightGCN and the variant that does not use layer combination (i.e., LightGCN-single) at different layers on Gowalla and Amazon-Book (results on Yelp2018 shows the same trend with Amazon-Book which are omitted for space).

LightGCN-single은 레이어 수가 많아질수록 성능 저하(over-smoothness 발생)
LightGCN은 레이어 많아질 수록 성능이 향상됨
→ Layer Combintion이 over-smoothness 문제를 효과적으로 완화함

*Amazon-Book, Yelp에서는 LightGCN-single 성능이 더 높다?

→ Layer combination에서 $\alpha_k = \frac{1}{K+1}$ 로 uniform하게 설정했는데,
적절한 하이퍼파라미터 튜닝을 통해 해결할 수 있다.

5. Experiments

Ablation and Effectiveness Analyses



▶ Impact of Symmetric Sqrt Normalization

LGC에서 세 가지 normalization 방법을 실험적으로 비교함

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|} \sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)},$$
$$\mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|} \sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)}.$$

- 왼쪽(타겟 노드 계수)만 정규화
- 오른쪽(이웃노드 계수)만 정규화
- 제곱근 제거하고 L1정규화 사용

Table 5: Performance of the 3-layer LightGCN with different choices of normalization schemes in graph convolution.

Dataset	Gowalla		Yelp2018		Amazon-Book	
Method	recall	ndcg	recall	ndcg	recall	ndcg
LightGCN-L ₁ -L	0.1724	0.1414	0.0630	0.0511	0.0419	0.0320
LightGCN-L ₁ -R	0.1578	0.1348	0.0587	0.0477	0.0334	0.0259
LightGCN-L ₁	0.159	0.1319	0.0573	0.0465	0.0361	0.0275
LightGCN-L	0.1589	0.1317	0.0619	0.0509	0.0383	0.0299
LightGCN-R	0.1420	0.1156	0.0521	0.0401	0.0252	0.0196
LightGCN	0.1830	0.1554	0.0649	0.0530	0.0411	0.0315

Method notation: -L means only the left-side norm is used, -R means only the right-side norm is used, and -L₁ means the L₁ norm is used.

- 양쪽 모두에 제곱근 정규화를 사용하는 것(LightGCN)이 가장 좋은 결과를 보임
- 두 번째로 좋은 방법은 왼쪽 side에만 L1 정규화(LightGCN-L₁-L)를 사용하는 것
- 양쪽을 대칭으로 정규화하면 sqrt 정규화에는 도움되지만 L1 정규화는 성능이 저하됨

5. Experiments

Ablation and Effectiveness Analyses



▶ Analysis of Embedding Smoothness

- Smoothness 측정 수식:

$$S_U = \sum_{u=1}^M \sum_{v=1}^M c_{v \rightarrow u} \left(\frac{\mathbf{e}_u}{\|\mathbf{e}_u\|^2} - \frac{\mathbf{e}_v}{\|\mathbf{e}_v\|^2} \right)^2,$$

Table 6: Smoothness loss of the embeddings learned by LightGCN and MF (the lower the smoother).

Dataset	Gowalla	Yelp2018	Amazon-book
Smoothness of User Embeddings			
MF	15449.3	16258.2	38034.2
LightGCN-single	12872.7	10091.7	32191.1
Smoothness of Item Embeddings			
MF	12106.7	16632.1	28307.9
LightGCN-single	5829.0	6459.8	16866.0

- 2-layer LightGCN은 MF와 비교했을 때 smoothness loss가 훨씬 낮음
- LGC를 통해 임베딩이 더 smoothing 되고, 추천에 더 적합해짐

6. Conclusion

- 본 논문은 NGCF에서 불필요한 요소(feature transformation, nonlinear activation)를 제거하여 더 가볍고 추천에 용이한 LightGCN 모델을 제안함
- LightGCN은 Light Graph Convolution, Layer Combination로 구성됨
- 초기 레이어만 학습 파라미터로 사용하여 가볍고 효율적인 모델
- Light Graph Convolution은 feature transform, nonlinear activation을 제거하여 간결한 모델을 만듦
- Layer Combination은 노드의 최종 embedding을 모든 layer embedding의 weighted sum으로 만듦. 이를 통해 self-connection 효과를 갖고, over smoothing 문제를 예방함



분석 23기 미니프로젝트2 논문 발제

감사합니다